Tallinna Pedagoogikaülikool
Matemaatika-loodusteaduskond
Informaatika osakond

Triin Lichfeld

BC4J - Java ärikomponentide algõpetus Oracle9*i* JDeveloper arenduskeskkonna baasil

Bakalaureusetöö

Juhendaja: Jaagup Kippar

Autor: ..... “.....” 2004
Juhendaja: ..... “.....” 2004
Osak. juhataja: ..... “.....” 2004

Tallinn 2004

Sisukord

Sissejuhatus

Leidub mitmeid andmebaasisüsteeme ja programmeerimiskeeli, mida ühel või teisel määral kasutatakse paljude tarkvaraliste infosüsteemide väljatöötamisel. Üheks enamlevinumaks nendest on Java programmeerimiskeel, võimaldades luua oma rikkaliku funktsionaalsusega mistahes rakendusi ja täita klientide lõputuid nõudmisi. Samuti pole olemas ühtegi suuremat infosüsteemi, mis ei põhineks mõnel kindlal ja turvalisel andmebaasisüsteemil.

Mitmete aastate jooksul on programmeerijate tööd hakanud lihtsustama mitmed visuaalseid komponente pakkuvad integreeritud arenduskeskkonnad, mis võimaldavad luua standarditel baseeruvaid mitmekihilisi rakendusi. Üheks selliseks on Oracle9*i* JDeveloper koos oma paljude integreeritud vahenditega.

Käesoleva töö eesmärk on pakkuda õppevahendit paberkujul, mis toob huvilise lähemale Oracle tehnoloogia pärusmaadele, ning pakub kasutajale algõpet Oracle9*i* JDeveloper arenduskeskkonna vahendusel tänapäeval üha aktuaalsust koguvate Java ärikomponentide (*Business Components for Java*) kasutamise võimaluste kohta.

Õppematerjal on koostatud järgmistel põhimõtetel:

- kasutaja peab saama ülevaate algelementide kohta, milleks on põhitõdede tutvustamine ja seletamine
- kasutajal peab olema võimalik jälgida õpetuse käiku reaalsete näidete ja seletuste varal
- kasutaja saab ise paralleelselt teoreetilise poole omandamisega, luua ka töötava rakenduse.

Õppematerjali valmimisel on olnud suur tähtsus informaatika eriala eelnevalt läbitud ainetel. Idee luua selline õppematerjal pärineb paljuski ainest “Infosüsteemid”, mille õppejõuks mulle oli õpetaja Priit Parmakson. Samuti lisades, et kogu töö põhineb Java programmeerimiskeelel, on õpetaja Jaagup Kippari poolt jagatud teadmised Java võimalustest siinkohal asendamatud.

Töö on ülesehitatud kuuele peatükile, mis vajadusel jagunevad alapeatükkideks.

Esimeses peatükis on antud hinnanguline ülevaade senise seisu kohta olemasolevate õppematerjalide osas. Peatükid 2-4 toovad lugeja lähemale arenduskeskkonna üldistele nüansidele, seletavad selle vajalikkust ja esmaseid töid, mis on vajalikud keskkonna ülesseadmiseks. Samuti kuuluvad vaatluse alla arenduskeskkonna enamkasutatavad ja algajale kasutajale vajaminevad tööriistad ning sisseehitatud viisardite kasutamine. Järgnevalt

toimub süvenemine ärikomponentidesse ja äriloogikasse. Üliõpilased, kes on läbinud aine "Andmebaaside projekteerimine" või "Infosüsteemid", peaksid olema andmebaaside ja andmemudelitega enamjaolt tuttavad.
Peatükkide 5 ja 6 alla on koondatud kaks praktikumi. Esimene praktikum on peaasjalikult ülesehitatud mõttel lasta kasutajal, kasutades arenduskeskonna kõrgema taseme vahendeid, ise algusest lõpuni töötav andmebaasirakendus valmis ehitada ning seejärel teha see kliendiplatvormile paigaldamiskõlblikuks. Töö põhineb enamjaolt genereerivatel vahenditel. Teine praktikum on keerulisem ja üritab tuua kasutaja lähemale arenduskeskkonna mootorile. Kasutatakse vähem genereerivaid vahendeid.

Kogu materjali loomisel on kasutatud tarkvaraliste vahenditena Oracle9*i* JDeveloper 9.0.3 versiooni arenduskeskkonnana ja Oracle9*i* Enterprise Edition Standard paketi andmebaasi haldusvahendeid ning serveritarkvara.

1 Olemasolevate õppematerjalide tutvustus ja võrdlus ning soovitused kasutamiseks

Olemasolevate materjalide alla kuulub palju BC4J Java ärikomponentide ja JDeveloper arenduskeskkonna õppematerjale, mis on lihtsalt kättesaadavad Oracle enda kodulehelt. Probleemne on aga emakeelse, see on eestikeelse materjali leidmine. Vaatamata otsingumootorites www.neti.ee, www.google.com mitmel erineval kujul päringuid tehes, märgatavaid tulemusi ei ilmnenud. Küll aga korraldatakse eesti IT-koolitusfirmade poolt üksikuid kursusi. Üheks taolise koolituse pakkujaks on "Adobase System", kes oma Java kursusesse on lisanud ka rakenduste programmeerimise JDeveloper arenduskeskkonna vahendusel Adobase Systems - Java's programmeerimine [25.04.2004]. Idee iseenesest hea, kuid kehv õppurile, kel puudub piisav finantstugi.

Inglisekeelse materjali osas on internet tunduvalt rikkalikum. Oracle kodulehelt on võimalik vaadata *flash* esitlusi JDeveloper keskkonna kasutamisest [1], ning hiljem ise järele proovida. Samuti leiab sealt otsingu abil hulganisti kirjandust ja metoodikat, mida Oracle korporatsioon iga oma tootega kaasa paneb.

Õppematerjalid (*Tutorial*) põhinevad tavaliselt mingitel Oracle andmebaasipaketiga kaasatulevatel skeemiobjektidel (andmetabelitel) ja internetist saada olevatel materjalidel, mida enamasi Oracle ise õppematerjalina pakub. Nende tabelite põhjal hakatakse õpetuste ja soovituste järgi JDeveloper'is näiteks BC4J ärikomponentidest koosnevat rakendust kokku panema. Esimest korda JDeveloper'ga ja ärikomponentidega kokkupuutuva kasutaja jaoks on Oracle poolt internetipõhised õpetused liialt keerulised. Enamasti on kogu süsteemi tabelisüsteem juba alguses keeruline mõista (palju piiranguid, indekseid, linkimisi, võtmeid). Tihemini kohtab internetis viiteid õppematerjalile "*Toy Store Demo*" [2] [6] (kujutab endast internetipõhist mänguasjade poodi) ning "*Online Orders*" (interneti tellimused). Esimene neist kahest on raske ja nõuab algajalt kasutajalt tunduvalt rohkem teadmisi kui algajale kohane. Teine seevastu on lihtsam ja arusaadavam.

Tähelepanu tasuks pöörata ka kirjandusele. Kuna Eesti kauplustest niisama lihtsalt kirjandust JDeveloper'i ja BC4J Java ärikomponentide kohta ei saa, siis leidub üksikuid tarkvaraarendusega tegelevaid firmasi, kelle raamatukogudest hea tahtmise korral vajaliku raamatu laenata saab.

Üheks selliseks oli "Oracle9*i* JDeveloper Handbook", mille ülesehitus oli väga kompaktne ja sisutihe, sõltumata sellest, kas seda käsiraamatut kasutab iseõppija või on ta kasutusel

kursuste abimaterjalina. Palju oli konkreetseid näiteid, mis olid kirjeldatud sammhaaval, et kasutaja saaks samal ajal ise kõike kaasa teha [9]. Seletatud oli piisavalt, tihti toodi välja alternatiivseid meetmeid, kuidas ühte või teist operatsiooni läbi viia, kust komponente leida ja palju muud. Teoreetiline osa oli lihtsalt seletatud, ei tekkinud probleemi võõrkeelsest tekstist arusaamisega, kuna seletused olid pigem pikemad kui konkreetsed ja raskesti sõnastatud (paljude internetist saadaolevate materjalide mureks on nende keeruline sõnastus, mida oleks arvatavasti ka emakeeles keeruline mõista). Taoline kirjeldav ja seletav ülesehitus on kasutusele võetud ka käesoleva õppematerjali loomisel.

Õppematerjal on mõeldud kasutamiseks nii iseõppijale kui ka juhendaja käe all õppijale. Soovitav oleks omada eelnevaid teadmisi või läbitud kursusi Java, JSP ja samuti andmebaaside kohta. Peaks teadma, millel põhinevad relatsioonilised andmebaasid. Võõras ei tohiks olla ka infosüsteemide üldine loomise metoodika. Kuna töös on põgusalt näidatud, kuidas UML[1] abiga on võimalik rakendusse loogikat lisada, siis tuleks kasuks tutvuda ka UML võimalustega [8].

Juhendajal on soovitav eelnevalt materjaliga tutvuda. Teoreetiline ja praktiline osa on siinses materjalis eraldatud, kuid annab juhendajale valikuvabaduse, kas alustada teoreetilise osa tutvustamisega praktilise töö toel, või lähtuda põhimõttest, et eelnevalt anda teadmisi süsteemi struktuuri kohta ning hiljem neid praktiseerida. Kogu materjali maht on arvestatud ligikaudu 20 aktiivse tunni peale, mis sisaldab teooriaga tutvumist ja praktikumide läbimist.

[1] UML – *Unified Modeling Language* (unifitseeritud modelleerimiskeel). UML kohta on võimalik lugeda Liina Lang'i bakalaureusetööd "UML-i õppematerjal"

2 Oracle9*i* JDeveloper arenduskeskkonna tutvustus, vajalikkus ja installatsioon

Kui on vaja leida vahendit, millega saaks paremini luua rakendusi, mis on mõeldud töötama kas interneti või intraneti vahendusel ning mille üheks peamiseks aluseks on Oracle andmebaasid, siis on optimaalne valida arenduskeskkonnaks Oracle korporatsiooni poolt pakutud arendusvahend Oracle9*i* JDeveloper, millel on spetsiaalselt Oracle enda tehnoloogia tarbeks integreeritud kõik vajaminevad funktsioonid ja vahendid.

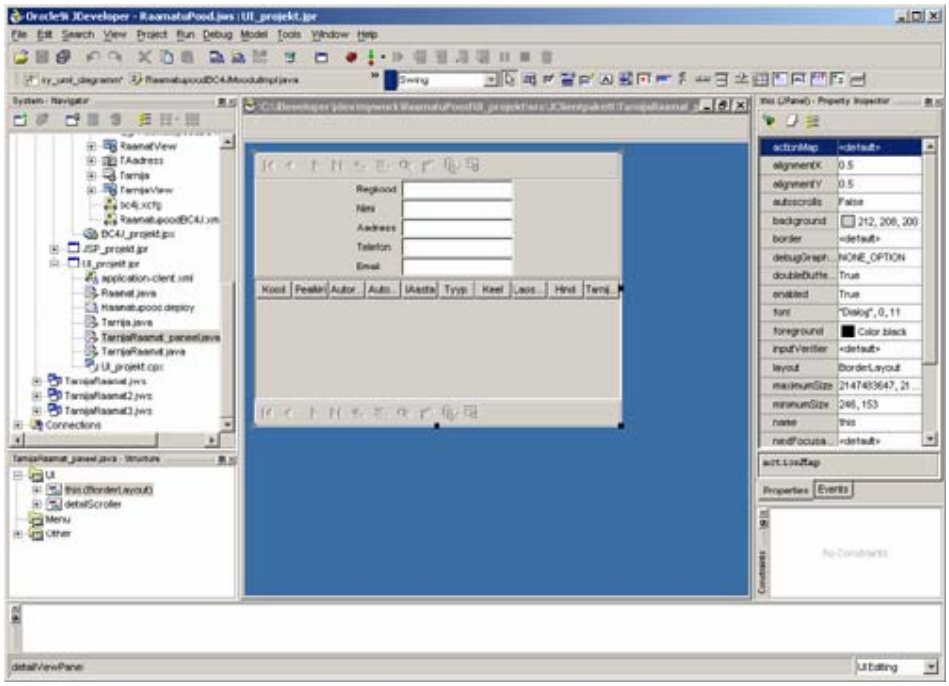
Oracle9*i* JDeveloper on integreeritud J2EE[2] arenduskeskkond, mille eesmärgiks on arendajatele pakkuda mitmekülgset tuge rakenduste loomiseks, silumiseks, seejärel Java, XML[3] ja SQL[4] rakenduste paigaldamiseks serveritele ning Web'i teenusteks. Kirjutatud on ta Java keeles.

Aastate eest, kui veel palju programmeerimistöid sai arendamise käigus tehtud käsitsi, on nüüdseks võetud kasutusele ohtralt visuaalseid *builder*-tüüpi vahendeid nagu ka Oracle9*i* JDeveloper – eelduseks programmeerijate vaeva vähendamine pikkade koodiridade kirjutamisel ning seeläbi aja ja rahaliste ressurside märgatav kokkuhoidmine. Praegusel ajal on just turule ilmunud ka JDeveloper 10*g* Preview, mis oluliselt eelmisest versioonist ei erine. Muutunud on veidi väljanägemine, lisandunud osaliselt rakenduste malle, kuid kõik operatsioonid ja loogika, mida käesolevas õppematerjalis käsitletakse on analoogne eelneva, see on Oracle9*i* JDeveloper arenduskeskkonnaga (vt. ka Joonis 1).

Oracle9*i* JDeveloper on täielikult ehitatud J2EE 1.3 versiooni toetusele omades ka EJB 2.0[5], Servlet 2.3[6] ja JSP 1.2[7] toetust, millest siin õppematerjalis peaks täielikult piisama algteadmiste omandamiseks [3]. Arenduskeskkond toetab kõige uuemaid Interneti standardeid ja vabavara vahendeid. Kõige olulisem aga siin punktis on J2EE raamistik, mida JDeveloper pakub kui väga laialdaste võimalustega Java ärikomponentide (BC4J[8]) paketti. Nende

[2] J2EE (Java 2 Platform, Enterprise Edition) – Sun Microsystems Java platvorm mitmekihiliste serverorienteeritud firmarakenduste jaoks. J2EE aluseks on Enterprise JavaBeans (EJB).
[3] XML (Extensible Markup Language) – laiendatav märgistuskeel, mis omab rohkem funktsionaalsust kui HTML.
[4] SQL (Structured Query Language) – pooleldi standardiseeritud päringukeel, mida toetab enamik andmebaasihaldureid.
[5] EJB (Enterprise JavaBeans) – äri java uba, mis defineerib mitmeastmeliste klient/serversüsteemide komponentarhitektuuri.
[6] Java servlet – väike serveril käitatav Java programmikeeles kirjutatud programm.
[7] JSP (Java Server Pages) – tehnoloogia dünaamiliste veebilehtede loomiseks ja need veebipõhised rakendused on sõltumatud serveritest ja muudest platvormidest (25.04.2004 http://java.sun.com/products/jsp/overview.html).
[8] BC4J (Business Components for Java) – Java ärikomponendid

komponentide peamine ülesanne on lihtsate ja loogiliste vahenditega pakkuda suuri J2EE rakendusi, mis on võimelised töötama erinevatel platvormidel.



Joonis 1 - JDeveloper arenduskeskkonna vaade

Oracle9*i* JDeveloper on uuema generatsiooni arenduskeskkond. Võrreldes temale eelnenud vahendiga, mille versiooniks oli JDeveloper 3.2, omab praegune kasutuses olev 9.0.3 versioon näiteks järgimisi uusi võimalusi:

- laiaulatuslik tugi Struts'i veebiarenduses (Strutsi raamistiku kohta on võimalik lugeda Mikk Normaku proseminaritööst ja diplomitööst)
- võimas sisseehitatud andmebaasiliides, mis võimaldab ehitada tabeleid, vaateid ja päästikprotsesse ja oskab tegeleda ka kolmanda osapoole draiveritega
- viisardid ja funktsioonid BC4J ärikomponentide kasutamiseks läbi kolme põhikihi
- JClient vahend kliendivaadete loomiseks, mis pakub mitmeid erinevaid võimalusi liideste lisamiseks loodavasse kliendirakenduse kasutajaliidesesse.

JDeveloper'i integreeritud arenduskeskkonna esivanemaks on Borland JBuilder, tuntud Java arenduskeskonnana. Aastal 1997 ostis Oracle Borland JBuilder'ilt osa koodi [7], et oma versioon välja töötada. Juhtus aga nii, et esimesed kaks versiooni olid rohkem või vähem kasutud ja ebaefektiivsed. Järgnevale, kolmandale versioonile, lisati BC4J Java ärikomponendid. See oli JDeveloper'i väljakujunemise pöördepunktiks. Pärast veel ühte

parandustega versiooni, tuli välja konkreetne, õige ja võimas arendusvahend Oracle9*i* JDeveloper.

Oracle9*i* JDeveloper'i kättesaadavus on lihtne. Vabavarana on ta saadaval Oracle kodulehelt eratarbeks ja õppeotstarbel, eeldusel, et kasutajaid sellele tarkvarale on vaid üks. Installeeritakse ta lahti pakkimise teel juurkataloogi. Valida on kahe erineva paketi vahel: standard ehk kõigi vahenditega pakett ja baaspakett. Baaspaketi erinevus standardpaketist seisneb osade instantside algupärases puudumises paketist, kuid vastavalt vajadusele on neid lihtne õpetuste järgi sinna paigaldada. JDeveloper ei vaja erinevalt mõnest muust Oracle arendusvahendist aluseks andmebaasiserveri klienti, vaid on täiesti sõltumatu.

Oracle9*i* JDeveloper (versioon 9.0.4.0) installatsioonipaketti on võimalik alla laadida aadressilt http://otn.oracle.com/software/products/jdev/htdocs/soft904w.html (olles eelnevalt ennast registreerinud): jdev904.zip (suurus 199MB). Süsteeminõuded [4] on esitatud järgmiselt:

- Operatsioonisüsteem – Windows 2000 SP3, Windows NT 4.0 SP6a, Windows XP SP1
- Protsessori tüüp ja kiirus (alates) – Pentium III 500 MHz
- Mälu – 512 MB RAM

Täielik paigaldus võtab anda alla kõvakettaruumi ligikaudu 400 MB.

3 JDeveloper'i integreeritud arenduskeskkonna ülevaade

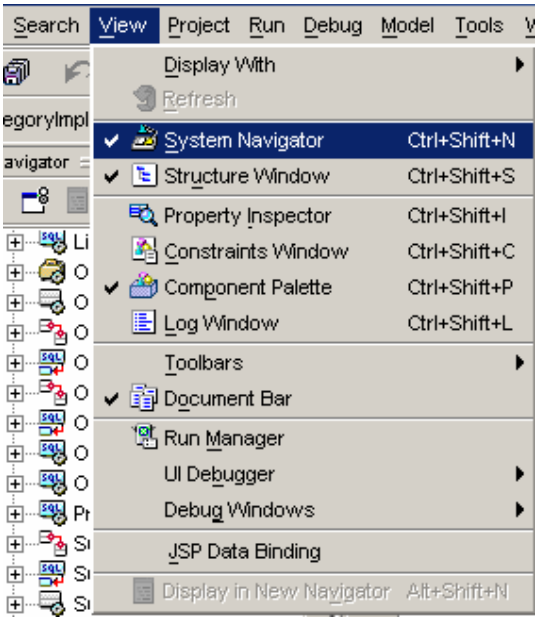
See peatükk on mõeldud JDeveloper'i integreeritud arenduskeskkonna (edaspidi IDE) vahendite ja viisardite tutvustamisele. Kuna vahendeid ja viisardeid on äärmiselt palju ning kõigi nendega pole algajal või veidi edasijõudnumal tegijal midagi tarka peale hakata, siis tulevad vaatluse alla vahendid, mida reaalselt võiks algajal kasutajal vaja minna või mille kohta võiks kasutajal olla teadmine, et nad üleüldse olemas on.

3.1 Vahendite tutvustus

IDE koosneb mitmetest halduritest ja komponentidest. Lisaks on tal mitmed erinevad paneelid komponentidega, mida mingi arendusetapi juures on tarvilik kasutada. Suur tähtsus on WYSIWYG[9] põhimõttel, mis väljendub sisseehitatud visuaalsel keskkonnal, kus saab komponente kokku lohistada ja nendele funktsioone määrata.

Järgnevalt teeme ülevaate komponentidest, mida on võimalik leida JDeveloper'i tööriistaribadelt ja paneelidelt.

Paljusid igapäevases töös vajalikke vahendeid saab kätte, kasutades selleks *View* menüüd (vt. Joonis 2).

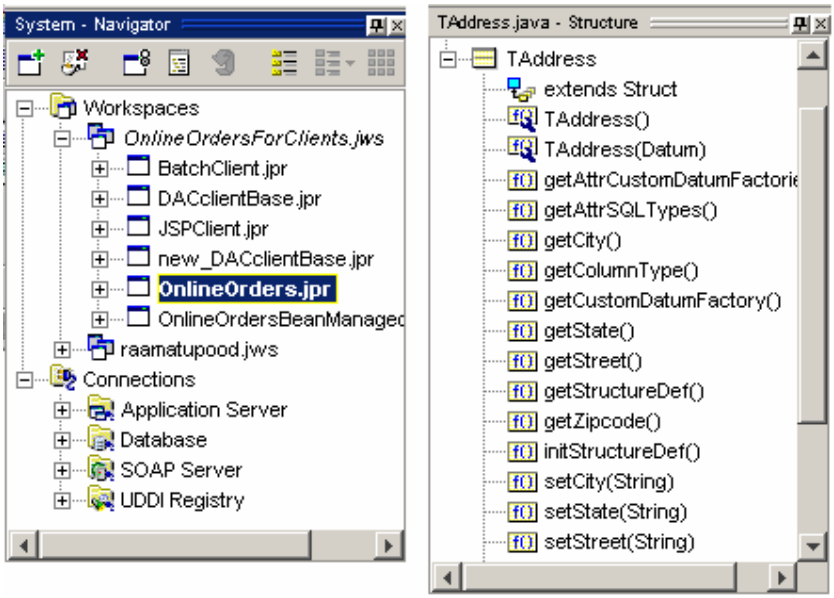


Joonis 2 - Valikud View menüüs

[9] WYSIWYG (What You See Is What You Get) – see, mida näed, selle ka saad.

View menüüs on tähtsamateks vahenditeks *System Navigator* ja *Structure Window* (vt. Joonis 3).

System Navigator'is ehk süsteemi navigeerimisaknas ehk süsteemipuus asuvad tööruumid ja nendesse kuuluvad projektid oma andmefailidega. Samuti ka ühendused, mida saab teha erinevatesse andmebaasidesse või rakenduseserveritesse. Enamus vajalikest toimingutest käib läbi süsteemipuu. Siit saab käivitada ka viisardeid.



Joonis 3 - Süsteemi kataloogi ja faili puu ning struktuuri akna vaade

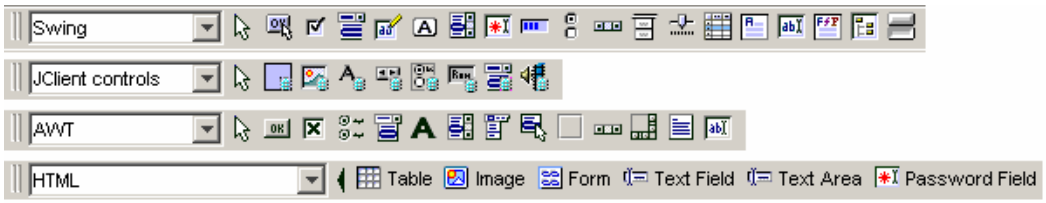
Tehes hiirega kahekordse klõpsu andmefailidel või lohistades neid redaktori aknasse, avaneb võimalus ka nende sisu vaadata, samuti täiendada neid faile mõne vajaliku koodireaga. Süsteemipuus olevad kaldkirjas failid tähendavad seda, et nad on salvestamata. See ilmneb siis, kui ühte või mitut faili on muudetud.

Struktuuri aknas (*Structure Window*) on võimalik näha, millistest meetoditest ja imporditud pakettidest näiteks koosneb üks või teine parajasti aktiivne olev fail. Tavaliselt on otstarbekas sellist struktuuri akent vaadata erinevate `.java`, `.xml` ja `.jsp` failide sisu täpsustamise puhul. Selle kaudu on väga lihtne leida koodis üles vajalik järg, ilma et peaks vahel sadadesse ridadesse ulatuvat koodi edasi tagasi kerima. Samuti oskab ta viidata võimalikele vigadele või puudustele, mis võivad olla olulised projekti osade kompaktsel koostööl.

Peale eelpool kirjeldatud kahe vahendi on võimalik *View* menüü abil avada ka rikkaliku valikuga komponentide palette. Nad sisaldavad peaaegu kõiki vajaminevaid visuaalseid vahendeid, et lihtsustada programmeerimise tööd. Vastav Java kood genereeritakse juba

JDeveloper'i enda loogika tasemel. Tavaliselt ei pääse ühegi keskmise suuruse ja keerukusega andmebaasi rakenduse puhul mööda *Swing*'i ja *JClient*'i komponentide palettist. Samuti, kui on soov luua veebiliidest sisaldavat rakendust, läheb vaja kindlasti komponente nii HTML kui ka JSP varamust, millel omakorda on sees rida muid komponente. Eraldi suur osa komponente kuulub BC4J komponentide varamusse.

Tavaliselt avaneb komponentide palett IDE paremas servas, või üleval redigeerimise akna kohal tööriista ribal (vt. Joonis 4). Avada võib neid ise, valides *View/Component Palette* või tavalisem variant on, et vastav tööriistu pakkuv aken avaneb siis, kui on antud mõista projekti mingit uut faili luues, et soovitakse kasutada ühtesid või teisi komponente. Näiteks soovitakse luua *JClient* rakendust, siis vastava nõude esitamisele avaneb vastav fail koos paremas küljes asuvate vahenditega.



Joonis 4 – Java kasutajaliidese visuaalsed vahendid tööriistaribadel

Lisaks sellele, et rakendusi saab kokku panna mitmeid komponente kasutades, on tähtis roll ka teistel JDeveloper'i menüüdes pakutavatel vahenditel. Kompileerimise ja käivitamise korral on olulised *Project* menüüs paiknevad *Make* ja *Rebuild*.

Make on otstarbekas kasutada terve tööruumi, projekti, paketi või failide kompileerimiseks. Hierarhiliselt kompileerib ta alati seest poolt välja poole ehk madalamal puus asuvad paketid enne kõrgemal puus asuvaid pakette. *Make* on kasutusel rohkem siis, kui tegu on üksikute failidega, milles on muudatusi tehtud. *Rebuild* on aeglasem kompileerimisvahend kui *Make*. Teda läheb vaja siis, kui on vaja kindlust, et kõik projekti või tööruumi kuuluvad failid oleks uuendatud ja kompileeritud. Sõltuvalt sellest, milliseid projekti osi ja faile muudetakse, tuleb valida kahe kompilaatori vahel. Suuremate projektide puhul on tavaline, et paar muutust mingis pealtnäha mitte olulises failis põhjustab muutuse ka mõnes teises temast sõltuvas failis. Ning kui juhtub, et genereeritud kood on segi paisatud käsitsi kirjutatud programmiosadega, võib juhtuda, et keskkond ise ei oska *Make* käigus kõiki sõltuvuses instantse uuendada ja kompileerida.

Project menüüst leiab veel *Project Settings* ja *Default Project Settings* käsu, mille kaudu saab projekti seadeid muuta. Tasub tähele panna, et *Default Project Settings* all määratud seaded

kehtivad kõigi sama tööruumi alla loodud projektidele, kui *Projekt Settings* seaded kohanduvad vaid valitud projektile.

Run menüüst on võimalik käivitada projektimooduleid või `main()` meetodit sisaldavaid `.java` faile.

Tools menüüs tasuks ka vaadata *Properties* käsku, mis avab akna, kus on võimalik JDeveloper'i enda keskkonda vastavalt soovile kohandada.

Kuna eelpool on kirjeldatud peamisi ja tihemini vajaminevaid vahendeid, siis usinam kasutaja võib kindlasti palju vajalikku informatsiooni leida juurde *Help* menüüst, mille kaudu on võimalik uurida lokaalset abiraamatut, kui ka suunduda veebi Oracle poolt pakutavasse tehnilisse võrku.

3.2 Rakenduste loomine JDeveloper'i viisardite abil

Kuna üheks peamiseks kasulikkuse teguriks on JDeveloper arenduskeskkonna kõikvõimalikud sisseehitatud vahendid, mis lihtsustavad rakenduste loomise käigus tööd ning on ühtlasi vajalikud kompaktse ja võimalikult vigadevaba koodi saamiseks, siis tuleb kindlasti peatuda viisardite juures, mis on ühest küljest JDeveloper'i arenduskeskkonna mõte. Kindlasti ei ole aga ainuüksi viisardeid kasutades võimalik luua suureulatuslikke rakendusi.

Enamikele nendest saab ligi lihtsalt valides JDeveloper arenduskeskkonnas vajaliku viisardi *File/New* komponentide valikust. Osad aga käivituvad iseenesest mingis arendusetapis ja pakuvad seda, mida momendil kõige otstarbekam teha oleks.

Järgnevalt tulevad tutvustamisele sagedamini kasutatavad ja vajaminevad viisardid ning lühidad seletused, mida nendega teha saab:

Projekti viisard *– Project Wizard*

Projekti viisardit kasutatakse projekti loomiseks, mille käigus antakse projekti failile (`.jpr`) nimi ning teekond kataloogi, kuhu JDeveloper asetab kompileeritud projektifailid. Samuti saab määrata projekti tüübi (näiteks ärikomponendid). Lisaks on võimalik lasta viisardil koostada HTML fail, mille sisuks on informatsioon projekti kohta. Kui projekti tüübiks on antud ärikomponendid, siis peale projekti viisardi lõppu käivitub ärikomponentide projekti viisard.

Paketi viisard *– Package Wizard*

Paketi viisard pakub põhimõtteliselt seda sama funktsionaalsust nagu ärikomponentide projekti viisard. Tema omaduseks on see, et tal puudub andmebaasiga ühenduse loomise osa.

Viisard käivitub, kui valida see süsteemipuust ärikomponentide projekti kaudu.

Andmebaasiühenduse viisard – *Connection Wizard*

See viisard pakub lihtsa ja kiire mooduse luua andmebaasiühendus. Vaja on vaid anda andmebaasiserveri kohta vajalikud andmed nagu serveri nimi, kasutaja nimi ja parool. Korrektsete andmete korral avab viisard ühenduse.

Ärikomponentide projekti viisard – *Business Components Project Wizard*

Viisard avaneb automaatselt, kui projekti viisardis oli projekti tüübiks valitud ärikomponendid. Teisel juhul on teda võimalik käivitada menüüst *Wizards/Business Components*. Ärikomponentide viisardis saab luua ühenduse andmebaasiga, anda paketile sobiv nimi ning soovi korral saab määrata, milliseid tabeleid kasutatakse olemiobjektide loomiseks. Samuti määratakse ka ühendused, mis baseeruvad nendes tabelites olevatel piirangutel. Seejärel saab määrata samuti soovikohaselt vaateobjektid (baseeruvad olemiobjektidel) ning vaatelingid (baseeruvad ühendustel).

Rakenduse mooduli viisard – *Application Module Wizard*

Vajalik vahend rakendusemooduli loomiseks, millel on rakenduses oma kindlad ülesanded. Rakendusemoodul kätkeb endas kogu rakenduse andmemudelit ja juhib ning koordineerib äriloogikat kõigi tema sisse kuuluvate instantside vahel, milleks on ärikomponendid (vaateobjektid jne). Viisardi vahendusel saab anda rakendusemoodulile nime ning määrata, milliseid vaateobjekte ja vaatelinke on tal vaja sisaldada. Samuti võib lubada viisardil genereerida ka Java fail, kui hiljem tekib tahtmist muudatusi programmi koodi abil rakendusemooduli töösse lisada.

Ühenduse viisard – *Association Wizard*

Selle vahendiga saab luua ühendusi. Loodavad ühendused võivad olla vaikimisi loodud olemasolevate tabelites määratud piirangute kaudu, kui ka luua täiesti uusi seoseid tabelite vahel, mida JDeveloper'i vahendusel genereeritakse. Viisardi käigus saab loodav ühendus omale nime, määratakse olemiobjektid, mille vahel ühendus kehtima hakkab ning atribuudid, mis määravad seosed olemiobjektide vahel ühenduse põhjal. Ka saab määrata ühendusele mitmesuguseid omadusi.

Vaatelingi viisard – *View Link Wizard*

Selle viisardi abil saab luua vaatelinke, mis baseeruvad varem loodud ühendustel ja on vajalikud *master-detail* vaadete loomisel. Viisardi käigus tuleb anda vaatelingile nimi, määrata *master* ja *detail* vaateobjekt, valida atribuudid, mis defineerivad seose *master* ja *detail* vaatelingi vaateobjektide vahel. Samuti saab täiendada päringuid SQL lausetega ning muuta vaatelingi omadusi.

Vaateobjekti viisard – *View Object Wizard*

Vaateobjekti viisard võimaldab luua selliseid vaateobjekte nagu vaja on. Nendeks võivad olla mingi olemiobjekti atribuudid, või valik mitme olemiobjekti atribuutidest. Vaateobjektile tuleb anda nimi, valida olemiobjektid ja nende atribuudid, mida tahetakse vaateobjekti paigutada. Sobiva SQL lausega tuleb luua loogika, et vajalikke atribuutide andmeid kätte saada.

BC4J JSP rakenduse viisard – *BC4J JSP Application Wizard*

Seda viisardit saab kasutada komplekti JSP veebilehtede loomiseks pärast seda, kui on loodud ärikomponente sisaldav projekt ja sellest BC4J kliendi andmemudeli definitsioon, mille käigus on paigaldatud XML faili, laiendiga `.cpx`, andmeseoste kohta käiv informatsioon. Viisardi käigus luuakse iga vaateobjekti kohta leht, mis tagab andmetele ligipääsu. Ärikomponentide JSP rakenduse viisard genereerib JSP rakenduse, mis põhineb BC4J ärikomponentide rakendusemoodulil ja temas sisalduvatel vaateobjektidel.

Ärikomponentide EJB paigalduse viisard – *Business Components EJB Deployment Wizard*

Selleks, et seda viisardit kasutada, peab olema loodud rakendusest `.jar` arhiivi fail vastavale paigaldusplatvormile vajalike komponentidega. Viisard juhib läbi ärikomponentide paigalduse profiilide tüüpide, genereerib vajaliku konfiguratsiooni, mille tulemusena klient pääseb ligi BC4J rakendusele. JDeveloper genereerib kõik vajalikud klassid ja kasutajaliidesed, mis tagavad kaug-ligipääsu rakendusele.

Appleti HTML faili viisard – *Applet HTML Fail Wizard*

Viisardi abil saab genereerida Java aplette[10] sisaldavaid HTML veebilehekülgi. Vastavalt etteantud informatsioonile genereeritakse `.html` fail, mis kujutab endast loodud apletile konteinerit. Töötab see põhimõttel, et brauseris avatud `.html` fail tirib apleti alla ja käivitab selle.

Domeeni viisard – *Domain Wizard*

Viisard domeeni loomiseks, mis sisuliselt loob atribuudi ja talle omase defineeritud andmetüübi.

Olemiobjekti viisard – *Entity Object Wizard*

Olemiobjekti viisard aitab luua olemiobjekte. Olemiobjektid võivad baseeruda juba olemasolevatel tabelitel või ka alles loodavatel tabelitel. Olemiobjekti loomisel, tuleb anda talle nimi, määrata atribuudid ning juhul, kui on soovi seda olemit hiljem kohandada, tuleb lubada luua ka Java fail, kuhu JDeveloper loob olemi kirjelduse klassi, olemiobjekti klassi ja

[10] Java aplett - HTML dokumenti manustatud väike Java programm, mis kujutab endast platvormist sõltumatut rakendusmoodulit.

olemi kollektsiooni klassi või mingi kombinatsiooni. Vaikimisi loob JDeveloper Java faili, mis sisaldab vaid olemiobjekti klassi.

HTTP servleti viisard *– HTTP Servlet Wizard*

Kasutatakse aktiivsesse projekti servleti loomiseks vastavalt määratud meetoditele ja parameetritele. Viisard loob projekti sisse paigalduse kirjelduse faili `web.xml`, mida saab käima lasta JDeveloper'i sisse ehitatud veebiserveris.

JClient andmemudeli viisard *– JClient Data Model Wizard (JClient Data Definition Wizard)*

Enne kui saab hakata tööle *JClient* vormide ja muu atribuutikaga, tuleb luua vastav kliendi andmemudeli defintsioon. Selleks luuakse andmemudeli viisardi abiga projekti sisse konfguratsiooni fail `.cpx`, milles hoitakse informatsiooni kliendi andmemudeli kohta.

JClient vormi viisard *– JClient Form Wizard*

JClient vormi viisardi abiga saab kiirelt luua olemasoleva BC4J projekti andmemudelil põhinevaid *master-detail* ja ühel tabelil põhinevaid vorme. Viisardis tuleb määrata, milliseid vaateobjekte kasutatakse *master*-osas ja milliseid *detail*-osas. Samuti saab vastavalt soovile valida nii vormi kui ka apleti vahel.

JClient paneeli viisard *– JClient Panel Wizard*

JClient paneeli viisard loob komplektse paneeli, mida saab lisada rakenduse raami. Samuti genereeritakse kood paneeli osade sidumiseks.

JClient tühja paneeli viisard *– JClient Empty Panel Wizard*

Viisardi ülesandeks on luua tühi *JClient* paneel, millel pole näiteks nuppe ega tekstivälju. Kasutatakse ühe tabeli andmete paneeli loomiseks olemasolevasse raami. Viisard genereerib koodi, mis on vajalik paneelide sidumiseks.

JClient graafiku viisard *– JClient Grapf Wizard*

Nagu nimigi ütleb, on viisardi eesmärk luua graafikuid, mis põhinevad BC4J andmeallikatel.

SOAP serveri ühenduse viisard *– SOAP Server Connection Wizard*

Viisard loob ühenduse SOAP serveriga.

Java veebi käivituse viisard *– Java Web Start Wizard*

Java veebi käivituse viisard on mõeldud Java klientide loomiseks, mis tegelevad kliendi masinates Java rakenduste ja applettide allalaadimise ja käivitamisega. Java Web Start on Java rakenduste haldur, mis kontrollib rakenduse startimisel internetiühenduse korral keskselt serverilt uuenduste ja paranduste olemasolu ning vajaduse korral installeerib nad automaatselt.

4 Java ärikomponentide tutvustus

Selles osas on põhirõhk Java ärikomponentide raamistiku koostisosade tutvustamisel. Oracle Java ärikomponendid põhinevad Java ja XML-i raamistikul, mis võimaldavad täielikku arendustööd, paigaldust ning mitmekihiliste andmebaasirakenduste loomist korduvkasutavate ärikomponentide kaudu. Sellised rakendused koosnevad tavaliselt kliendipoolsest kasutajaliidesest, mis on kirjutatud Java või HTML keeles, ühest või mitmest äriloogikakihi komponentidest, mis võimaldavad mitmekülgset äriloogikat ja ärikomponentide vaatekülgi. Samuti on üheks kindlaks koostisosaks ka andmebaasiserver, mis sisaldab andmeid. JDeveloper'i keskkonnas on hea koheselt BC4J ärikomponentide poolt genereeritud koodi testida sisseehitatud moodulite vahendusel, ilma et peaks selleks vahepeal oma arenduskäigus olevat rakendust mõnda teise serverisse eksportima, mis seda teha suudaks.

Sellise mitmekihilise Java ärikomponentide raamistiku võimalusi kasutav rakendus omab vaateid, äriloogikat ja kohandatavat koodi oma komponentides, mida on võimalik kasutajatel jagada. Neid komponente on lihtne kasutada ja hallata, samuti ka korduvalt kasutada ja muuta, ning seepärast ei tekita probleeme nende samade komponentide paigaldamine muutmata kujul mistahes BC4J toetust omavale platvormile.

Projekteerimisaegne vaade koosneb mitmetest komponentidest projektis:

- *Application Module* – rakenduse moodul
- *Entity Objects* – olemiobjektid
- *Associations* – ühendused
- *View Objects* – vaateobjektid
- *View Links* – vaatelingid

Komponentide omadusi ja äriloogikat saab edukalt luua viisardite ja redaktorite abiga, mille põhjal BC4J genereerib koodi. Teoreetiliselt on seda koodi võimalik ka ise kirjutada, kuid siinkohal peab tegema kaks märkust: esiteks, pole otstarbekas teha käsitsi seda, mida on pandud tegema arenduskeskkond oma sisseehitatud loogika põhjal, ja teiseks, kui kord alustada käsitsi koodi kirjutamist, siis mingil hetkel võib juhtuda, et tekib tõrge sisseehitatud genereeritud loogilise koodi ja muudetud ehk käsitsi kirjutatud koodi vahel ning nad ei tööta koos. Siiski siinkohal peab mainima, et ka käsitsi tuleb üht-teist koodi lisada, eriti kui tegemist on näiteks probleemiga, mille lahendamiseks arenduskeskkond ise vahendeid ei paku. Sel juhul tuleb vastavad meetodid ise Java koodi lisada. Vajalikud klassid ja meetodid leiab interneti lehelt "*Oracle Business Components for Java 9.0.2*" aadressil:

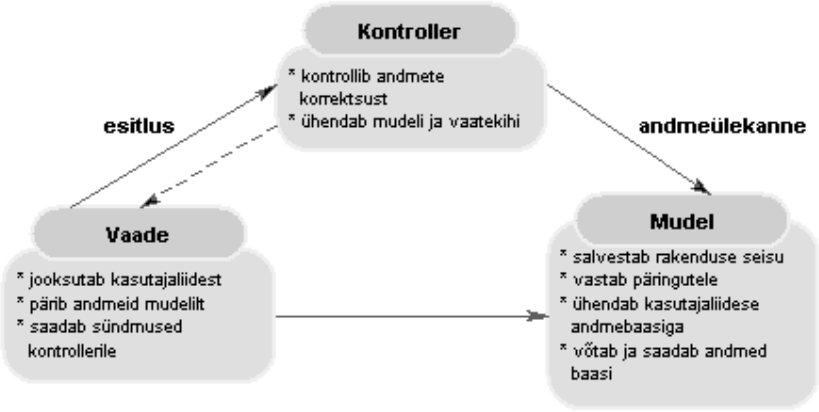
http://www.deakin.edu.au/its/isg/dba/docs/9iasrel2/web.902/q20211/index.html. [21. aprill 2004]

Komponendid on organiseeritud projekti pakettide vahel. Iga komponent koosneb komponendi kirjeldusest XML kujul (*XML Component Definition*) ja ühest või enamast Java klassist, mis sisaldab selle komponendi teostust (*Java Implementation Class*). XML fail sisaldab metaandmeid ehk andmeid rakenduse andmete iseloomu ja seadete kohta, mida arendamise käigus viisardite kasutamisel luuakse. Java failis asub aga objektide kohta kood. Iga objekt on paigutatud paketti, omades kaustakujuliselt organiseeritud ülesehitust.

4.1 *Model-View-Controller* ülesehitus

Model-View-Controller jaotab rakenduse näitlikult kolmeks eraldiseisvaks kihiks, millel on oma ülesanded (vt. Joonis 5). Iga kiht omab kindlat ülesannet. Nimetatakse neid järgmiselt:

- *Model* – mudel
- *View* – vaade
- *Controller* – kontroller



Joonis 5 - Mudel-Vaade-Kontroller skeem

4.1.1 Mudelikiht

Mudelikiht ühendab kasutajaliidese andmebaasi või mõne muu andmehoidla külge ehk on nagu konteiner, mis sisaldab programmi andmeid ja äriloogikat. JDeveloper'is on mudelikiht ja selle loomine Java ärikomponentide ülesanne. Mudelikiht tegeleb peaasjalikult andmete andmebaasist võtmisega ja sinna saatmisega. Enamasti on see koodi pikkuse mõistes kõige

mahukam osa rakendusest, kuid kasutades BC4J komponente, on see tegevus programmeerimise käigus suurel määral lihtsustatud. Mudelikiht on sama nii J2EE kui ka JClient[11] rakendustele, sõltumata sellest, kas nad on loodud kasutades EJB või BC4J komponente. Mudelikiht luuakse ka sõltumata sellest, mis tüüpi rakendusega on tegemist.

4.1.2 Vaatekiht

Vaatekihi ülesandeks on mudeli andmetega toimetamine. Vaatekihid erinevalt mudelikihist JClient ja J2EE rakendustes ei sarnane, kuna vaatekihi ülesandeks on esitada kasutuskõlblik kasutajaliides. Enamasti kasutab JClient Swing'i komponente. Vaatekihi loomiseks kasutatakse JDeveloper'is *drag-and-drop* (tiri-ja-tõmba) meetodit komponentide lisamiseks rakenduse kasutajaliidesele, kuigi keerulisemate rakenduste puhul tuleb ka käsitsi üksjagu koodi kirjutada. Vaatekihi mõte on suhelda kontrollerikihiga ja saata talle lõppkasutaja tegevusele vastavaid viiteid, mille põhjal kontrollerikiht hakkab oma ülesannet täitma.

4.1.3 Kontrollerikiht

Kontrolleri kiht ühendab mudeli ja vaate kihid. Kontrollerikiht tegeleb äriloogikaga, mis kuulub BC4J kihti. BC4J ulatub kahele kihile – mudelikihile ja kontrollerikihile. Iga tegevus, mille lõppkasutaja ette võtab, saadetakse läbi kontrollerikihi, misjärel see kiht vastavalt saadud päringule otsustab, mida edasi teha. Kuidas kontroller töötab, on tema enda otsustada.

4.2 Rakenduse moodul

Rakendusemoodul on olemiobjektide, vaateobjektide, assotsiatsioonide ja linkide konteineriks. Ta on tähtsaim komponent, mis peab kogu muudest komponentidest koosneva paketi tööle panema. Rakendusemooduli ülesandeks on panna käima ühendus andmebaasiga, seda kontrollida, seejärel ükshaaval panna tööle teised ärikomponentide osad, mille tulemusena on võimalik näha kogu projekti senise töö tulemust. Sagedamini kasutatavateks meetoditeks on `createViewObject()`, `findViewObject()`, `createViewObjectFromQueryStmt()`,`findViewLink()`,`remove()`.

[11] JClient – Java Swingi vahenditega loodav JDeveloper'i kasutajaliidese rakendus

4.3 Olemiobjektid

Olemiobjektid on vajalikud äriloogika ja andmebaasis asuvate andmete kokkuviimiseks, mistõttu pole programmeerijatel vaja muret tunda selle pärast, mismoodi andmed baasi jõuavad või sealt välja tulevad. Kasutaja pääseb olemiobjekti andmetele ligi läbi ühe või mitme vaateobjekti, niisama nendele ligi ei pääse. Mingile teatud olemiobjektile võib olla korraga võimalik ligineda mitme vaateobjekti kaudu.

Olemiobjektidel on oma äriloogika, mis sisaldab:

- valideerimise loogikat
- vaikimisi väärtustamise loogikat
- kustutamise loogikat
- kalkuleerimise loogikat

Olemiobjektideks on näiteks "Raamat" ja "Tarnija", millel on omad atribuudid, mis neid olemeid iseloomustavad. Seosed nende olemiobjektide vahel väljenduvad assotsiatsioonides ehk ühendustes. Java ärikomponentide viisardite abil luuakse olemasolevate andmebaasi tabelite põhjal olemiobjektid. Iga andmebaasi tabel ongi tegelikult olemiobjekt, mille iga veerg annab olemiobjekti atribuudid. Nendel atribuutidel on oma loogika, omades primaar- ja/või võõrvõtmeid, täpsuskriteeriumeid või tähemärkide arvu.

Olemiobjekte on ka ise võimalik luua, mispuhul pole vajalik andmebaasis mingi tabeli olemasolu. Viisardi töö käigus luuakse olemiobjektist tabel ja atribuutidest tabeli veerud.

4.4 Assotsiatsioonid ehk ühendused

Assotsiatsioon ehk ühendus loob seose kahe olemiobjekti vahel. Seosed saavad olla üks-ühele või üks-mitmele või mitu-mitmele. Kindlate võtmete kaudu saab üks olemiobjekt ligi teise olemiobjekti andmetele.

Assotsiatsioonid on kahesuunalised, nende kaudu saab info liikuda nii *master-detail*[12] kui ka *detail-master* suunas.

4.5 Vaateobjektid

Vaateobjektid on ärikomponendid, mis kasutavad SQL päringuid, et saada andmebaasist kätte vajalikke vajadusel filtreeritud kujul atribuute kuvatavaks informatsiooniks. Kasutajale annab

[12] *master-detail* – vormiasetuse tüüp, kus *master*-osa on tavaliselt allikaks, mille alusel *detail*-ossa kuvatakse päringu tulemusena kirjed

see võimaluse nende andmetega edasi töötada, neid muuta, lisada, kustutada või lihtsalt vaadelda ja sorteerida. Vaateobjektide vahelised seosed on organiseeritud vaatelinkide kaudu.
Vaateobjektide kaudu on võimalik näiteks osa atribuute jätta kasutajale nähtamatuks, kuigi päring samaaegselt edastab kogu tabeli informatsiooni. Sageli vajalik näiteks turvalisuse mõttes kaitsta veerge või ridu, milles on andmeid, mida ei soovi kasutajale välja näidata. Samuti saab teha päringuid mitmetest tabelitest nii, et kasutaja saab aru nagu tuleks andmed ainult ühest tabelist. Lisaks saab vaateobjekte ka tõhusalt ära kasutada selleks, et muuta veergude nimesid, ilma et see andmebaasitabeli enda veergude nimedele muudatusi teeks ja samuti konkreetselt mingi ühe vajaliku vaate piires mingit ajutiselt loodud atribuuti kalkuleerida.
Võimalik on luua mitu vaateobjekti ühele olemiobjektile ning vaateobjekt saab selekteerida andmed SQL päringuga mitmetest olemiobjektidest.

4.6 Vaatelingid

Vaatelinke kasutatakse kahe vaateobjekti vahel seose loomiseks. Erinevus olemiobjektidel baseeruvatest assotsiatsioonidest on see, et nad on ühesuunalised, suunaga *master-detail* mitte vastupidi. Vaatelink määrab ära allika (*source*) vaateobjekti ja sihi (*destination*) vaateobjekti. Samuti nagu teiste komponentide puhul saab ka siin lisada ja luua vaatelinke viisardit kasutades. Siinjuures teeb viisard vaatelingi, mida edaspidi saavad hakata kasutama rakenduse moodulid.
Kokkuvõtteks:

- **olemiobjektid** moodustavad andmebaasis olevad tabelid
- **assotsiatsioonid** seovad omavahel need olemiobjektid ehk andmebaasitabelid
- **vaateobjektid** edastavad kasutajale nende tabelite sisu
- **vaatelingid** seovad omavahel kaks vaateobjekti

4.7 Valideerimine ja domeenid

Valideerimine toimub läbi Java ubade kasutamise, mis koosnevad olemasolevast koodist, mida saab korduvalt kasutada olemiobjektide atribuutide valideerimisel.
Domeenid on kasutaja enda poolt loodud andmetüübid. Viisardite kaudu on mugav olemiobjektide ja vaateobjektide atribuutidele valiidsusloogikat seada. Valiidsusloogika rakendamiseks on olemas `validate()` meetod.

4.8 Paigaldamine

Kuna iga valmis ehitatud rakendus tuleb ka kasutajale kättesaadavaks teha, peab ta lõpuks saama kuhugi paigaldatud, näiteks mingile rakenduseserverile. Paigaldus tähendab rakenduse ettevalmistust paigaldamiseks mingile kindlale platvormile, kus lõppkasutaja sellega tööle hakkab. Paigaldada võib seda mitmel meetmel, millest tuleb allpool juttu. Oluline on, et kasutajaliidese kood oleks piisavalt kompaktne ja viidud vajaduste tasemele mõttega, et kogu kliendi kasutajaliidese rakenduse vajadusi oleks arvestatud juba rakendusemooduli loomisel. See tähendab, et lisaks vaateobjektidele, mida rakendus kasutab, tuleks luua sinna sisse ka vaatelingid, mis oleks ise valmis koordineerima *master-detail* tööd vastavalt kasutaja nõudmistele. Taoliselt loodud rakendus saab kiiresti pöörduda rakendusemooduli poole ja saab sealt vajalikud vaateobjektid ning hoiab kokku sellega käivitumise aega ja võrgu koormust. Et kogu süsteem korralikult toimiks on vaja silmas pidada paari asja:

- lõpliku rakenduse server peab aru saama BC4J Java ärikomponentidest (OC4J ja Oracle Application Server või WebLogic)
- kindel paigaldusprofiil, mis tähendab sisuliselt mingit kindlat laadi kliendi loomist, kasutades selleks JDeveloper'i BC4J vahendeid (BC4J JSP, BC4J UIX JSP, BC4J Struts JSP)
- paigaldusprofiili ülesseadmine.

Iga paigaldusmeetodi jaoks on vaja luua paigaldusprofiil. BC4J paigaldusprofiili faili laiendiks on `.bcdeploy`.

4.8.1 Lokaalne paigaldus

Lokaalse paigalduse all on kaks olulisemat moodust: BC4J Java ärikomponentide arhiivina paigaldamine ja JClient rakenduse arhiivina paigaldamine.

Lokaalse paigalduse tarbeks luuakse lihtne BC4J Java ärikomponentide arhiiv (`.jar`) ning see paigaldatakse projekti kataloogi. Seda teeb JDeveloper ise peale seda, kui on selleks antud vajalik käsk '*Deploy*' `<projekti_nimi>.bcdeploy` profiilil. JDeveloper loob arhiivi ja paigutab selle väljundi kataloogi. Tavaliselt asub see failipuus järgmiselt:

`<jdev_install>/jdev/mywork/Workspace/Project/classes/<bc4jarhiiv>.jar`

Samuti saab `<projekti_nimi>.bcdeploy` omale kaks alamprofiili, millest üks sisaldab faile, mida Oracle Application Server hakkab jagama kliendiga ja teine, mis sisaldab Java klasse ja XML faile.

JClient paigaldus tähendab sisuliselt tema sidumist BC4J Java ärikomponentide paigalduse arhiiviga. Eelnevalt luuakse JDeveloper'is vajalik JClient rakendus ning sellest omakorda nagu eespool kirjeldatud arhiiv. Vastavalt vajadusele tuleb üle vaadata ka loodud arhiivi paigaldusprofiili seaded.

4.8.2 Kaugpaigaldus

BC4J Java ärikomponentidel põhinevaid J2EE rakendusi saab paigaldada, mistahes J2EE ühilduvatele rakenduse serveritele. Nendeks võivad olla näiteks JBoss, WebLogic või WebSphere. Kõige lihtsam variant on kasutada paigalduskohaks Oracle9*i* Application Serveri "OC4J" konteinerit, milleks vajalikud toimingud viib JDeveloper9*i* integreeritud arenduskeskkond oma vahenditega ise läbi.

4.8.3 EJB-paigaldus

BC4J Java ärikomponentidega on võimalik luua kahte erinevat sorti EJB sessioonioa tüüpi, milleks on tavaline EJB sessiooniuba ehk teenuse sessiooniuba ning eelmisest võimsam "*AppModule*" sessiooniuba [6].

Tavaline sessiooniuba kasutab BC4J raamistiku objekte oma enda privaatrakenduses ning jätab need varju eemalasuva kliendi eest. Teda kasutatakse siis, kui mingil põhjusel peab rakenduse andma edasi kolmandale osapoolele programmeerijatele, kes selle kasutajaliidese kallal edasi töötama peavad. Võimsam ja suurem nn. "*AppModule*" sessiooniuba on kasutatav vastupidiselt tavalisele, võimaldamaks BC4J objektidele ligipääsu eemalasuvatele klientidele. See moodus on mõttekas kasutada juhul, kui on teada, et kliendi kasutajaliides on lõpuni viidud (JSP, Servlet, Swing) sama tiimi poolt ning ei liigu erinevate osapoolte vahel.

Võimalik on kasutada EJB paiglduse puhul klassikalist BC4J-l põhinevat Service Session Bean meetodit, kui ka seda käsitsi teha, kasutamata BC4J vahendeid.

5 Praktikum I: BC4J viisarditel põhineva rakenduse loomine

Peatükk "Praktikum I" on ettenähtud juhiste järgi BC4J Java ärikomponentide rakenduse loomiseks. Töö käigus valmib väike neljal andmebaasi tabelil põhinev infosüsteem. Tabeliteks on `Raamat`, `Tarnija`, `Ost` ja `Ostusisu`.

Eesmärgiks on sammhaaval läbida lihtne mitmetel viisarditel põhinev õpetus, mis on mõeldud algtaseme omandamiseks andmebaasirakenduse loomisel BC4J Java ärikomponente kasutades.

5.1 Andmebaasi kasutaja loomine

Selleks, et pääseda ligi Oracle andmebaasiserveris olevatele tabelitele ja nendega üldse midagi ette võtta, peavad olema kasutajatel vastavad õigused, või kuuluma vastavad tabelid sellele kasutajale.

Kui JDeveloper'i kasutajal pole Oracle andmebaasiserverile võimaldatud ligipääsu, tuleb luua uus kasutaja ning anda talle parool ja kasutajaõigused. Selleks tuleb pöörduda andmebaasiserveri haldaja poole või kellegi poole, kellel on õigus kasutajaid lisada.

Järgnev õpetus kirjeldab, kuidas administraatori kasutajaõigusi omav kasutaja saab luua baasi uue tavakasutaja. Seda on vajalik kasutada siis, kui baasis pole loodud veel kasutajat. Kui on juba kasutajanimi olemas, saab jätkata alapunktist 5.2 Tabelite loomine Oracle andmebaasi.

5.1.1 Kasutaja lisamine Oracle andmebaasi

Otstarbekas on luua uus kasutaja põhimõttel, et iga uus projekt, mida tehakse, oleks baasis kergelt nime järgi ära tunda. Selles õpetuses on kasutajanimeks `bc4j_pood` ja parooliks `bc4j`.

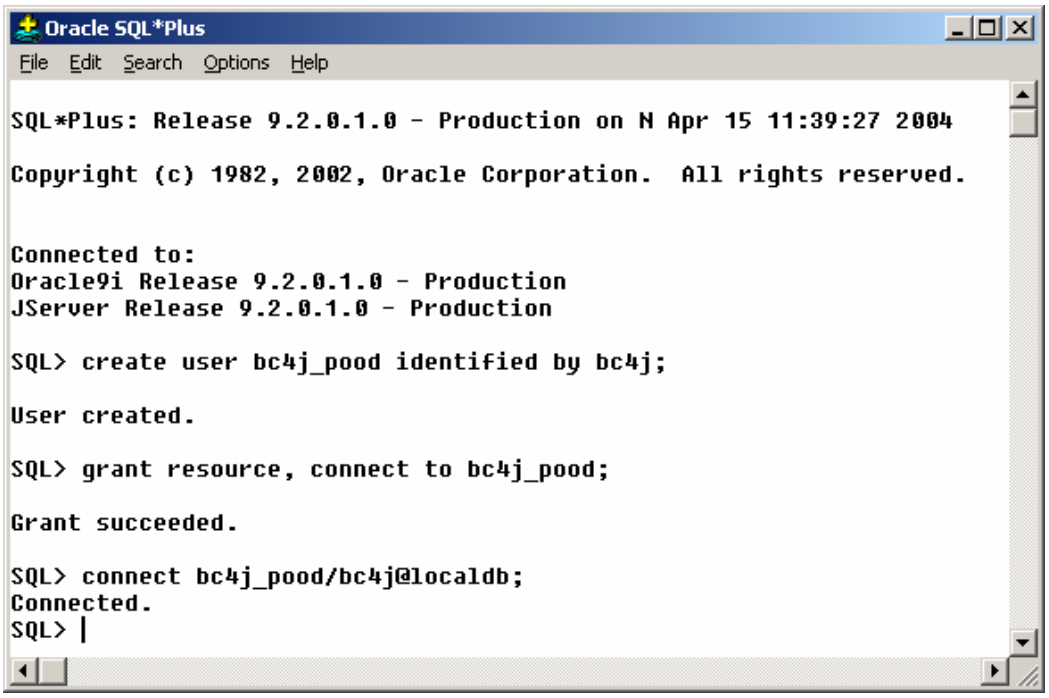
Kasutaja loomiseks saab edukalt kasutada ***SQL Plus*** programmi (vt. Joonis 6).

1. Ava Oracle programmide hulgast *SQL Plus* programm. Logi sisse administraatori õiguseid omava kasutajanime ja parooliga.
2. Loo uus kasutaja, kirjutades SQL viiba järele järgmise lause (vt. Joonis 6):
   ```
   create user bc4j_pood identified by bc4j;
   ```

Kui kasutaja lisamine õnnestus, teatab *SQL Plus* sellest lausega '`User Created`'. Selle lause abil on loodud kasutaja ning andmebaas identifitseerib teda antud parooli järgi. Loodud kasutajale tuleb anda ka õigused, et tal oleks võimalik teostada lihtsamaidki toiminguid.

3. Lisa uus rida lausega (vt. Joonis 6):
   ```
   grant resource, connect to bc4j_pood;
   ```

Toimingu õnnestumise korral, annab *SQL Plus* vastuseks '`Grant succeeded`'. Nüüd omab kasutaja kahte õigust: *resource* ja *connect*. *Resource* ilma lisaparameetriteta annab kõige tavalisemad ressursid, näiteks ühenduse aja või protsessori sessiooni kasutuse aja, millega piiratakse kasutajat, et ta ei saaks üleliia andmebaasis võimust võtta. *Connect* annab minimaalse, kuid piisava õiguse oma kasutaja piires andmebaasi kasutada, sinna tabeleid luua, muuta, kustutada või nendega muid toiminguid teha.

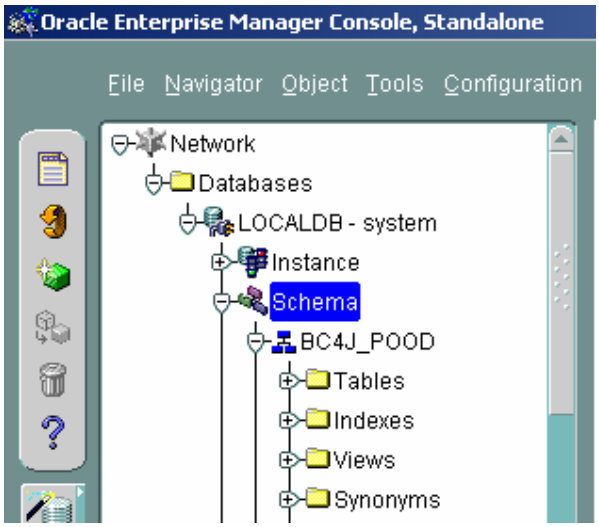


Joonis 6 - SQL Plus programm

Viimaks tuleb loodud kasutajaga andmebaasi sisse logida.

4. Anna järgmine korraldus, kus `<andmebaasi_nimi>` asemel kirjuta oma andmebaasi-nimi (vt. Joonis 6):
   ```
   connect bc4j_pood/bc4j@<andmebaasi_nimi>;
   ```

Nüüd on loodud Oracle andmebaasi kasutajanimeline skeemi objekt (*Schema object*) (vt. Joonis 7). Kasutajale antud talle vajalikud õigused ja ühendatud ta andmebaasi.



Joonis 7 - Skeem Oracle halduskonsoolis

Järgmiseks ülesandeks on tabelite paigaldamine skeemi loodud vastava kasutaja alla Oracle andmebaasi.

5.2 Tabelite loomine Oracle andmebaasi

Tabelite loomise puhul skeemi alla on oluline, et oleks kindlates piirides teada, kuidas infosüsteem, mida ehitatakse, peaks loogiliselt välja nägema. Siin tuleb paika panna täpsed kriteeriumid olemite ja tema atribuutide kohta ning vajadusel lisada ka seosed erinevate tabelite vahele. Selle alusel luuakse andmebaasi tabelid ja muud instantsid, mida Oracle oma andmebaasis vajalikuks peab.

Selles õpetuses saab tabeleid lisada eelnevalt loodud skriptide käivitamise abil.

Millest peaks koosnema tabelite loomise skript? Kui vaadata skriptifaili sisu, siis on see tavaline SQL keele skripti fail, mille saab *SQL Plus* programmis käivitada (vt. ka Lisa 1-5). Algab ta lausega `SET ECHO OFF`, mis pärast skripti käivitamist ei too kogu faili sisu ekraanile. Suurte failide puhul on see kasulik, kuna hiljem on raske ekraanil infot edasi-tagasi kerides vajalikku järge üles leida. Lõpus on viisakas see käsk taas sisselülitada `SET ECHO ON`.

Järjekord, mille alusel lihtne skript koostada, võiks olla järgmine: tabelite kustutamine (*drop table*), järjendite kustutamine (*drop sequence*), atribuutide tüüpide kustutamine (*drop type*),

järjendite loomine (*create sequence*), atribuutidele tüüpide loomine (*create type*) ja tabelite loomine (*create table*) [6].

Oluline on anda tabelitele, atribuutidele, võtmetele ja muudele komponentidele arusaadavad nimed. Erinevate tabelite primaar- ja võõrvõtmed, mis on nimetatud näiteks `PK`, `PK1`, ..., või `FK`, `FK1`, ..., on edaspidi väga raske olemiobjekidega seostada.

Oracle SQL keeles erinevad andmetüübid veidi tavapärasemast SQL keelest. Näiteks Oracle SQL kasutab stringi tüübina `VARCHAR2`, millele võib anda sulgudesse ka pikkuse, näiteks `VARCHAR2(30)`. Arvulise atribuudi tüübina, millega saab tehteid teha, saab kasutada `NUMBER` andmetüüpi. Numbri pikkust tähemärkides ning kohti peale koma saab märkida stiilis `NUMBER(8,2)`.

Rida `create sequence ostu_kood_seq start with 100001;` loob järjendi `Ost` tabelisse atribuudile "kood", mis tähendab, et kood algab numbriga 100001 ja suureneb ühe võrra iga järgmise andmesisestuse korral (100002, 100003, …).

Hiljem andmeid tabelisse lisades saab atribuut "kood" omale väärtuse järgmisel kujul (`tarnija_regkood` on võõrvõti, mis seob omavehel tarnija ja raamatud tarnija registrikoodi kaudu):

```
insert into raamat (kood, pealkiri, autor_enimi, autor_pnimi, i_aasta,
tyyp, keel, laoseis, hind, tarnija_regkood)
values(raamatu_kood_seq.nextval, 'Laste loomaaed', 'Steven', 'Rource',
'2001', 'KK', 'eesti', 56, '125.00', '3456920984');
```

Vahel on vajalik luua ise atribuutidele tüüpe. Näiteks aadressile on selle tegevuse abil üsna mõttekas oma tüüp luua. Selleks luuakse uus tüüp kui objekt ja antakse talle selle objekti sees vajalikud atribuudid nagu on näha järgnevast koodilõigust:

```
create type t_Aadress as object(
  tn_maja varchar2(30),
  linn varchar2(30),
  indeks varchar2(5)
);
```

Ning andmete sisestusel näeb atribuudi tüüp välja järgmine:

```
insert into tarnija (regkood, nimi, aadress, telefon, email)
values ('3593489238', 'Kapsaraud', t_Aadress('Vase 6',
'Tallinn', '12456'), '56564556', 'kapsaraud@raud.ee');
```

Kui skript on valmis kirjutatud (vt. ka Lisa1), tuleb ta *SQL Plus* programmis käivitada.

1. Lisa SQL viiba järele järgmine rida:

```
SQL> @"C:\JDeveloper\jdev\mywork\tabelid_loomine.sql"
```

Kasutaja võib vabalt ise valida, kus tema skriptifail asub, oluline aga on, et tee failini poleks liiga pikk, kuna *SQL Plus* ei suuda seda välja lugeda (üle 64 tähemärgi muutub kohati keeruliseks).

Skriptifaili esmakordse käivitamise järel tuleb ekraanile alguses rida veateateid. Need tähendavad seda, et skripti alguses on antud käsk tabelite, järjendite ja tüübi kustutamiseks, kuid neid andmebaasis veel ei eksisteeri. Õnnestunud toimingu puhul tuleb iga tehtud komponendi kohta vastav teade. Seejärel on tabelid Oracle andmebaasi loodud ning neid on võimalik vaadelda, kasutades Oracle andmebaasi haldusvahendeid.

Nüüd tuleks lisada andmebaasi tabelitesse ka mõned andmed:

2. Käivita *SQL Plus* programmis järgmised failid:

   `tarnija_andmed.sql` (*vt. Lisa 2*)

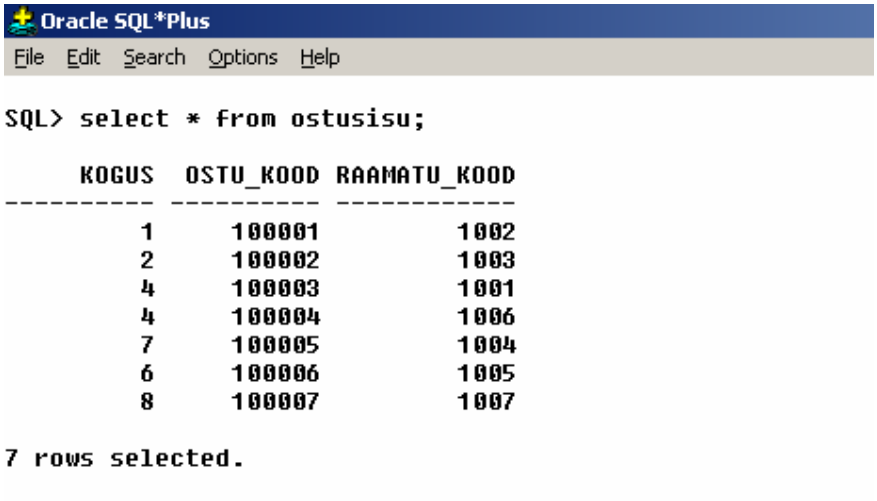
   `raamat_andmed.sql` (*vt. Lisa 3*)

   `ost_andmed.sql` (*vt. Lisa 4*)

   `ostusisu_andmed.sql` (*vt. Lisa 5*)

Olles need käivitanud ja tööle saanud, on sellega vajalikud objektid skeemi alla Oracle andmebaasi loodud.

3. Kontrolli tabelite ja sisu olemasolu lausega `SELECT * FROM <user_table>;` (vt. Joonis 8).



Joonis 8 - päring SQL Plus programmis

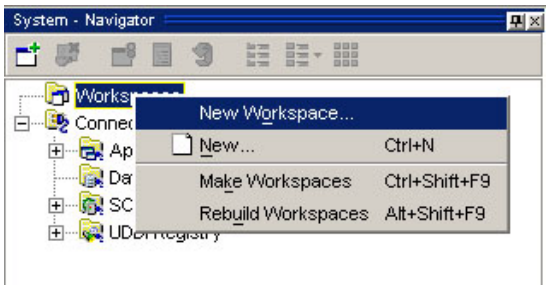
5.3 Arenduskeskkonda tööruumi loomine

Nüüdseks kui kõik vajalikud eeltööd on tehtud, saab asuda andmebaasirakendust programmeerima. Selleks, et alustada uue tööga, on vajalik luua tööruum. Tööruum kujutab endast tekstifaili laiendiga `.jws`, mis sisaldab projektide kohta käivat informatsiooni. Fail asub enamasti kõvakettal koos teiste antud tööruumi projektidesse kuuluvate failidega.

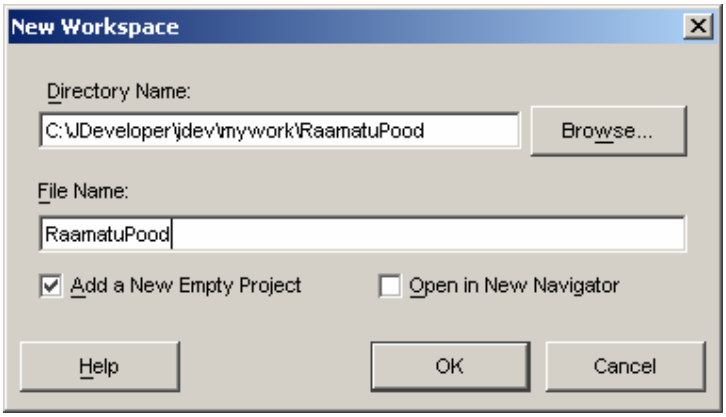
1. Ava JDeveloper.

Tavapäraste seadistuste juures peaks olema vasakul pool avatud *System-Navigator* tööaken (edaspidi süsteemipuu). Kui seda mingil põhjusel ei ole, saab selle avada *View* menüüst valides '*System Navigator*'. Esimest korda JDeveloperi käivitades pole sinna veel loodud ühtegi tööruumi.

2. Loo uus tööruum, avades parema hiireklõpsuga süsteemipuus `Workspaces` (edaspidi tööruum) kataloogi alammenüü ning vali sealt '*New Workspace*' (vt. Joonis 9).



Joonis 9 – Valik System Navigatori all uue tööruumi loomiseks



Joonis 10 - Tööruumi loomise dialoogiaken

3. Tööruumi dialoogiaknas anna kataloogile nimeks “RaamatuPood”. Jälgi, et tee failini ei muutuks, kuna JDeveloper loob vaikimisi kõik tööd *mywork* kausta sisse. Pane failile nimeks “RaamatuPood” (vt. Joonis 10). Märgi ruut *Add a New Empty Project*, mis lisab ka projektifaili tööruumi. Kinnita ‘*OK*’.

Kuna tööruumi käigus sai lubatud luua ka koheselt uus tühi projekt, siis avaneb dialoogiaken *New Project*.

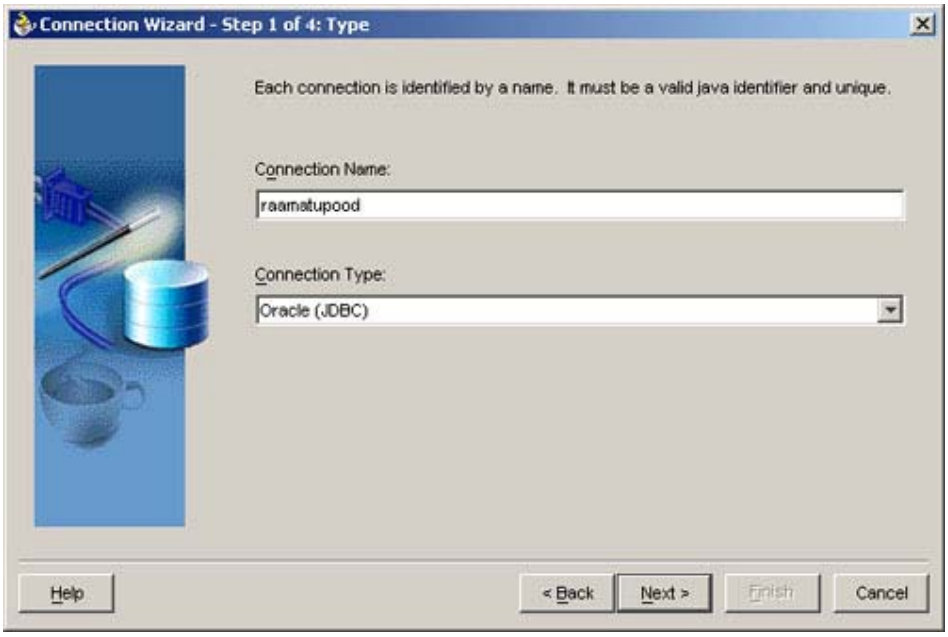
4. Anna projekti kataloogile ja failile nimeks “BC4J_projekt”. Vajuta ‘*OK*’.
5. Salvesta loodud töökeskkond *File/Save all*.

Praeguseks on loodud uus tööruum ning tühi projekt, mis kuvatakse ka süsteemipuus.
Windows’i kataloogipuusse loob JDeveloper kõik vajaminevad projekti failid *mywork* kausta sisse, kuid soovi korral on võimalus muuta projekti failide asukohta. Soovitav on silmas pidada, et tööruum ja projekt koos oleksid, muidu võivad failid kaduma minna ja ei hoia omavahel sidet.
Tööruumi kataloogile ja failile ning tema sees paiknevale projektile ja projektifailile ei ole kohustulik anda ühesuguseid nimesid. See on pigem kasutaja enda otsustada, kuidas talle mugavam on. Samuti ei pea ka muud loodud komponendid kandma samasid nimetusi. Oluline ja esteetiline on nimetada programmi osi nii, et neid segamini ei ajaks ning peale vaadates enam-vähem aru saaks, millega tegu on.

5.4 Ühenduse loomine andmebaasiga

Varasemalt sai loodud Oracle andmebaasi vajaminevad tabelid. Et loodavad rakendused nendes olevat informatsiooni kasutada saaks, on vaja JDeveloper’i arenduskeskkonna ja andmebaasi vahele luua ühendus.

1. Liigu süsteemipuusse ning kataloogi `Connections` peal vali parema hiireklõpsuga alammenüüst ‘*New Database Connection*’. Andmebaasiga ühenduse loomine toimub viisardi vahendusel.
2. Anna ühendusele nimeks “raamatupood” ning jäta ühenduse tüübiks Oracle(JDBC) (vt. Joonis 11).



Joonis 11 - Ühenduse tüübi määramine

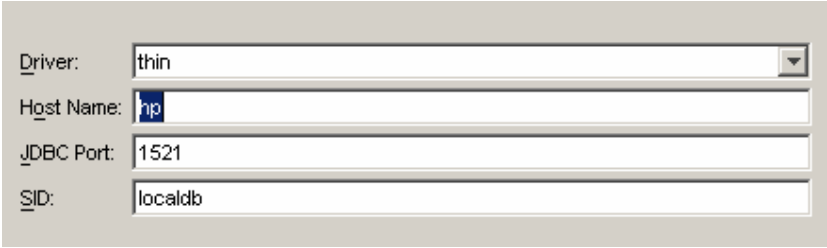
Joonis 12- Ühenduse autentimine

3. Järgmisel lehel kirjuta enda Oracle andmebaasi kasutajanimi ja parool. Selles õpetuses on kasutajanimeks "`bc4j_pood`" ja parooliks "`bc4j`". *Role* rida jäta tühjaks. Märgi ära ruut *Deploy Password* (vt. Joonis 12). Vajuta '*Next*'.

Deploy password on kasulik ära märkida, kuna JDeveloper loob selle tarbeks `.xml` faili, kus asub kasutaja parooli kirjeldus. Enamasti kasutatakse seda ajutiselt infosüsteemi loomise

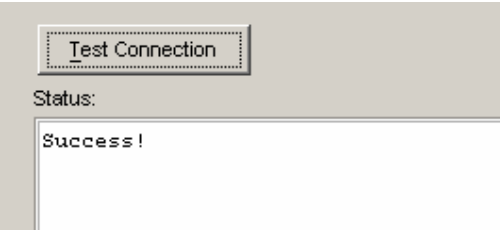
käigus, kui programmil on põhjust näiteks testimise korral andmebaasi poole pöörduda. Sel juhul pole vaja kasutajal iga kord parooli sisestada.

4. Kirjuta *Host Name* väljale andmebaasi serveri nime, milleks võib olla ka servermasina IP aadress. Kirjuta ka servermasina *SID* nimi. *Driver* ja *JDBC Port* jäta puutumata (vt. Joonis 13). Vajuta '*Next*'.



Joonis 13 - Andmebaasiserveri seaded ühenduse loomiseks

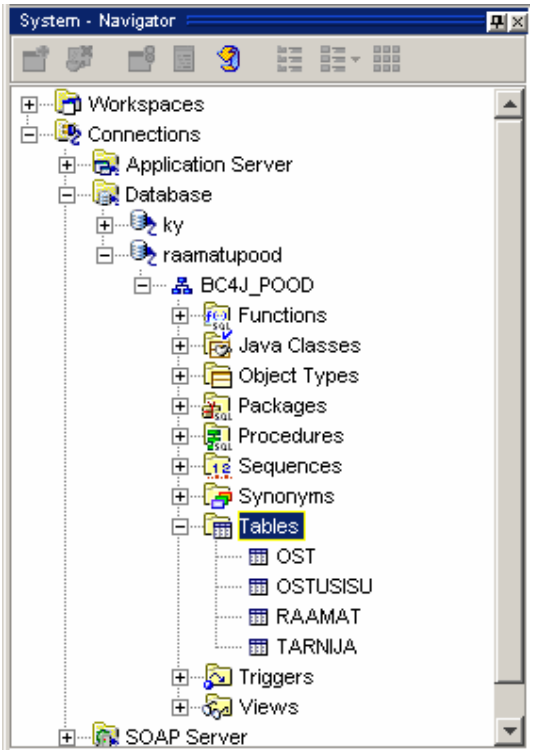
5. Testi ühendust, vajutades nuppu '*Test Connection*' (vt. Joonis 14). Õnnestunud ühenduse puhul annab viisard vastuseks '*Success!*'. Vajuta '*Finish*'.



Joonis 14 - Ühenduse testimine

Kui testimisel ilmnes mingi viga, tuleb üle vaadata viisardi eelmiste lehekülgede seaded ning proovida uuesti.

Andmebaasiga ühenduse loomise lõppedes tekib süsteemipuusse *Connections* kataloogi sisse andmebaasiühenduste alla loodud ühendus "raamatupood". Sisenedes sinna, peavad olema näha kõik neli tabelit, milleks on "raamat", "tarnija", "ostusisu" ja "ost" (vt. Joonis 15).



Joonis 15 - Tabelid loodud ühenduse "raamatupood" all

5.5 BC4J projekti loomine viisardite abil ja testimine

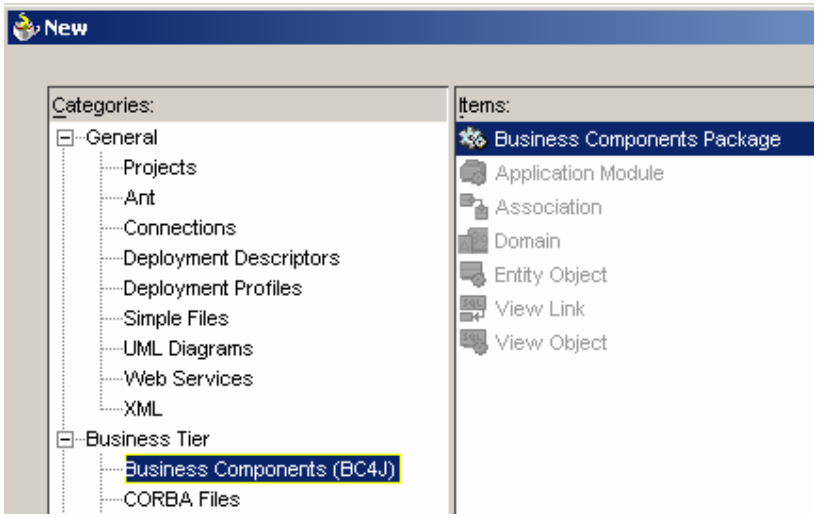
BC4J kihti on vaja selleks, et näiteks Java rakendused või JSP rakendused saaksid sealt kaudu andmeid baasist kätte ja neid esitleda. Samuti on vaja neid andmeid kasutajakihis töödelda ning seejärel tagasi saata. Seega BC4J kiht on nagu mootoriks, mis töötab vastavalt kasutajakihist tulevatele käskudele.

BC4J kihi loomine toimub järgmiselt:

1. Vali menüüst *File/New* või süsteemipuust `BC4J_projekt.jpr` sõlmel parema hiireklõpsuga alammenüüst '*New*'.

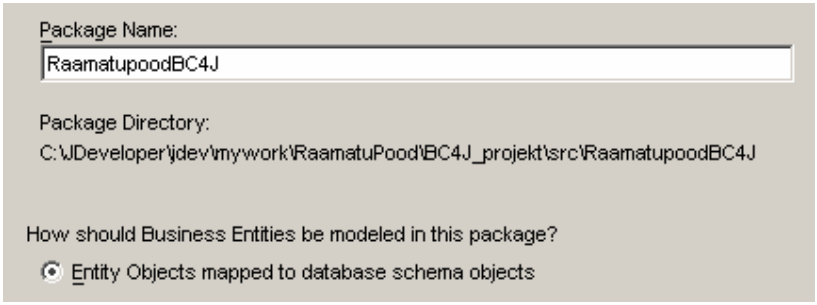
Iga ettetuleva uue komponendi loomiseks saab kasutada ka *File/New* käsku, samuti ka klahvikombinatsiooni CTRL+N, mille järel avaneb uue elemendi loomiseks vajalikke komponente pakkuv dialoogiaken.

2. Avanevast aknast ava vasakust paneelist sõlm *Business Tier*, seejärel vali sõlmest *Business Components (BC4J)* ning paremast paneelist vali komponent *Business Components Package* (vt. Joonis 16). Vajuta '*OK*'.



Joonis 16 - Komponentide valiku põhiaken

3. Järgmisena liigu edasi avanenud ärikomponentide paketi viisardi sissejuhatavalt lehelt ning anna paketile nimeks "RaamatupoodBC4J" ja jäta muud seaded samaks (vt. Joonis 17). Vajuta '*Next*'.

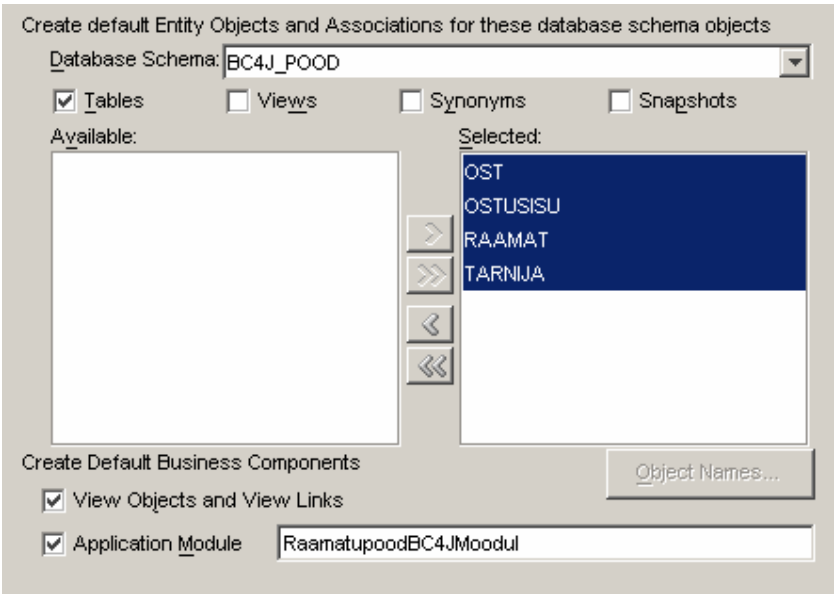


Joonis 17 - Ärikomponentide paketi viisardi paketi dialoog

4. Ühenduse lehel kirjuta ühenduse nimeks *Connection Name* väljal "raamatupood". Ülejäänu jäta nii nagu on ja liigu edasi '*Next*'.

Kui seni on kõik seadistused õigesti läinud, siis on ekraanil avatud leht, kus saab valida oma rakenduse tarbeks vajaminevaid tabeleid. Andmebaasi skeemiks peaks olema juba vaikimisi viisardi poolt valitud "`BC4J_POOD`". Samuti peaks vasakul olema näha kõik tabelid, mis said enne andmebaasi loodud.

5. Vali kõik tabelid, kasutades selleks kahekordse noolega nuppu. Märgi märkeruudud "*View Objects and View Links*" ja "*Application Module*" ning anna moodulile nimeks "RaamatupoodBC4JMoodul" (vt. Joonis 18). Vajuta edasi '*Next*'.



Joonis 18 - Ärikomponentide andmetabelite valiku leht

View Objects and View Links on vaateobjektid ja vaatelingid, mida JDeveloper kohe koos olemiobjektidega genereerib ning mis põhinevad andmebaasitabelitest saadud seoste põhjal. Samuti luuakse ka kohe rakendusemoodul (*Application Module*) vaikimisi seadetega, mis omavahel süsteemiosad tööle paneb ja lubab näiteks testida süsteemi erinevates etappides. Rakendusemoodilit kasutavad ka kõik teised hiljem juurde ehitatavad rakendused (JSP, JClient), sest nad põhinevad sellele moodulile (moodulis kirjeldatud vaadetele).

6. Vaata üle kokkuvõtval lehel, milliseid objekte JDeveloper viisardi lõppedes projekti genereerib. Kui kõik on korras, vajuta '*Finish*'.

Peale mõningast ragistamist tekib süsteemipuusse rida uusi kirjeid. Lähemal vaatamisel näeb süsteemipuus olemiobjekte, nendel põhinevaid vaateobjekte ning olemite vahelisi seosed. Samuti on loodud sinna projekti moodul, mis kogu süsteemi koos tööle paneb. Mooduli Java failis seotakse omavahel kõik instantsid ning `main()` meetodi ülesandeks on kogu süsteemi käivitamine.

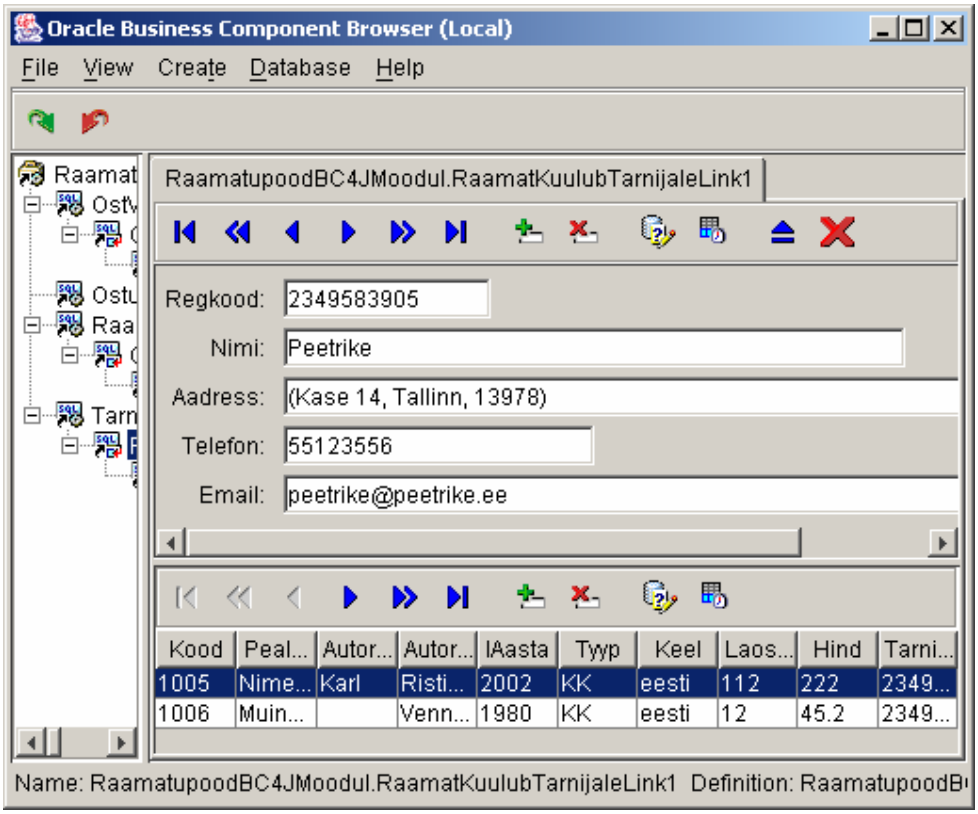
Senise töö tulemusena sai loodud terviklik töötav loogiline süsteem. Et selles veenduda, tuleb süsteemi testida. Selleks on JDeveloper'i sisseehitatud ärikomponentide brauser (*Business Components Browser*).

5.5.1 Testimine

Testimist saab läbi viia järgmiselt:

1. Vali süsteemipuus projekti seest **RaamatupoodBC4JModul** ning sealt parema hiireklõpsuga avades alammenüü, vali ***Test***.
2. Käivitunud ärikomponentide brauseri ühenduse paneelis jäta vaikimisi valikud ja vajutada '*Connect*'.

Sõltuvalt arvuti kiirusest ja programmi suurusest kulub vähe aega enne kui Oracle ärikomponentide brauser käivitub.



Joonis 19 - Sisseehitatud ärikomponentide brauser

Kui brauser on avanenud, on näha vasakul pool paneelis vaateobjekte ja vaatelinke nende vahel (vt. Joonis 19). Kuna viisardis sai lubatud JDeveloper'il genereerida moodul vaikimisi

seadetega, siis on brauserisse kogutud kõik vaateobjektid nende seoste põhjal, mis andmebaasitabelitest on saadud. Testimisel tuleks üritada sisestada, kustutada või muuta andmeid ja jälgida, kuidas seejärel toimivad seosed.

3. Ava `RaamatKuulubTarnijaleLink1`, valides alammenüüst *Show*, või tee topeltklõps lingi nimel.

Paremasse aknasse peaks nüüd avanema *master-detail* leht. Üleval on tarnija andmed ja all tarnijale kuuluvate raamatute andmed. Raamatu andmed on seotud tarnija registrikoodiga.
Klõpsates hiirega sinistel noolenuppudel, saab kirjeid edasi ja tagasi kerida. Muutmisel, lisamisel või kustutamisel saab kasutada kahte noolega nuppu, mis asuvad aknas vasakul pool ülal. Roheline (*Commit*) kinnitab muutuse ja salvestab selle ka baasi. Punane (*Rollback*) seevastu tühistab tegevuse. Tasub tähele panna ka seda, et vaadetel on erinevad numbrid. Vaade "1" on alati sõltumatu vaateobjekt, mis tähendab, et ta näitab kõiki kirjeid, mis andmebaasi tabelis on. Näiteks võib võrrelda omavahel `RaamatView1` ja `RaamatView2`. `Raamatview1` näitab kõiki kirjed, seevastu `RaamatView2` vaid neid, mis kuuluvad konkreetsele tarnijale.
Andmetega opereerimiseks on mõeldud *detail*-osa. Testi *detail*-osa.

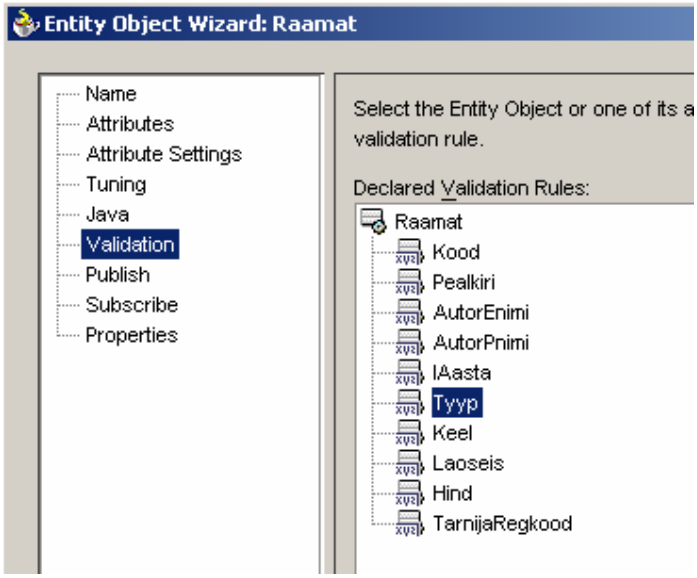
Ülesanded:

- Lisa mõnele tarnijale raamatuid.
- Vaata, kuidas andmed teiste vaatelinkide all muutuvad.
- Mõtle, millist äriloogikat oleks vaja lisada andmete kohta.

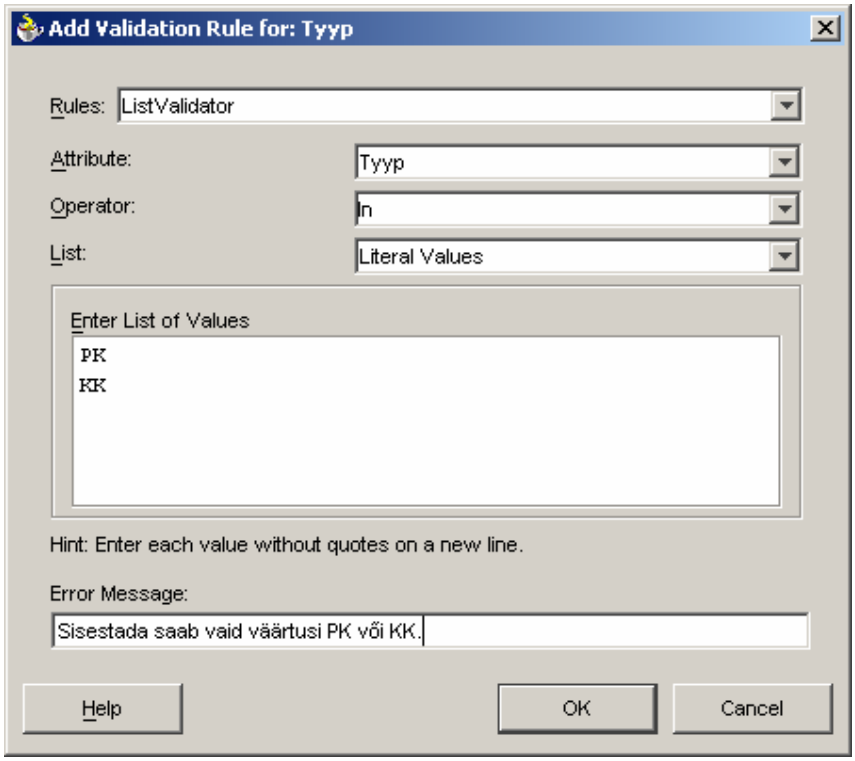
5.5.2 Valideerimine

Valideerimine on vajalik atribuutidele mingite piirangute seadmiseks. Piiranguid võib olla mitmesuguseid. Näiteks olemiobjekti Raamat atribuuti "Tyyp" võiks saada määrata vaid kahte moodi: pehme köide – PK ja kõva köide – KK. Kui sellised piirangud on sisse viidud, siis lõppkasutaja ei saa midagi muud sisestada peale nende kahe lubatud väärtuse.

1. Tee topeltklõps Raamat olemiobjektil või vali parema hiireklõpsuga alammenüüst *Edit Raamat*. Avanevast aknast vali vasakult menüüst *Validation*. Edasi vali paremal pool näha olevate atribuutide hulgast `Tyyp` ja vajuta nuppu '*New*' (vt. Joonis 20).



Joonis 20 - Olemiobjekti viisard



Joonis 21 – "Tyyp" atribuudi valideerimise seaded

2. Avanenud lehel (vt. Joonis 21) vali *Rules* rippmenüüst *ListValidator* ja vaata, et *List* väljal oleks *Literal Values*. *Literal Values* lubab sisestada vaid mittearvulisi väärtusi. *Enter List of Values* tekstivälja kirjuta valiidsuskriteeriumiks üksteise alla "PK" ja "KK". Soovi korral sõnasta ka veateade. Kinnita sisestus '*OK*'.

Raamat olemiobjekti atribuutide nimistusse lisandunud atribuudi “Tyyp” külge tekib valideerimise kirje.
Süsteemipuusse on lisandunud Java fail `RaamatImplMsgBundle.java`, milles on meetod veateate kohta, mida oli võimalik määrata valideerimise käigus. `Raamat.xml` fail on saanud lisaks kirjelduse “Tyyp” atribuudile, kuhu on lisandunud *ListValidationBean* meetod. XML faili sisse võib soovi korral ka ise valideerimise ube kirjutada.

3. Testi uuesti moodulit. Vali vasakult RaamatView1. Katseta Tyyp väljale sisestada mingi KK või PK erinev väärtus. Vajuta ENTER või TAB klahvi veendumiseks, et vale sisestuse korral annab programm veateate ning vale väärtusega kirjet ei aktsepteeri.

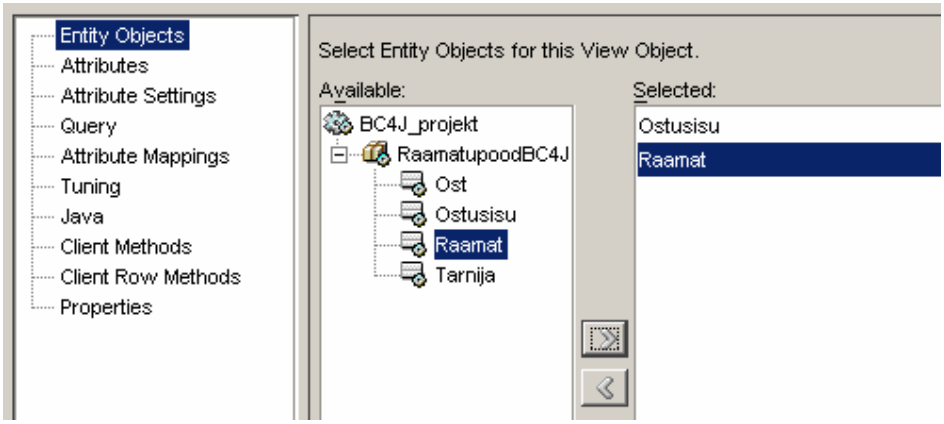
5.5.3 Kalkuleerimine

Sageli on vaja, et süsteem oskaks mingite väljade puhul ühtteist kokku arvutada. Näiteks on vaja, et hind sõltuks kogusest, või erinevate objektide hinnad saaks kokku liita summaks. Selleks on JDeveloperil olemas ka vastav vahend.
Testides moodulit on näha, et OstuKuuluvRaamatuKoodLink1 *detail* paneel võiks anda veidi rohkem informatsiooni. Vaja oleks näidata üksikeseme hinda ja esemete arvust sõltuvat kogusummat.
Selleks tuleb luua uus kalkuleeritud atribuut “SUMMA”. Kalkuleerimine toimub lihtsa avaldise abil, kus hind korrutatakse kogusega:

```
Summa = Ostusisu.Kogus*Raamat.Hind
```

Enne “Raamat” olemihulga atribuudi “Hind” lisamist “Ostusisu” vaatesse, tuleb omavahel siduda kaks olemihulka: “Ostusisu” ja “Raamat”.

1. Vali süsteemipuus parema hiireklõpsuga `OstusisuView` alammenüüst *Edit OstusisuView*.
2. Avanenud aknast vali *Entity Objects* (vt. Joonis 22).



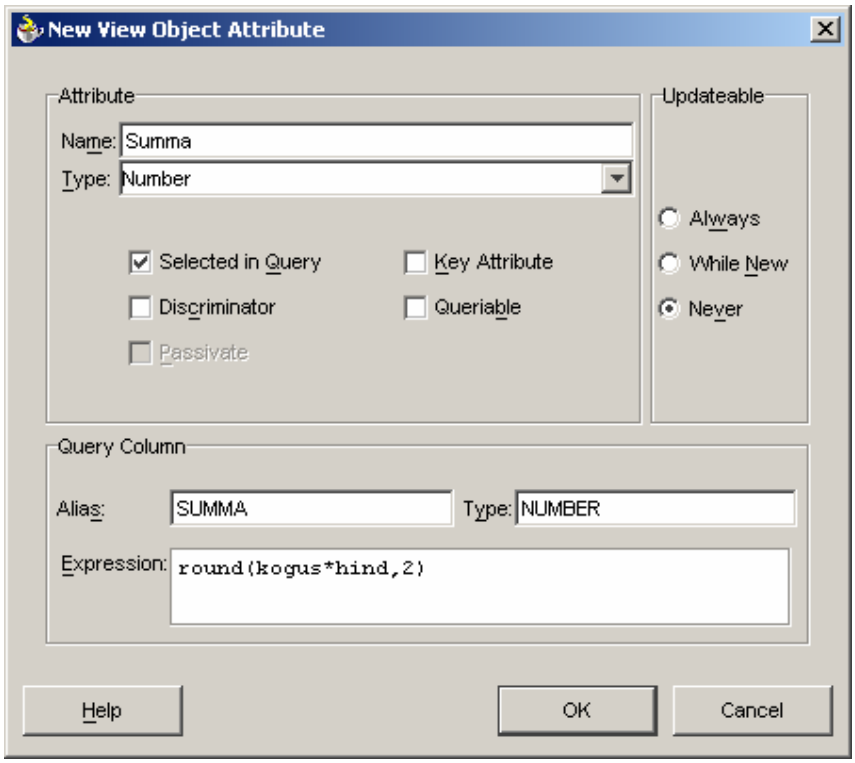
Joonis 22 - Olemiobjekti seaded

Kuna "Hind" on "Raamat" olemiobjekti atribuut, siis tuleb määrata ka seos. Paremal, valitud olemiobjektide aknas, peaks juba olema valitud Ostusisu olemiobjekt. Samuti tuleb sinna lisada ka Raamat.

3. Vali Raamat olemiobjekt ja lisa see noolenupu abil paremasse paneeli. Vajuta '*Apply*'.
4. Vali vasakust menüüst Attributes. Atribuutide lehel vali Raamat olemiobjekti atribuutide hulgast Hind ning lisa see noolenupu abil paremasse paneeli.

"Hind" atribuudiga liigub kaasa ka "Kood". See peabki nii olema kuna "Kood" on "Raamat" tabelis primaarvõtmeks ning "Ostusisu" tabelis võõrvõtmeks. Ostusisu vaade sisaldab nüüd kaks korda "Raamat" tabeli atribuuti "Kood", mis pole aga momendil oluline. Mitte tarvilikke atribuute saab lõppkasutaja eest teha varjatuks.
Järgmisena tuleks luua uus atribuut, milleks on "Summa".

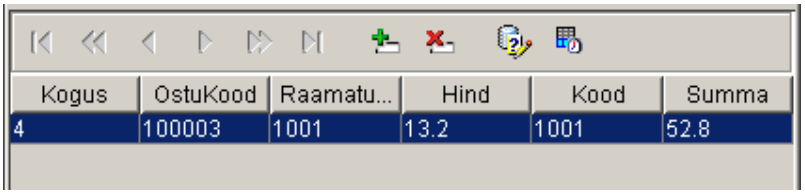
5. Vajuta samas aknas nuppu '*New*'. Avanenud dialoogiaknas *New View Object Attribute*, anna atribuudile nimeks "Summa". Tüübiks vali rippmenüüst "Number" ning märgi märkeruut *Selected in Query*, mis laseb sisestada arvutuslause. *Alias* nimeks pane "SUMMA" ja vaata, et tüüp oleks seal ka "NUMBER". Expression väljale kirjuta järgmine lause (vt. ka Joonis 23):
   ```
   round(kogus*hind,2)
   ```
   Kinnita '*OK*'.



Joonis 23 - Loodava vaateobjekti seaded

Vaateobjekti viisardis on nüüd näha lisatud atribuuti "Summa" talle omistatud avaldisega.

6. Testi uuesti moodulit ja vaata, kuidas muutub summa kui suurendada või vähendada `OstuKuuluvRaamtuKoodView1` detail paneelis koguseid.
7. Muuda koguse algset väärtust ja vajuta '*Commit*' (roheline noolega nupp). Kui koheselt pole tulemust näha, uuenda vaadet minnes *master*-paneelis eelmisele või järgmisele kirjele ja siis tagasi tulles (vt. Joonis 24).



Joonis 24 - Päringutulemus detail paneelis

Ülesanded:

- Loo analoogne arvutus laoseisu tarbeks. Nii palju kui on mingit raamatut ostetud, selle võrra kahaneb laos olevate raamatute arv.

- Valideeri ostu kogust nii, et kasutaja ei saa rohkem raamatuid osta kui lattu järele on jäänud.

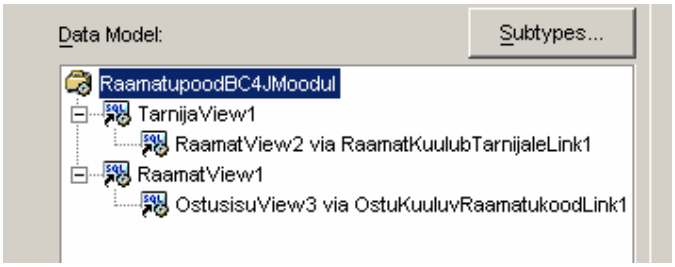
5.5.4 Mooduli kohandamine

Moodulit on vaja tihti enne natuke kohandada, kui sellest kasutajaliides vormida. JDeveloperi keskkonna poolt automaatselt genereeritud moodul võib vahel sisaldada ülearuseid vaateid, mida lõppkasutajal ei pruugi vaja minna. Sellepärast tasub hoolikalt läbi mõelda, mida vaadetesse jätta.

Testimise käigus saab selge ülevaate, millised võiksid olla mooduli vaated.

Master-detail rakendus võiks anda võimaluse vaadelda tarnijatele kuuluvaid raamatuid tarnijate kaupa ja ostetud raamatuid raamatute kaupa.

1. Vali `RaamatupoodBC4JMoodul` ja parema hiireklõpsuga alammenüüst *Edit RaamatupoodBC4JMoodul*.
2. Avanenud viisardis korrigeeri andmemudeli vaadet nii, et sinna jääksid soovitud vaated. Selleks eemalda *Data Model* paneelist OstView1 ja tema alamkataloog, kasutades noolenuppu. Tee sama ka OstusisuView1 vaatega (vt. Joonis 25).



Joonis 25 - Mooduli viisardi andmemudeli seaded

Vajuta '*OK*'. Salvesta kogu projekt. Testi moodulit.

Ülesanded:

- Lisa andmemudelisse vaade, mis näitab millal mingit raamatut on ostetud.
- Kohanda mudelit vastavalt oma äranägemisele.

Kokkuvõte:

On valminud BC4J äriloogikat kasutav kiht, mis on aluseks lõppkasutaja rakenduste loomiseks. Loogiliste operatsioonidena on lisatud kalkuleerimine ja valideerimine. Rakenduse moodul on viidud sellisele tasemele, et ta oleks lihtne ja mõistetav.

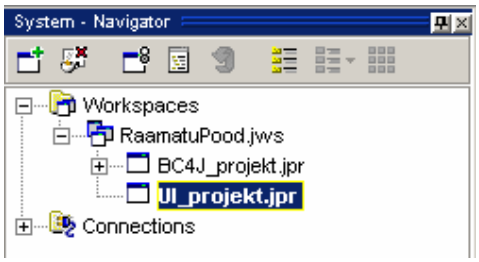
5.6 Java JClient rakenduse loomine

Selles alapeatükis saab võimaluse õppida tundma, kuidas luua valminud mudelile Java komponentidega kasutajaliides. Kuna igal projektil on oma funktsionaalsus, siis tuleb kasutajaliidese kood eraldada ärikomponentide koodist ning luua Java rakenduse tarbeks uus projekt.

5.6.1 Kasutajaliidese projekti ja kliendi andmemudeli loomine

1. Loo uus projekt liikudes süsteemipuus `Raamatupood.jws` tööruumi peale ning vali alammenüüst *New Empty Project*. Projekti loomise dialoogi aknas anna kataloogile ja failile nimeks "UI_projekt".

Süsteemipuu peaks nüüd näitama uut loodud projektifaili `UI_projekt`.jpr (vt. Joonis 26).



Joonis 26 - Tööruum uue projektifailiga UI_projekt.jpr

Lõppkasutaja rakenduse loomisel tuleb eelnevalt määrata, milliseid vaateobjekte ehk tabeleid rakenduses kasutama hakatakse. See informatsioon kirjutatakse kliendi andmemudelisse.
Kliendi andmemudeli loomiseks tuleb teha järgmised toimingud:

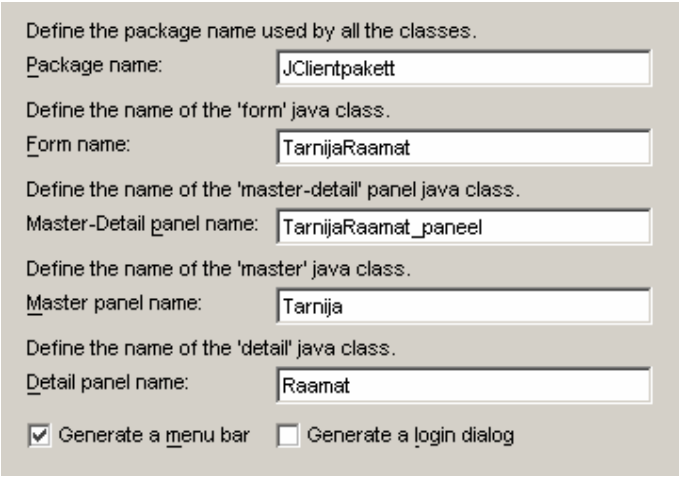
2. Ava "*New*" komponentide aken. Vasakust paneelist ava *Client Tier* sõlm ning vali sealt *Swing/JClient for BC4J*. Paremast paneelist vali *Business Components Client Data Model*.
3. Avanenud andmemudeli viisardi kirjelduse lehel jälgi, et ärikomponentide projektiks oleks valitud "BC4J_projekt.jpr" (ülejäänud väljad on vastavalt projektile vaikimisi õigesti täidetud, kuid pole keeruline seda kontrollida). Vajuta '*Next*'.
4. Jäta andmemudeli kirjelduse nimi "RaamatupoodBC4JMoodul". Vajuta '*Next*' ja '*Finish*'. Salvesta kogu projekt.

Nüüd on süsteemipuusse lisandunud kliendi andmemudeli fail `UI_projekt.cpx`, vajaliku informatsiooniga, millele tuginedes saab klient ühenduse ärikomponentide rakenduse mooduliga.

5.6.2 JClient vormi loomine

Järgmiseks tuleb luua JClient vorm JDeveloperi viisardi abil.

1. Vali parema hiireklõpsuga `UI_projekt.jpr` alammenüüst "*New*". Ava *Client Tier* sõlm ja vali sealt *Swing/JClient for BC4J*. Paremast paneelist vali *Form* ning vajuta '*OK*'.
2. Avanenud vormi viisardist vali vormi tüübiks *Master-Detail Tables* ja märgi ära ka *Form*.
3. Lehel *Form Layout* jäta kõik vaikimisi määratud seaded.
4. Lehel *Data Model* jälgi, et andmemudeli kirjelduse väljale oleks kirjutatud "RaamatupoodBC4JMoodul". Vajuta '*Next*'.
5. Paneeli vaates vali *master*-paneeliks TarnijaView1, millepeale *detail*-paneeli kuvatakse RaamatView2. Vajuta '*Next*'.
6. Atribuutide lehel, kust saab valida atribuute *master*- ja *detail*-paneeli, jäta kõik vaikimisi valitud atribuudid ning vajuta '*Next*'.
7. Lehel *File Names* nimeta klassid nagu näha joonisel 27. Vajuta '*Next*' ja '*Finish*'. Salvesta kogu projekt.



Joonis 27 - JClient rakenduse Java failidele nime andmine

Projekt on täienenud nüüd nelja Java faili võrra. Töölaual on avatud `TarnijaRaamat.java` *UI Editor* ehk visuaalne töölaud, kus saab teha muudatusi vastavalt oma soovile, üritades kasvõi väljade suurusi kohandada. Et saada ülevaade, kuidas loodud liides töötab, tuleb rakendus käivitada.

8. Vali süsteemipuust `TarnijaRaamat.java` ja käivita see valides parema hiireklõpsuga alammenüüst '*Run*'. Liigu mööda kirjeid ja lisa või muuda neid soovi korral.

Ülesanne:

- Loo analoogselt uus *master-detail* leht, kust on võimalik jälgida raamatute müüki.

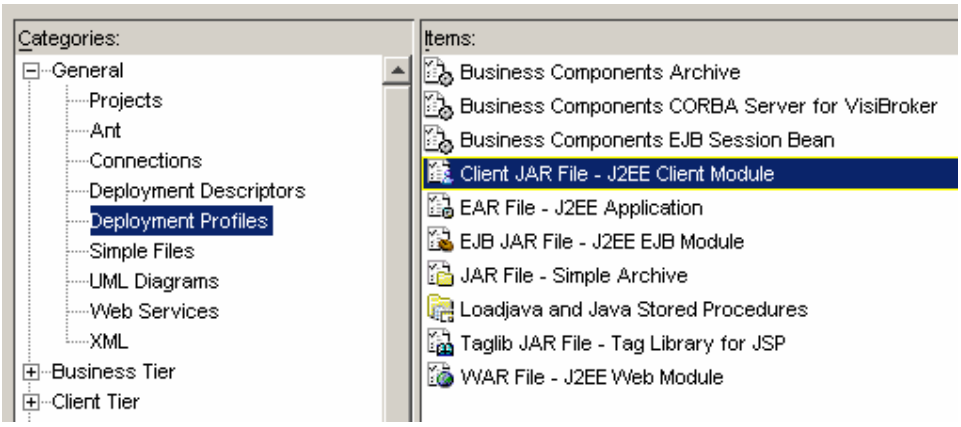
Kokkuvõte:

Selles punktis on selgeks saanud, milliseid etappe tuleb läbida kasutajaliidese loomiseks: uue projekti loomine ja kliendi andmemudeli loomine, milles sisalduv infomatsioon aitab kasutajakihil suhelda andmekihiga. Seejärel Java Swingi komponentidel põhineva *master-detail* rakenduse loomine ja selle käivitamine.

5.7 Java rakenduse paigaldusprofiili ja JAR faili loomine ning käivitamine

Java JClient rakenduse paigaldamiseks on vaja luua JAR fail, mis kõik projekti elemendid kokku pakib ja ette valmistab eemalasuvatesse seadmetesse paigaldamiseks. Esmalt on vaja luua paigaldusprofiil, mis sisaldab paigaldatava paketi kohta kõike vaja minevat informatsiooni.

1. Vali `UI_projekt.jpr` alammenüüst '*New*'. Ava vasakul koponentide kataloog *General* ja vali sealt *Deployment Profiles*. Paremast paneelist vali *Client JAR File – J2EE Client Module* (vt. Joonis 28). Vajuta '*OK*'.



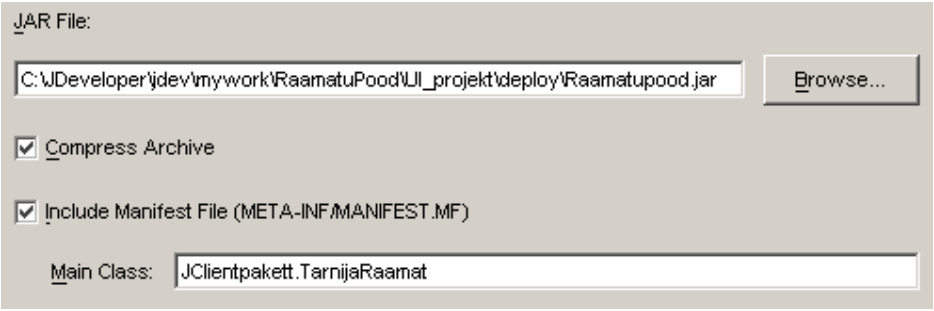
Joonis 28 - Paigalduse profiili valimine komponentide aknast

2. Salvesta paigaldusprofiil nimega "Raamatupood.deploy". Vajuta '*Save*'.
3. Profiili seadete dialoogis vali vasakust paneelist *Dependency Analyzer* ja märgi järgmised märkeruudud:
   - *JClient Runtime*
   - *Oracle XML Parser v2*
   - *JDeveloper Runtime*
   - *BC4J Runtime*
   - *Oracle JDBC*
   - *Connection Manager*
   - *BC4J Oracle Domains*

   Märgi ära ruut *Include All of Their Contents in the Output*.
4. Vasakult menüüst vali *JAR Options* ja tee märge märkeruutu *Compress Archive*. Kontrolli, et JAR faili asukoht oleks järgmine:
   `C:\JDeveloper\jdev\mywork\RaamatuPood\UI_projekt\deploy\Raamatupood.jar`

Loodav JAR fail peab sisaldama manifesti faili, mis on vajalik isekäivituva rakenduse tekitamiseks.

5. Tee märge ruutu *Include Manifest File (META-INF/MANIFEST.MF)* ja kirjuta *Main Class* tekstivälja `JClientpakett.TarnijaRaamat` (vt. Joonis 29). See on main meetodit sisaldav `.class` fail. Talle tuleb ette anda alati ka pakett, et käivitamise käigus see `.class` fail üles leitaks. Vajuta '*OK*' ja salvesta seejärel projekt.



Joonis 29 - JAR arhiivi seaded

6. Vali loodud `Raamatupood.deploy` alammenüüst *Deploy to JAR file*.

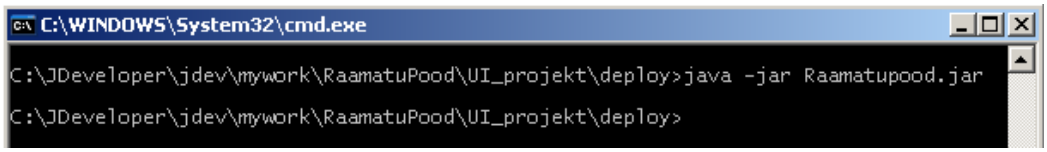
JDeveloper ragistab mõnda aega ning pakib JAR faili kokku. Logi aknasse ilmub informatsioon:

```
Running dependency analysis...
Wrote JAR file to
C:\JDeveloper\jdev\mywork\RaamatuPood\UI_projekt\deploy\Raamatupood.jar
Elapsed time for deployment:  12 seconds
----  Deployment finished.  ----
```

Paigalduse töö lõppedes tuleb loodud `Raamatupood.jar` fail käivitada, et veenduda tema töökorras olekus.

7. Ava commander või muu taoline shell ning liigu kataloogini, kus asub loodud `.jar` fail. Käivitamiseks sisesta viiba järele käsk (vt. ka Joonis 30):

   ```
   java –jar Raamatupood.jar.
   ```



Joonis 30 - Käsklus shellis

Kui kõik seaded on paigas, peaks rakendus käivituma. Kui aga juhtub ilmnema viga, mis viitab sellele, et kompilaator ei suuda leida `main()` meetodit sisaldavat klassi, tuleks üle vaadata kas JAR failis asub kataloog META-INF ning selle sees fail MANIFEST.MF. Selleks paki mõne pakkimisprogrammiga JAR fail lahti ja vaata, mis seal sees on.
MANIFEST.MF fail avaneb *Notepad* programmi abil. Veendu, et see fail sisaldaks analoogset informatsiooni:

```
Manifest-Version: 1.0
Main-Class: JClientpakett.TarnijaRaamat
Created-By: Oracle9i JDeveloper 9.0.3
```

Vajadusel muuda `.deploy` profiili seadeid, klõpsates hiirega kaks korda `Raamatupood.deploy` failil süsteemipuus. Peale muudatuste tegemist tuleb pakkida ka uus JAR fail.

5.8 JSP projekti loomine ja käivitamine

JSP projekt on vajalik luua selleks, et rakendust brauseri kaudu kasutada saaks. Ta paigutatakse kas serverile või kasutatakse lokaalses masinas vastavalt vajadusele.
Selleks, et selline asi tööle saada, tuleb luua uus projekt.

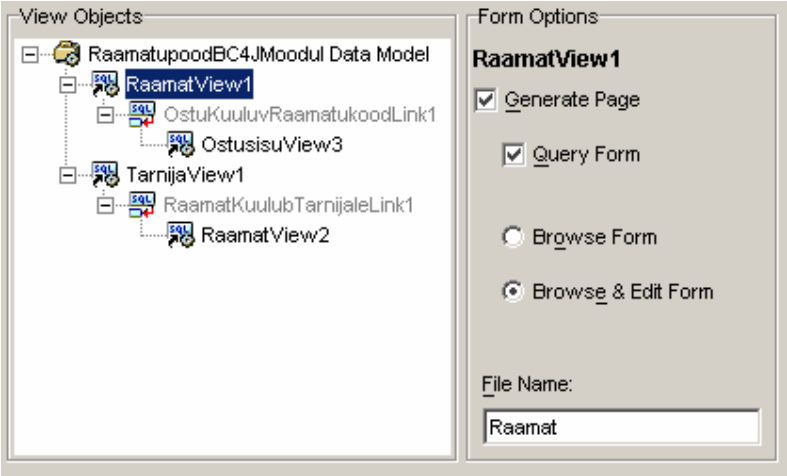
1. Ava uus tühi projekt, valides `RaamatuPood.jws` alammenüüst *New Empty Project*. Nimeta nii kataloog kui ka fail nimega "JSP_projekt" ning vajuta '*OK*'. Salvesta projekt.

Ka JSP rakenduse puhul tuleb luua esmalt vajalik andmemudel.

2. Ava loodud projektifailil parema hiireklõpsuga alammenüü ning vali sealt '*New*'.
3. Mine *Web Tier* sõlme ja vali sealt *JSP for Business Components*. Vali paremast paneelist *Business Components Client Data Model* ning kinnita valik '*OK*'.
4. Määra seaded dialoogiaknaisse nii nagu seda on seletatud peatükis 4 alampunktis 4.6.1 "Kasutajaliidese projekti ja kliendi andmemudeli loomine".

Tulemusena peab tekkima süsteemipuusse `JSP_projekt.jpr` kausta sisse fail `JSP_projekt.cpx`.
Järgmine samm on luua JSP leht.

5. Ava JSP projekti kataloogil parema hiireklõpsuga uuesti '*New*' aken. *Web Tier* komponentide alt vali *JSP for Business Components* ja paremalt paneelist *Compete JSP Application*. Vajuta '*OK*'.
6. Avanenud JSP rakenduse viisardis jälgi, et andmemudeliks oleks valitud "RaamatupoodBC4JMoodul" ning liigu järgmisele lehele.



Joonis 31 - JSP rakenduse viisardi vaateobjektide vormi seaded

7. Määrata vaateobjektid, mida JSP lehelt on võimalik vaadata ja muuta. Selleks vali vasakult vaateobjektide hulgast RaamatView1. Paremast paneelist *Form Options* vali, jäta vaikimisi seaded (vt. Joonis 31).
8. Tee sama TarnijaView1 vaateobjektiga.
9. OstusisuView3 ja RaamatView2 puhul keela *Form Options* paneelist valitud vaateobjekide jaoks lehe loomine, eemaldades märgistuse ruudust *Generate Page*. Vajuta 'Next'.
10. Vaatelinkide vormi lehel aktsepteeri kõik vaikimisi seaded. Liigu edasi ja lõpeta viisard '*Finish*'. Salvesta kogu seni tehtud töö.
11. Käivita JSP rakendus `main.html` faili alammenüüst valides '*Run*'.

Sõltuvalt arvuti võimsusest ja kiirusest see ka vaikimisi määratud brauseris avaneb.
Linkidega navigeerides saab liikuda erinevate vaadete vahel ning sisestada, muuta ja kustutada andmeid.

6 Praktikum II: BC4J rakenduse loomine IDE vahenditega

Selles peatükis luuakse lihtne BC4J loogikal põhinev projekt, kasutades JDeveloper'i integreeritud arenduskeskkonna madalama taseme viisardilisi vahendeid, selleks et tutvustada, millistest sammudest koosneb rakenduse käsitsi ülesehitamine. Rakendus põhineb kahel andmebaasitabelil (tegemist väljamõeldisega – andmed ei pruugi olla adekvaatsed) - `TARNIJA` ja `RAAMAT`, eesmärgiks näidata, kuidas on omavahel võimalik siduda raamatuid tarnijatega. Sageli on tarvilik projekti käigus kasutada ainult madala taseme viisardeid. Kõrgema taseme viisardid, mis kõik töö programmeerija eest ära teevad, ei anna alati igakülgselt rahuldavat tulemust. Hilisem kohandamine ei pruugi olla enam nii lihtne, kui seda algusest peale teha. Seda tähtsam on aga siinkohal terviklik kontseptsioon, et märgata kõiki vajadusi, mis programmi loomisel tuleb täita.
Alljärgnevalt luuakse kolm erinevat projekti, kasutades vaid madalama taseme viisardeid ning kohandades rakendust rohkem iseenda vajadustele.

6.1 BC4J projekti loomine

6.1.1 Tööruumi, projekti ja andmebaasi ühenduse loomine

Kõigepealt tuleks luua uue süsteemi tarbeks uus tööruum.

1. Ava parema hiireklõpsuga süsteemipuus `Workspaces` sõlme alammenüü ja vali '*New Workspace*'.
2. Anna kataloogile ja failile nimeks "TarnijaRaamat". Seekord eemalda linnuke *Add a New Empty Project* eest ning samuti ära märgi märkeruutu *Open in New Navigator*. Kinnita seaded ja valikud '*OK*'. Salvesta.

Edasi tuleb luua BC4J komponente sisaldav projekt.

3. Vali tööruumil `TarnijaRaamat` parema hiireklõpsuga avanevast alammenüüst '*New Project*'.
4. Avanevast projektide nimistust vali vasakpoolsest akna osast *Project Containing New Business Components*. Vajuta '*OK*'.

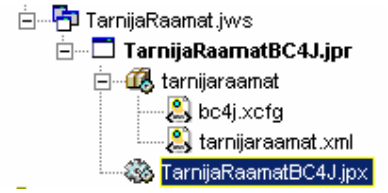
Järgmisena avaneb viisard, kus on võimalik määrata seaded projekti asukoha ja nime kohta.

5. Anna projekti kataloogile ja failile nimeks “TarnijaRaamatBC4J”. Liigu ‘*Next*’ abil järgmisele lehele ja anna vaikimisi projektile nimeks “tarnijaraamat”. Vajuta ‘*Next*’ ja ‘*Finish*’.
6. Avanenud Java ärikomponentide paketi viisardis vajuta ‘*Next*’, kui avanes sissejuhatav leht. Jälgi, et paketi nimi oleks “tarnijaraamat”. Muud seaded jäta sellel lehel nii, nagu viisard vaikimisi pakub (märgitud on *Entity Objects mapped* …). Vajutada ‘*Next*’.

Järgmisena avaneb leht, kus tuleb üle vaadata või vajadusel muuta andmebaasiühenduse seadeid. Vana ühenduse olemasolul võib seda edukalt kasutada. Kui aga eelmist ühendust pole alles, siis ‘*New*’ nupuga saab ühenduse taastada.

7. Vaata üle ühenduse seaded, vajadusel loo uus.
8. Testi ühendust ja vajuta ‘*Finish*’.

Ühenduse loomise viisard sulgub, kui oli vaja luua uus ühendus. ‘Finish’ tuleb vajutada ka kohe pärast ühenduse loomise läbimist Java ärikomponentide paketi viisardis, kuna vastasel juhul loob viisard ise standardse ärikomponentide paketi nagu eelmises näites. Selles praktikumis on ettenähtud luua pakett ise vahendeid valides.



Joonis 32 - Loodud BC4J Java ärikomponentide projekti failid süsteemipuus

Kõiki juhiseid täpselt järgides peaks nüüd olema süsteemipuus näha `TarnijaRaamat`.jws tööruumifail, selle sees `TarnijaRaamatBC4J`.jpr ärikomponentide projekt, mille sees omakorda `tarnijaraamat` pakett ja `TarnijaRaamatBC4J.jpx` andmebaasiühenduse fail (vt. Joonis 32). Paketis `tarnijaraamat` on kaks faili, millest ühte on koondatud informatsioon paketi kohta ja teine, milles olulist infot ei ole (osadel juhtudel ei pruugi seda faili üldse tekkida).

6.1.2 Olemiobjektide ja vaateobjektide loomine

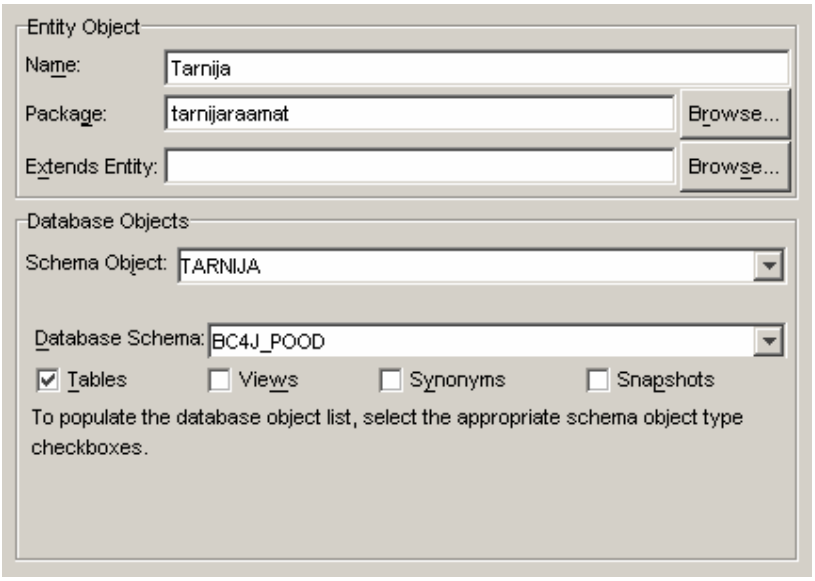
Tekitatud projekti saab edasi arendada tuues sisse ärikomponente. Esmaseks ülesandeks on luua mingi äriloogika, lisades olemiobjekte ja nendel põhinevaid vaateobjekte.

Olemiobjektide lisamiseks tuleb läbida järgmised etapid.

1. Vali süsteemipuust `TarnijaRaamat.jpr` ning parema hiireklõpsuga vali alammenüüst '*New*'.
2. Avanenud komponentide aknas ava vasakust paneelist *Business Tier* sõlm ja vali sealt *Business Components (BC4J)*. Parempoolsest menüüst vali *Entity Object* ning kinnita valik '*OK*'.

Avaneb olemiobjekti viisard, milles saab valida vajalikke olemiobjekte oma projekti.

3. Vajuta '*Next*' juhul kui avanes sissejuhatav leht ning vali *DataBase Objects* alt *Schema Object* rippmenüüst "Tarnija" (eelnevalt tasub tähele panna, et *Database Schema* oleks õige). See täidab ka samal lehel oleva nimelahtri (vt. Joonis 33). Vajuta '*Next*'.



Joonis 33 - Tarnija olemiobjekti seaded olemiobjektide viisardi aknas

Avanenud peaks olema viisard, kus saab atribuute lisada, eemaldada või muuta valitud andmebaasitabelisse kuuluvate olemasolevate atribuutide järjekorda.

4. Jäta alles kõik atribuudid ja vajuta '*Next*'.
5. Atribuutide seadete lehel jälgi, et *Select Attribute* rippmenüüst valides "RegKood" (primaarvõti), oleks sellele atribuudile märgitud järgmised märkeruudud: *Persistent*, *Mandatory*, *Primary Key*, *Queriable*. Vajuta '*Next*'.

Enamusel juhtudel on märkeruudud täidetud vaikimisi sõltuvalt sellest, millist kitsendust (*constraint*) andmebaasi atribuudile rakendatakse (NOT NULL, PRIMARY KEY, FOREIGN KEY jne). *Primary Key* ja *Mandatory Key* on tavaliselt alati vaikimisi märgitud, kuna *Primary Key* ei saa omada tühja väärtust ning *Mandatory* viitabki sellele, et väli on NOT NULL.

6. Järgmisel lehel, kus on Java seaded, vaata, et oleks märgitud *Entity Object Class – Generate Java File* ning meetodiks *Accessors*. Vajuta '*Next*'.
7. Genereerimise lehel jäta märkimata ruut, mis laseb luua vaikimisi vaateobjekti. Vajuta '*Next*' ja avanevalt kokkuvõtvalt lehelt vaata üle seni tehtud korraldused. Kui need sobivad, siis vajuta 'Finish'. Salvesta seni tehtud töö.

Nüüd on olemas lisaks BC4J Java ärikomponentide projektile ka üks olemiobjekt - `Tarnija`. Selle objekti tarbeks tuleb luua ka vaateobjekt.

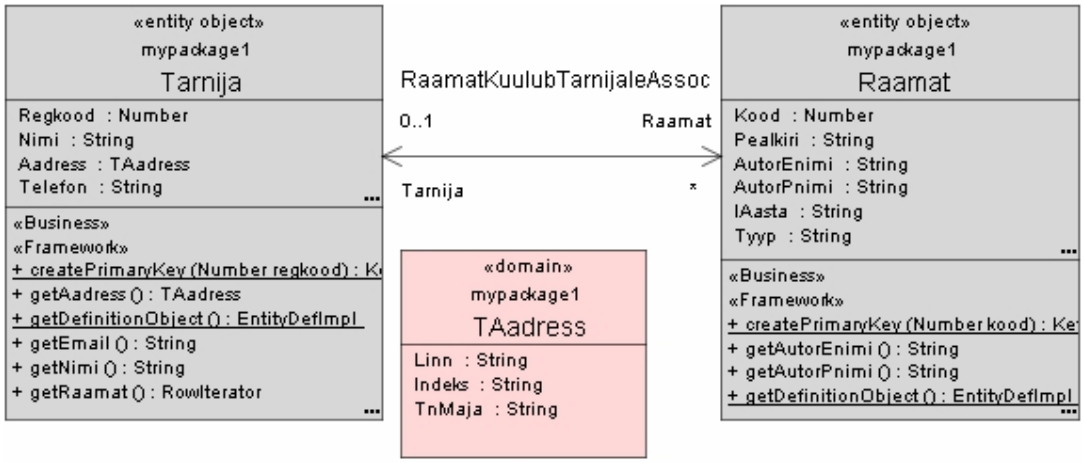
8. Ava olemiobjektil alammenüü ja vali sealt '*New Default View Object*'.

Olemiobjekte ja seoseid nende vahel saab luua ka kasutades selleks UML klassidiagrammi. Selleks luuakse kohe algul loodud projekti sisse UML diagramm, mille jaoks tuleb valida projekti alammenüüst '*New UML Diagram*' (vt. Joonis 34) ning komponentide hulgast '*Class Diagram*'.

Joonis 34 - UML diagrammi valik alammenüüs

Klassi diagrammi puhul tuleb talle määrata pakett (UML diagramm kuulub eraldi paketti), mis erineb rakenduse komponentide põhipaketist. Paketti määramiseks tuleb vajutada rohelise ristiga '*Create Element*' nuppu ja anda paketile nimi.
Diagrammi koostamine toimub andmebaasitabelite lohistamise teel ühenduste alt diagrammi aknasse. Eelnevalt peab olema loodud ühendus vastavasse andmebaasi, kus tabelid asuvad. Selles näites läheks vaja kahte tabelit: Tarnija ja Raamat. Lohistades nad ükshaaval avanenud UML diagrammi lehele, luuakse automaatselt vastavad Java ja XML koodi sisaldavad

olemiobjektid ja ka assotsiatsioon (kui neid on). Käesoleva näite puhul peaks tulemus välja nägema, nagu joonisel 35.



Joonis 35 - UML klassi diagramm

Ülesanded:

- Loo alternatiivselt Tarnija olemiobjektile ka Raamat olemiobjekt.
- Loo Raamat olemiobjektile vaateobjekt.
- Ürita aru saada, millised klassid tekkisid ja mis meetodeid neis kasutatakse.

6.1.3 Seose ja vaatelingi loomine

Eeldusel, et eelmised ülesanded on tehtud, see tähendab, et on loodud ka "Raamat" olemiobjekt ja vaateobjekt, saab hakata siin punktis tegelema ühendustega. Eelmises punktis loodud olemiobjektidega genereeris JDeveloper automaatselt nende olemiobjektide vahele ühenduse, mis põhineb andmebaasi tabelis olevatel võõrvõtmetel.

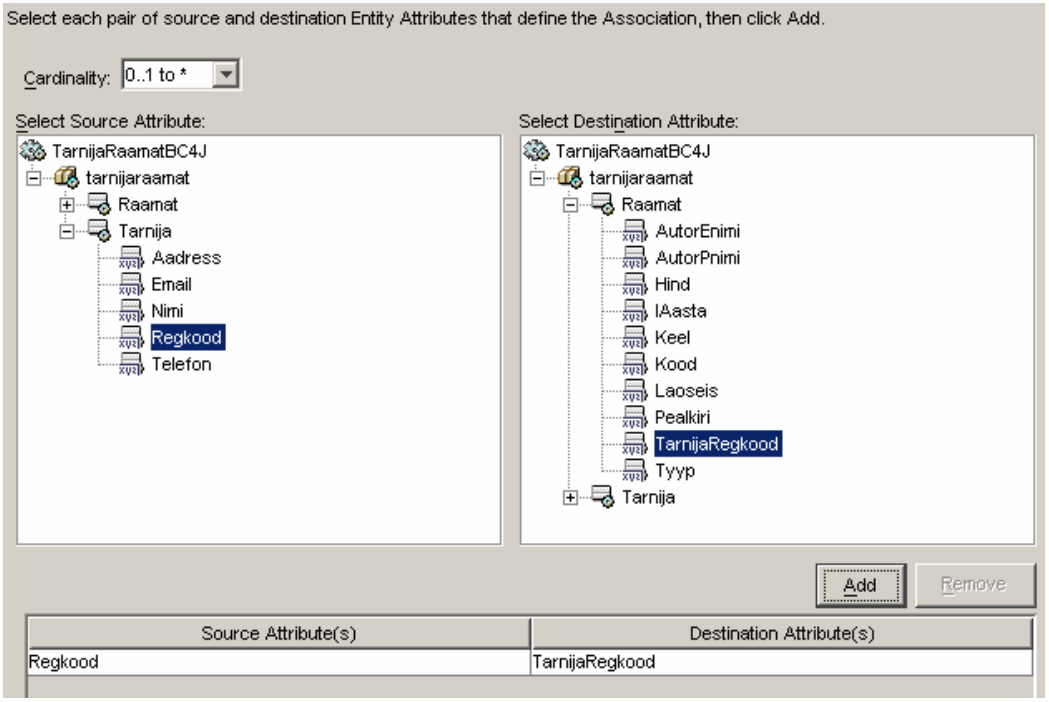
Vastandiks automaatselt genereeritud ühendusele, saab genereerida ka ise ühenduse või üldse mitte siduda kahte olemiobjekti. Kasutades sellist võimalust, pole vahel andmebaasi tabeleid luues üldse tarvidust võõrvõtmete määramise järele.

Praegu on süsteemipuus näha automaatselt loodud `RaamatKuulubTarnijaleAssoc` assotsiatsioon . See on täiesti töötav, kuid ülevaatlikkuse mõttes võiks selle ise uuesti luua.

1. Kustuta vana assotsiatsioon klõpsates parema hiireklahviga failile `RaamatKuulubTarnijaleAssoc` ning vali alammenüüst '*Erase RaamatKuulubTarnijaleAssoc from disk*', mis kustutab faili ka kõvakettalt.

2. Loo assotsiatsioon valides kas Tarnija või Raamat olemiobjekti ning vali '*New Association*' alammenüüst.
3. Avanenud assotsiatsiooni viisardis liigu sissejuhatavalt lehelt edasi järgmisele lehele, kus anna nimeks "TarnijaRaamatAssoc". Jälgi, et paketi nimi oleks "tarnijaraamat". Vajuta '*Next*'.

Avaneb leht olemiobjektidega, kus tuleb luua seos olemiobjektide Tarnija ja Raamat vahel (vt. Joonis 36).



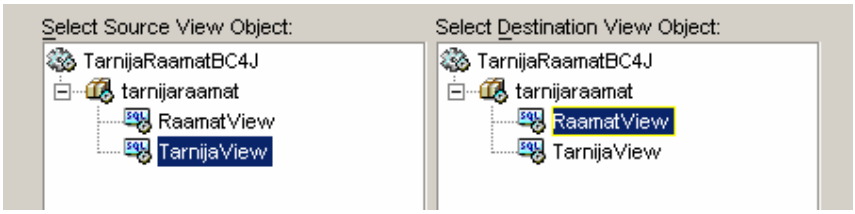
Joonis 36 - Tarnija-Raamat seose määramine

4. Ava vasakult paneelist "Tarnija" olemiobjekt ja vali atribuut "Regkood".
5. Paremast paneelist ava "Raamat" olemiobjekt ja vali sealt võõrvõtmeks "TarnijaRegkood". Vajuta '*Add*', mis lisab valitud atribuudid alumisse paneeli. Akna ülemises osas jälgi, et seos oleks määratud 0...1 to *. Vajuta '*Next*'.
6. Järgmisel lehel jäta vaikimisi seaded ja liigu edasi. Kokkuvõtval lehel veendu, et tehtud valikud edastavad vajaliku informatsiooni. Vajuta '*Finish*'. Salvesta kogu seni tehtud töö.

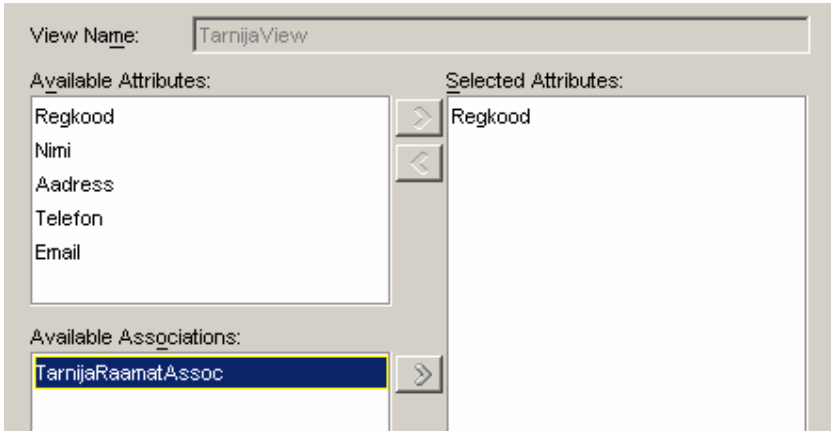
6.1.4 Vaatelingi loomine

Eelmises punktis loodud assotsiatsioonile on vaja luua ka vaatelink, mille põhjal *master-detail* rakendus informatsiooni esitleb.

1. Ava `TarnijaRaamatAssoc` alammenüü ja vali sealt '*New View Link*'.
2. Avanenud viisardis anna nimeks "TarnijaRaamatLink" ning liigu järgmisele lehele, kus tuleb määrata vaateobjektid.
3. Vali allika vaateobjektiks "TarnijaView" ning sihi vaateobjektiks "RaamatView" (vt. Joonis 37). Vajuta '*Next*'.



Joonis 37 - Vaateobjektide määramine vaatelingi "TarnijaRaamatLink" tarbeks



Joonis 38 - Tarnijavaate atribuutide lisamine vaatelingile "TarnijaRaamatLink"

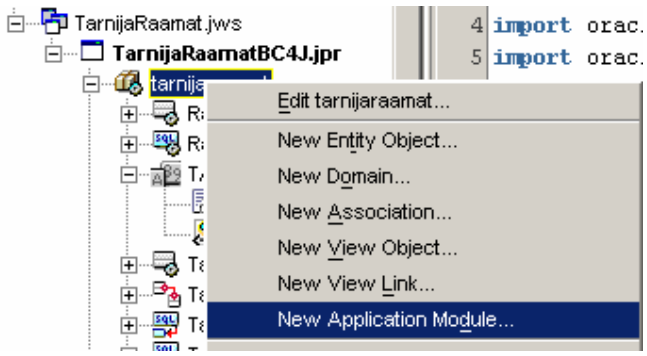
4. Järgmisel lehel vali allika atribuudiks *Available Associations* alt "TarnijaRaamatAssoc", mis automaatselt lisab paremasse paneeli atribuudi "Regkood" (vt. Joonis 38). Liigu järgmisele lehele.

Sihtatribuutide lehel peaks olema juba paremale paneeli valitud "TarnijaRegkood", kuna eelmisel lehel sai valitud atribuudid assotsiatsiooni kaudu. Kui seda pole siis tuleb korrata sama tegevust, nagu allikaatribuutide lehel.

5. Veendu, et sihtatribuutide lehel oleks paremale paneeli valitud “TarnijaRegkood”. Vajuta ‘*Next*’ ja kontrolli ‘*Test*’ nupu abil, kas päring töötab. Sulge teatise aken, kui see kinnitab korrektset päingut. Liigu järgmisele lehele.
6. Sea vaateobjektile “TarnijaView” seosetüüp “0..1” ning vaateobjektile “RaamatView” seosetüüp “*”. Ülejäänud seaded jäta nii nagu viisard seda vaikimisi pakub. Vajuta ‘*Next*’ ja ‘*Finish*’. Salvesta kogu töö.

6.1.5 Rakenduse mooduli loomine

1. Vali süsteemipuus `tarnijaraamat` paketi alammenüüst ‘*New Application Module*’ (vt. ka Joonis 39).

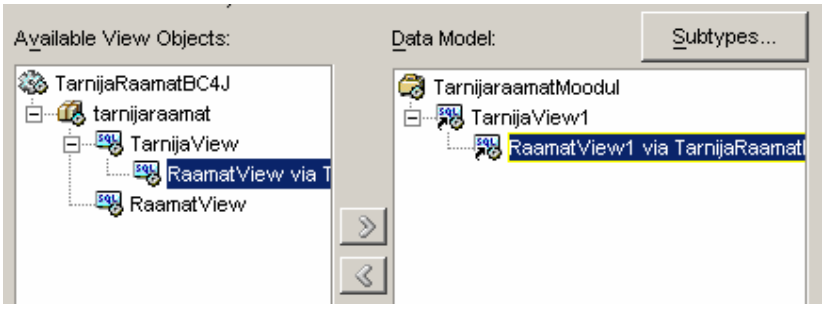


Joonis 39 - Valik rakendusemooduli viisardi avamiseks

2. Avanenud rakendusemooduli viisardis vajuta ‘*Next*’, kui avanes sissejuhatav leht ning anna moodulile nimeks “TarnijaraamatMoodul”. Vajuta ‘*Next*’.

Järgmisena avaneb andmudeli leht, kus tuleb seada vaateobjektid sellisel põhimõttel, et raamatuvaade “RaamatView” on tarnijavaate “TarnijaView” järgi määratud. See tähendab, et valides mingi tarnija, saab selle tarnijale kuuluvate raamatute nimistu. Tee viisardis järgmised seaded *Data Model* lehel:

3. Vali vasakust paneelist olemasolevate vaateobjektide seast “TarnijaView” ning parema noolenupuga liiguta see parempoolsesse valitud vaateobjektide aknasse.
4. Raamatuvaate sidumiseks moodulis vaateobjektiga “TarnijaView”, vali vasakust paneelist “RaamatView via TarnijaRaamatLink” ning paremast paneelist märgi ära seal olev “TarnijaView”. Vajuta paremat noolenuppu, mille järel valitud vaateobjektide aknasse ilmub “TarnijaView” ja tema alamobjektiks “RaamatView via TarnijaRaamatLink” (vt. Joonis 40). Vajuta ‘*Next*’.



Joonis 40 - Vaateobjektide valik andmemudelile

5. Järgmistel lehtedel jäta vaikimisi seaded ja vaata üle kokkuvõttev leht. Vajuta '*Finish*'. Salvesta kogu töö.
6. Testi valmis moodulit.

6.2 *Master-Detail* Java rakenduse loomine

Kogu varasem esimeses praktikumis tehtud rakendus sai loodud JDeveloperi genereerivaid vahendeid kasutades. Seekord tuleb JClient vorm luua käsitsi. Sageli on lihtsam kõik komponendid algusest peale ise ehitada, kui hakata genereeritud vormi kohandama.

Master-detail Java rakenduse käsitsi loomine on rohkem aega nõudvam, kuid annab enam vabadust näiteks vaateelementide ja paneelide väljanägemise kohandamise osas kui seni tehtud äriloogikal põhineva süsteemi ülesehitamine.

Loodava Java rakenduse põhisammud on järgmised:

- Rakenduse projekti loomine
- Projekti sidumine BC4J ärikomponentide projektiga
- Tarnijavaate paneeli loomine
- Raamatuvaate paneeli loomine
- Kasutajaliidese kohandamine

6.2.1 Java rakenduse projekti loomine ning sidumine BC4J ärikomponentide projektiga

Java rakenduse jaoks tuleb esmalt luua uus omaette projekt.

1. Vali süsteemipuus `TarnijaRaamat.jws` tööruumi alammenüüst '*New Empty Project*'.
2. Anna *New Project* dialoogiaknas kataloogile ja failile nimeks "UI_manuaalProjekt". Vajuta '*OK*' ja salvesta tööruum.

Järgmiseks tuleb siduda omavahel loodud Java rakenduse projekt ja BC4J ärikomponentide projekt.

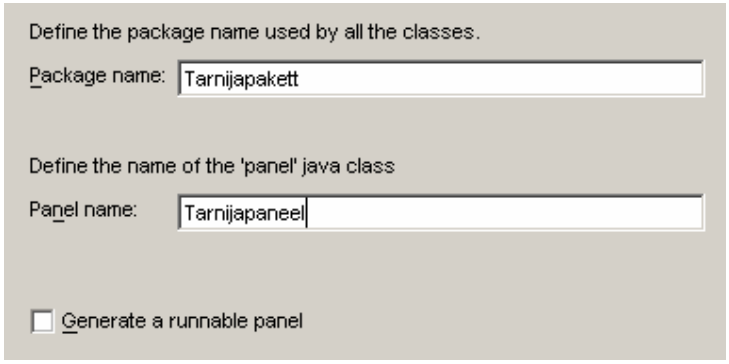
3. Vali `UI_manuaalProjekt.jpr` projekti alammenüüst '*New*'. Ava *Client Tier* sõlm ning vali sealt *Swing/JClient for BC4J*. Komponentide nimistust paremalt paneelist vali *Business Components Data Model*.
4. Andmemudeli viisardis *Definition* lehel veendu, et vaikimisi täidetud väljad oleks õiged.
5. Järgmisel lehel vaata, et kirjelduse nimeks oleks "TarnijaraamatMoodul". Vajuta '*Next*' ja '*Finish*'. Salvesta kogu seni tehtud töö.

Süsteemipuusse peab olema tekkinud fail `UI_manuaalProjekt.cpx`.

6.2.2 Rakenduse master-paneeli loomine

Töö Java rakenduse kasutajaliidesega algab paneelide loomisega. Esmalt tuleb luua paneel, mis hoiab endas tarnijavaate esitlemiseks vajalikke komponente. Seekord tuleb alustada tühja paneeliga.

1. Ava käesoleval projektisõlmel uuesti '*New*' aken. Vali *Client Tier* alt *Swing/JClient for BC4J*. Komponentide alt vali *Empty Panel* ning vajuta '*OK*'.
2. Avanenud viisardis jäta esimesel lehel kõik seaded samaks ning teisel lehel jälgi, et paketi nimi oleks "Tarnijapakett" ning anna paneelile nimeks "Tarnijapaneel".

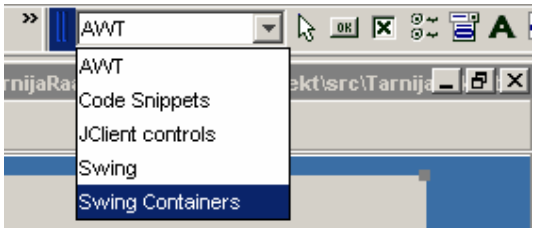


Joonis 41 - Tühja *master*-paneeli viisardi kataloogi ja faili nimeseaded

3. Jäta märkimata *Generate a runnable panel*. Vajuta '*Next*' ja '*Finish*'(vt. Joonis 41). Salvesta projekt.

Töölaual on nüüd avatud kasutajaliidese redaktori aknas `Tarnijapaneel.java` fail.
Oluline osa tööst tuleb teha kasutades peamiselt struktuuri akna elemente, kuhu saab Java komponente lisada, ning omaduste inspektorit (*Property Inspector*) nende kohandamiseks.
Järgnevalt tuleb lisada paneeli mõned komponendid. Selleks on vaja kõigepealt luua konteiner.

4. Kui `Tarnijapaneel.java` fail pole veel visuaalse töölaua aknas avatud, ava see valides `Tarnijapaneel.java` faili alammenüüst '*UI Editor*'.
5. Struktuuriaknast vali `this`-sõlm ning omaduste inspektoris vali *layout* väljal rippmenüüst *BorderLayout*. Salvesta tööruum.
6. Konteineri loomiseks vali ülevalt komponentide paleti rippmenüüst *Swing Containers* (vt. Joonis 42) ning võta komponent *JPanel*.



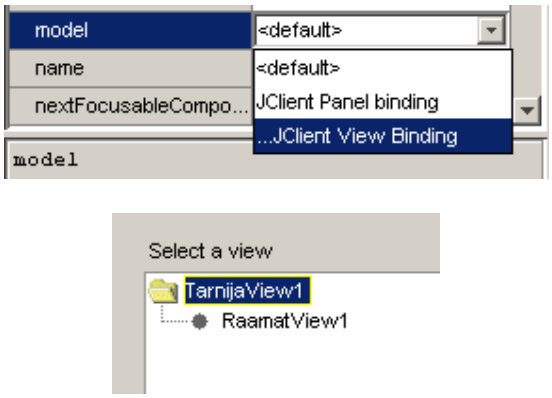
Joonis 42 - Java kasutajaliidese komponentide paleti rippmenüü

7. Mine struktuuri aknasse ja ning vajuta `this`-sõlme ikoonil, mille järel tekib `this`-sõlme alla uus sõlm `JPanel`.
8. Vali see `JPanel`-sõlm ning liigu omaduste inspektorisse. Otsi üles *name* väli ja kirjuta sinna "Tarnijapaneel". *Constraints* väli määta "*Center*", mille peale tekib redaktorissse akna keskele must ruut. Paigutuseks vali *layout* väljale *GridBagLayout*. Salvesta.

GridBagLayout asemel võib vabalt ka muid paigutuse tüüpe kasutada, kuid visuaalse tööruumi kasutamise korral on *GridBagLayout* algajale lihtsam kasutada, kuna objektid paigutatakse paneelidesse lahtrite abil ning nende asukoht ja suurus on määratud lihtsalt muudetavate omadustega. Teisel juhul võib kasutada niinimetatud "null" paigutust, mis võimaldab objekte ekraanil lohistada.

Järgnevalt tuleb lisada navigeerimisriba, mille abil saab andmete kuvamist ekraanil juhtida.

9. Vali sruktuuriaknast `this(BorderLayout)`-sõlm ning komponentide paleti rippmenüüst *JClient controls*. Komponentide alt vali *JUNavigationBar* ning klõpsa hiirega uuesti `this`-sõlmel, mille järel paneeli keskele ilmub navigeerimisriba. Paigalda ta omaduste inspektoris *constraints* omadust muutes ülemisse serva ning anna nimeks "TarnijaRiba".
10. Järgmiseks seo navigeerimisriba tarnija BC4J ärikomponendiga. Selleks vali omaduste inspektoris *model*-väljal rippmenüüst *JClient View Binding* ning vali avanenud aknast TarnijaView (vt. Joonis 43). Vajuta '*OK*' ja salvesta seni tehtud töö.



Joonis 43 - Navigeerimisriba sidumine vaateobjektiga

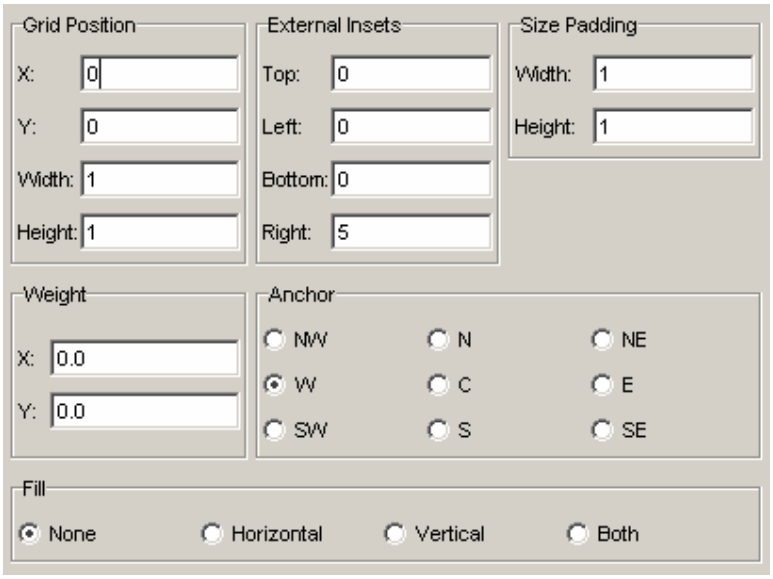
Järgnevalt tuleb lisada tarnijapaneelile tekstiväljad.

11. Vali komponentide paletist *Swing* komponentide alt *JLabel* ning seejärel struktuuri aknast `Tarnijapaneel`-sõlm. Anna nimeks "TarnRegkoodSilt".

Redaktorisse tekib üles vasakusse nurka sildilahter. Selle asukoha muutmiseks tuleb kasutada *constraints* seadeid.

12. Vajuta paneelile tekkinud sildilahtril parema hiireklahviga ning vali alammenüüst *Constraints Window*.
13. Anna väljadele väärtused, mis on toodud joonisel 44. Kinnita valikud.

Võib juhtuda, et vahepeal on tarnijapaneeli paigutus paigast nihkunud ning selle tõttu ei liigu sildilahter paneeli keskele. Sel juhul tuleks kontrollida, et `Tarnijapaneel` oleks seatud keskele (*Center*) ning seejärel püüda paigutada silti uuesti.



Joonis 44 - Komponentide positsioneerimise aken

14. Muuda sildi nime kirjutades *text*-väljale omaduste inspektoris "Registrikood". Salvesta kogu töö.
15. Tekstivälja loomiseks kasuta *JTextField* komponenti ning lisa ta `Tarnijapaneel`-sõlme.
16. Muuda paigutust nii, et tekstiväli asuks sildi kõrval. Anna tekstiväljale nimeks "RegkoodVali".

Järgmiseks tuleb loodud tekstiväli siduda Tarnija tabeli atribuudiga "Regkood".

17. Vali `Tarnijapaneel`-sõlme omaduste inspektorist *document* rippmenüüst *JClient Attribute Binding* ning vali avanenud aknast vaatevaliku paneelist "TarnijaView" ning kõrvalt atribuutide paneelist "Regkood". Vajuta '*OK*'.
18. Muuda tekstivälja laiust, andes omaduste inspektoris *column* väljale väärtuseks 10. Salvesta kogu töö.

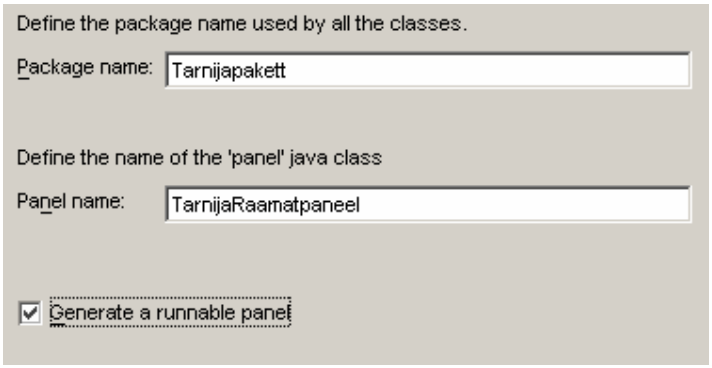
Ülesanne:

- Loo eelneva õpetuse põhjal tarnijapaneeli veel mõned sildid ja tekstiväljad.

6.2.3 Testimiseks ettevalmistus ja testimine

Mingis etapis tuleb loodavat Java rakendust testida. See on vajalik veendumiseks, et loodud paneel töötab korralikult enne, kui ehitada juurde teine paneel. Selleks tuleb luua rakendusele `main()` meetodit sisaldav põhipaneel, mille kaudu rakendust käivitada saab.

1. Vali `UI_manuaalProjekt.jpr` alammenüüst '*New*'. Vali tühi paneel (*Empty Panel*) *Swing/JClient for BC4J* komponentide seast.
2. Viisardis veendu, et andmemudeli kirjeldust sisaldaks "TarnijaRaamatMoodul".
3. Järgmisel lehel jäta paketi nimeks vaikimisi pakutud "Tarnijapakett" ning paneelile anna nimeks "TarnijaRaamatpaneel". Märgi ka märkeruut *Generate a runnable panel*, mis loob `main()` meetodit sisaldava Java faili (vt. Joonis 45). Vajuta '*Next*' ja '*Finish*'.



Joonis 45 - Tühja põhipaneeli seaded

Visuaalsel töölaual peaks olema nüüd avatud `TarnijaRaamatpaneel.java` fail.

4. Vali struktuuriaknast `this`-sõlm ning omaduste inspektoris määra paigutuse tüübiks *layout*-väljal *BorderLayout*.

Järgmiseks sammuks on lisada põhipaneeli sisse tarnijapaneel. Selleks on vaja lisada uus paan, mille mõõtmeid on hiire abil võimalik soovi järgi muuta.

5. Vali *Swing Containers* komponentide hulgast *JScrollPane* ja vajuta `this`-sõlmel. Omaduste inspektoris määra ta asukoht paneeli ülemisse ossa ning anna nimeks "TarnijaPaan".

Selleks, et paanil eelmises osas loodud sildid ja väljad näha oleks, on vaja lisada paar rida `TarnijaRaamatpaneel.java` faili.

6. Vali `TarnijaRaamatpaneel.java` alammenüüst *Code Editor*. Otsi koht, kus defineeritakse objekte ja lisa sinna järgmine lause:

```
private Tarnijapaneel tarnijaPaneel;
```

7. Täienda `jbInit()` meetodit järgmiste lausetega (vt. Joonis 46):

```
tarnijaPaneel = new Tarnijapaneel(panelBinding);
TarnijaPaan.getViewport().add(tarnijaPaneel);
```

```
/**
 *
 * the JbInit method
 */
public void jbInit()
{
  this.setLayout(borderLayout1);
  this.add(TarnijaPaan, BorderLayout.NORTH);
  tarnijaPaneel = new Tarnijapaneel(panelBinding);
  TarnijaPaan.getViewport().add(tarnijaPaneel);

}
```

Joonis 46 - jbInit() meetod TarnijaRaamat.java failis peale tarnijapaneeli lisamist

8. Kompileeri fail, valides alammenüüst '*Make*'. Salveta kogu tehtud töö.
9. Testi, valides `TarnijaRaamatpaneel.java` alammenüüst '*Run*'. Keri kirjeid ja veendu, et väljadele ilmuksid õiged vasted. Vajadusel muuda tekstiväljade pikkust.

6.2.4 Rakenduse detail-paneeli loomine

Detail paneeli loomine toimub analoogselt master paneeli loomisele.

1. Loo `UI_manuaalProjekt.jpr` alla uus tühi paneel.
2. Nimeta paneel "Raamatupaneel" ning jäta märkimata *Generate a runnable Panel*.
3. Sea loodud paneeli paigutuse tüübiks *BorderLayout*.
4. Ava `this(BorderLayout)`-sõlm, vali sealt `BorderLayout1` ning muuda nimi, kirjutades sinna "raamatPaigutus". Salvesta.
5. Loo raamatupaneelile konteiner nagu tarnijapaneelil. Anna nimeks "Raamatpaneel" ning paiguta ta keskele. Paigutuse omaduseks määra *GridBagLayout*.
6. Lisa konteineri külge ka kerimispaan (*JScrollPane*). Nime väljale kirjuta "RaamatPaan".

Järgmisena tuleb lisada rakenduse *detail*-ossa tabel. Selleks on Java Swing komponentide seas olemas komponent *JTable*.

7. Vali Swing komponentide seast *JTable* ning aseta ta sõlme `RaamatPaan`.
8. Anna tabelile nimeks "RaamatTabel".
9. Vali *model*-rippmenüüst *JClient Attribute list binding*, et siduda tabel andmebaasis olevate tabeli kirjetega. Avanenud mudeli seadeid pakkuvast aknast vali RaamatView1 ning lisa mõned atribuudid. Vajuta '*OK*' ja salvesta tööruum.

Viimatise töö tulemusena on vajalik *detail*-paneel valmis.
Raamatupaneeli kuvamiseks põhipaneelil "TarnijaRaamatpaneel", tuleb sinna luua esmalt kerimispaan.

10. Ava visuaalsel töölaual (*UI Editor*) `TarnijaRaamatpaneel.java` ning vali *Swing Container* komponentide hulgast *JscrollPane* ning lisa ta `this`-sõlme.
11. Anna nimeks "RaamatuPaan" ja paiguta keskele. Salvesta kogu töö.

Järgnevalt tuleb täiendada faili, milles asub põhipaneeli kood.

12. Ava fail `TarnijaRaamatpaneel.java` koodiredaktori aknas.
13. Lisa defineering:

```
private Raamatupaneel raamatuPaneel;
```

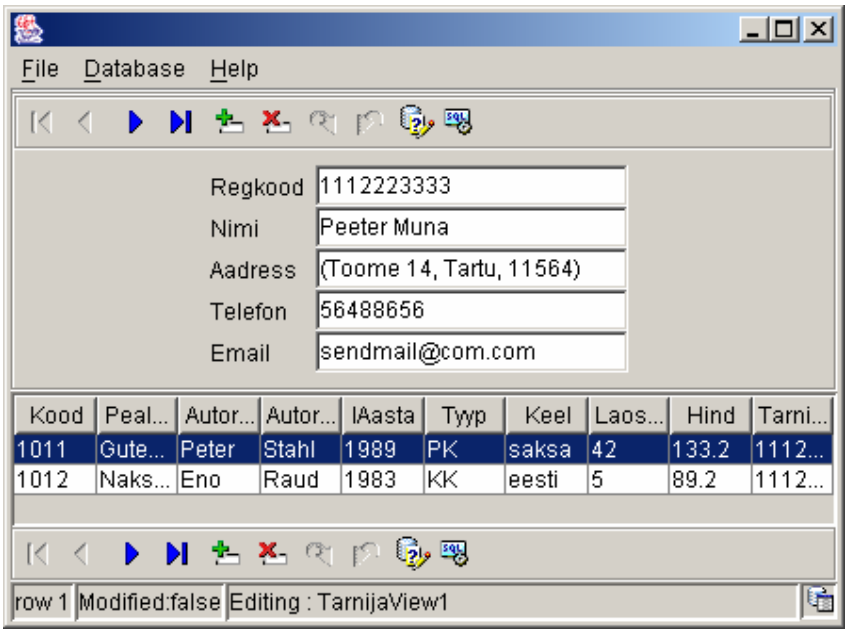
```
/**
 *
 * the JbInit method
 */
public void jbInit()
{
  this.setLayout(borderLayout1);
  this.add(TarnijaPaan, BorderLayout.NORTH);
  this.add(RaamatuPaan, BorderLayout.CENTER);
  tarnijaPaneel = new Tarnijapaneel(panelBinding);
  TarnijaPaan.getViewport().add(tarnijaPaneel);
  raamatuPaneel = new Raamatupaneel(panelBinding);
  RaamatuPaan.getViewport().add(raamatuPaneel);

}
```

Joonis 47 - jbInit() meetod peale raamatupaneeli lisamist

14. Lisa `jbInit()` meetodi sisse järgmised laused (vt. ka Joonis 47):

```
raamatuPaneel = new Raamatupaneel(panelBinding);
RaamatuPaan.getViewport().add(raamatuPaneel);
```

15. Kompileeri fail ning käivita (vt. Joonis 48).



Joonis 48 - Valminud JClient rakendus

Ülesanded:

- Uuri kasutajaliidese loomise käigus tekkinud Java faile.
- Muuda koodis rakenduse akna suurust.
- Muuda koodis raamatupaneeli tulpade pealkirju ja laiusi, kasutades selleks meetodeid `setHeaderValue()` ning `setMaxWidth()`.

6.3 JSP rakenduse loomine

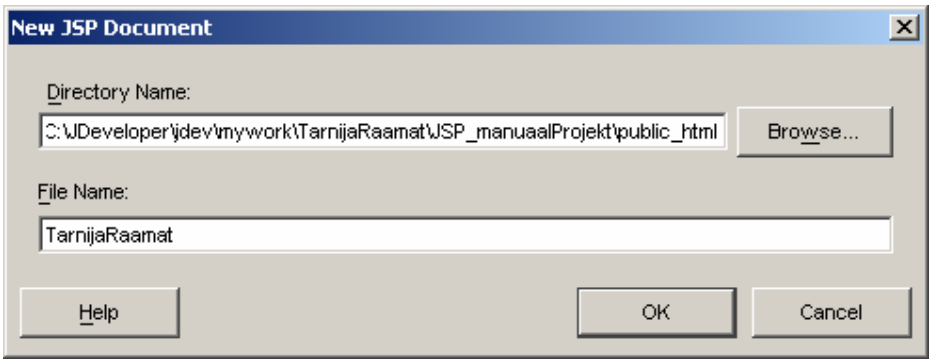
Selles punktis luuakse veebirakendus, mille eesmärgiks on kuvada tarnijale kuuluvaid raamatuid. Osa tööst seisneb ka HTML koodi lisades ja muutes.

Enne kui asuda JSP rakendust ehitama, tuleb tähele panna, et JDeveloper kirjutab kasutajaliidesesse atribuutide nimed ja väljad vaikimisi BC4J kihi vaateobjektide nimede järgi. Enamasti on vaja nimesid muuta, et need kasutajale arusaadavamad oleks. Muutusi saab teha BC4J ärikomponentide projektis (näiteks `TarnijaRaamatBC4J.jpr`), vaateobjekti atribuutide viisardis. Selleks valida mingi vaateobjekt (näiteks `TarnijaView`) ning

struktuuriaknast mingi atribuut ja teha sellel hiirega topeltklõps. Viisardis *Control Hints* lehel tuleb kirjutada väljale *Label Text* soovikohane nimetus. Andmebaasis need ei muutu.

6.3.1 Projekti loomine ja sidumine BC4J ärikomponentide projektiga

1. Loo `TarnijaRaamat.jws` tööruumi uus tühi projekt nimega "JSP_manuaalProjekt". Salvesta.
2. Vali loodud projektifaili alammenüüst '*New*'.
3. Ava *Web Tier* sõlm ning vali *JavaServer Pages (JSP)* komponentide hulgast *JSP Page*.
4. Avanenud dialoogiaknas muuda failinimi, kirjutades sinna "TarnijaRaamat". Kataloogi nimi jäta muutmata (vt. Joonis 49). Vajuta '*OK*'.



Joonis 49 - JSP projekti kataloogi- ja nimeseaded

Süsteemipuusse lisandus kaks faili: `TarnijaRaamat.jsp` ning `web.xml`.
Järgmiseks tuleb kohandada faili `TarnijaRaamat.jsp`. Vaikimisi on ta koodiredaktoris avatud ning temaga on koos ka komponentine palett.

5. Muuda vastavalt soovile lehe pealkirja taagide `<title>…</title>` vahel.
6. Lisa taagide `<body>..</body>` sisse üksteise alla kaks pealkirja "Tarnija" ja "Raamatud".

```
<body>
      <h2>The current time is: </h2>
      <p><jsp:expression> new java.util.Date() </jsp:expression></p>
      <h2>Tarnija</h2>
      <h2>Raamatud</h2>
</body>
```

Salvesta projekt.

Tarnija pealkirja alla oleks loogiline lisada nüüd ka vastav vaade. Selleks, et andmeid sinna kuvada, on vaja need andmed BC4J ärikomponentde paketist kätte saada. Tuleb luua vahendaja kihiks andmemudel nagu iga teisegi kasutajaliidese rakenduse puhul.

7. Ava pojektifaili alammenüüst '*New*' ning vali *Web Tier* sõlmest *JSP for Business Components* komponentide loetelust *Business Components Client Data Model*.
8. Andmemudeli viisardis aktsepteeri kõik vaikimisi seaded, misjärel süsteemipusse lisandub fail `JSP_manuaalProjekt.cpx`. Salvesta kogu projekt.

Nüüd on võimalik lisada vaatekomponente.

9. Paiguta kursor `TarnijaRaamat.jsp` failis rea `<h2>Tarnija</h2>` alla. Vali komponentide paletiks rippmenüüst *BC4J Connections* ning komponentide hulgast vali *ApplicationModule*.
10. Avanenud aknas veendu, et andmemudeli kirjeldus oleks "TarnijaraamatMoodul", ülejäänud lehtedel jäta kõik vaikimisi seaded. Vajuta '*Finish*'. Salvesta kogu seni tehtud töö.

`TarnijaRaamat.jsp` faili lisandus rida, mis loob sideme JSP veebirakenduse ja BC4J ärikomponentide paketti kuuluva rakenduse mooduli vahel.

```
<jbo:ApplicationModule id="TarnijaraamatMoodul"
definition="JSP_manuaalProjekt.TarnijaraamatMoodul"
releasemode="Stateful"/>
```

6.3.2 Andmeväljade lisamine JSP veebirakendusse ning lõppviimistlus

Andmeväljade lisamise korral on oluline tähele panna, et vajalikud vaateobjektid õigetele väljadele satuks. Iga andmevälja ette tuleb lisada ka andmesilt, mis viitab mingile vaateobjektile.

1. Paiguta kursor `TarnijaRaamat.jsp` failis viimati lisatud rakendusemooduli alla. Loo andmesilt rakenduse *master*-ossa Tarnijavaate tarbeks. Selleks vali BC4J komponentide paletist *DataSource* komponent ning vali vaateobjektide hulgast `TarnijaView1` ning kirjuta *id*-väljale "tarnija_id". Vajuta '*Finish*'.

Koodiredaktorisse lisandus järgmine rida:

```
<jbo:DataSource id="tarnija_id"  appid="TarnijaraamatMoodul"
viewobject="TarnijaView1"/>
```

2. Lisa samuti andmesilt *detail*-ossa Raamatuvaate tarbeks rea `<h2>Raamatud</h2>` alla. *Id*-väljale kirjuta "raamat_id" ning kuvatavate ridade arvuks määra *rangesize*-väljal "5".

Enamuse komponentide lisamisel rakendusse luuakse nende komponentide kohta ka vastav `.jsp` fail. Nende failide abil on lihtne koodi kaudu rakenduses muudatusi teha. JDeveloper genereerib kõik elemendid tuginedes inglise keelele. Seetõttu tuleb ise piisavalt koodis elementide nimesid muuta, et kasutajale esitletav leht valdavalt eestikeelne oleks. Järgnevate komponentide lisamisel rakendusse tasub kindlasti üle vaadata ka lisatud `.jsp` fail. Kellele HTML keel natukenegi tuttav, peaks taipama, kuidas koodi kohandada.

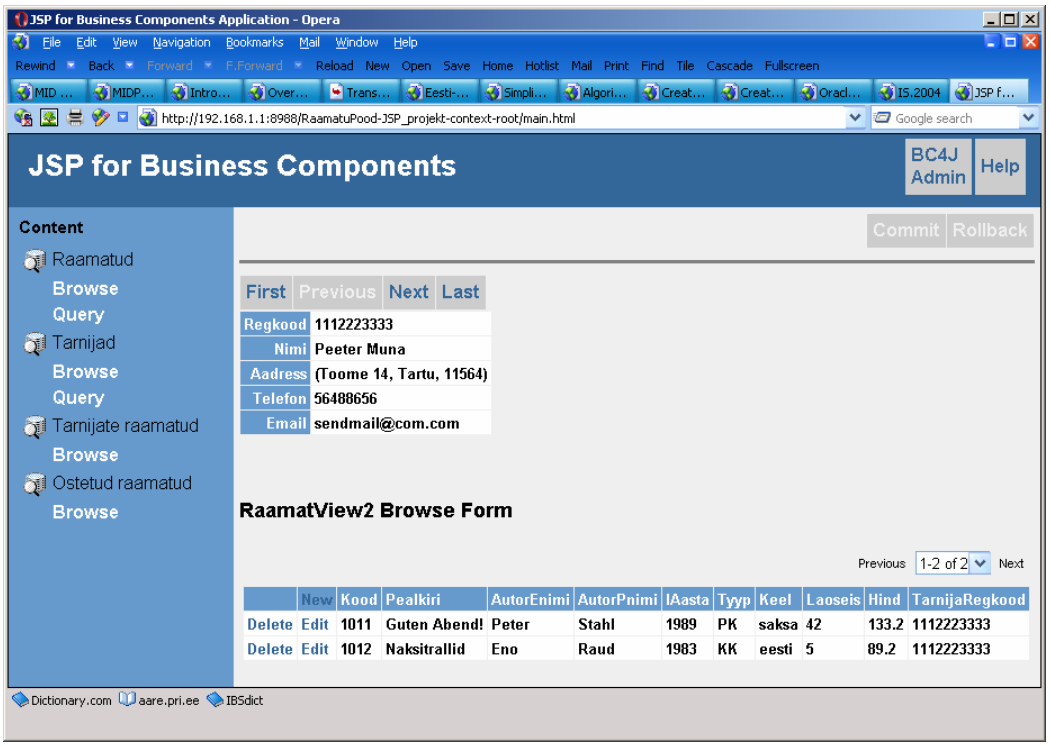
Lehekülgede vahetamiseks või kirjete edasi-tagasi kerimiseks lisatakse tavaliselt veebilehele ka vastavad lingid. Enamasti on nad lihtsalt ühesõnalised väljendid stiilis "Edasi", "Tagasi", "Eelmine", "Järgmine" või muud kasutajale arusaadavad viited. Selleks, et JSP rakenduse puhul vastava lingi vajutamise korral vaated vahetuksid, on vaja andmehaldurit.

3. Aseta kursor uuesti rea `<h2>Raamatud</h2>` alla, viimase lisatud koodirea järele. Vali komponentide paneliks *BC4J Component Tags* ning otsi üles *DataHandler* komponent. Mooduliks vali rippmenüüst "TarnijaRaamatmoodul" ning vajuta '*Finish*'. Salvesta kogu töö.
4. Järgmiseks lisa navigeerimise lingid, kasutades selleks komponenti *Data Navigate*. Vali *datasource*-rippmennüüst "tarnija_id", et linke rakendataks Tarnijavaatele.
5. Viisakama kuju andmiseks muuda failis `DataNavigateComponent.jsp` linkide nimed eestikeelseteks.
6. Komponent *DataRecord* kuvab andmebaasis olevad andmeid. Vali *DataRecord* komponentide paletilt ning *datasource*-väljale vali rippmenüüst "tarnija_id".

Sellega on töötav JSP rakenduse *master*-osa valmis.

7. Lisa ka *detail*-osa, kuhu kuvatakse andmed raamatute tabelist. Selleks paiguta kursor rea `<h2>Raamatud</h2>` alla viimase sinna lisatud koodirea järele ning vali komponentide hulgast *DataTabel*. Andmeallika väljale vali rippmenüüst "raamat_id". Vajuta '*Finish*'. Salvesta kogu töö.

8. Käivita JSP veebirakendus, valides `TarnijaRaamat.jsp` alammenüüst '*Run*' (vt. Joonis 50).
9. Vaata brauseris loodud rakendus üle, katseta ja tee soovi korral muutusi.



Joonis 50 - Valminud JSP rakenduse "tarnijate raamatud" vaade

Kokkuvõte

Õppematerjali koostaja ei ole professionaalne Oracle9*i* JDeveloper arenduskeskkonna kasutaja. Töö käigus tuli palju tundma õppida äriloogikat ja aru saada ka BC4J Java ärikomponentide kasutamise põhimõtetest. Selletõttu on töös järgitud stiili, mida on kasutatud erinevates teistes sarnastes õppematerjalides. Pikkadest ja keerulistest õppematerjalidest on välja sünteesitud lihtne ja arusaadav algajale tarvilik õpetus. Aluseks välja mõeldud infosüsteemi "Raamatupood" peaks olema piisavalt lihtne mõista ja katsetada.

Töö tulemusena valmis õppematerjal, mis annab algajale mitmekülgse ülevaate JDeveloper arenduskeskkonnast ja tema komponentidest. Samuti arusaamise BC4J Java ärikomponentidest ja nende kasulikkusest ja vajalikkusest rakenduste loomisel.

Õppematerjali lugedes peaks kasutaja olema saanud teada, mis on BC4J Java ärikomponendid, milleks neid kasutatakse ja kuidas nad on organiseeritud. Samuti teab ta, mis on kolmekihiline süsteem "mudel-vaade-kontroller" ning milleks see vajalik on. Kui praktiline osa on vaid läbi loetud, peaks kasutaja enam-vähem ette kujutama, kuidas reaalsuses rakenduse loomine välja võiks näha ja milline ta tulemus olla võiks.

Õppematerjali praktilise osa valmis saanud kasutajal peaks olema ülevaade, kuidas tavaliselt rakenduste tegemine algab ja mida tuleb erinevates etappides teha ja mis komponente kasutada, et soovitud tulemuseni jõuda. Ülesannete lahendamise käigus tekkinud praktiline kogemus ja ise mõistatamine peaks olema andnud juurde kindlust ja kogemusi juhendaja abita rakendusele läheneda. Praktilise osa läbijal peaks olema tekkinud ideid, kuidas näitena toodud süsteeme täiendada, parandada või ise täiesti uusi infosüsteeme luua.

Summary

Bachelor paper “Study Guide for BC4J Java Business Components based on Oracle9*i* JDeveloper” is about today’s most efficient technology creating applications based on business components. This paper was created because no tutorials have been written in estonian before.

This guide is divided into two main parts: part 1 and part 2. Part 1 describes theory about needs, installation, business components, features, 3-tier application development and is divided into four chapters. Part 2 consist of two practice example applications. First is created using full benefit of different wizards that help users to develop their applications from the beginning to the end. In first practical lesson users will learn to create tables into the Oracle’s database schema and then create a connection from the JDeveloper. Wizards generate necessary objects into project automatically. The second example needs some more hand written elements and is created using less wizards. Important is developing applications using component’s properties, different UI components and layouts. Most of the guide is illustrated with figures and screenshots.

Structure of the practice part of this bachelor paper based on “Oracle9*i* Handbook” which is a good example how to do a good tutorial – explainable and practical.

Users must have some experience in Java, in JSP and have some knowlegde about relational databases.

Kasutatud kirjandus

1. Oracle9i JDeveloper Online Demonstrations http://otn.oracle.com/products/jdev/viewlets/viewlet-archive0903.html [25.04.04].
2. "Welcome to the The BC4J Toy Store Demo" - Steve Muench, Oracle Corporation http://otn.oracle.com/sample_code/products/jdev/bc4jtoystore/index.html [25.04.04].
3. Oracle9i JDeveloper [Reviewers Guide – Oktoober 2002].
4. Installing Oracle9*i* JDeveloper Version 9.0.4 http://otn.oracle.com/products/jdev/htdocs/904/install.html#system [25.04.04].
5. Expert Oracle9*i* Database Administration [Sam. R. Alapati – 2003].
6. Simplifying J2EE and EJB Development with BC4J (Steve Muench, BC4J Development Team – Aprill 2002) http://www.cse.buffalo.edu/~qiwang/cse562/J2EE_EJB.htm [25.04.04].
7. What You Need To Know Before Building Applications with JDeveloper 9*i* [Paul Dorsey, Dulcian, Inc.]
8. Optimizing Development Productivity Using UML in Oracle9*i* JDeveloper [An Oracle White Paper – Detsember 2001]
9. Oracle9*i* JDeveloper Handbook [Peter Koletzke, Avrom Faderman, Paul Dorsey - 2002]

Lisa 1 – tabelite loomise skript

```
SET ECHO OFF

drop table ostusisu;
drop table raamat;
drop table tarnija;
drop table ost;

drop sequence ostu_kood_seq;
drop sequence raamatu_kood_seq;
drop type T_AADRESS;


prompt
prompt
prompt OST ja RAAMAT tabeli koodi sequenceri loomine .................
prompt
prompt

create sequence ostu_kood_seq start with 100001;
create sequence raamatu_kood_seq start with 1001;


prompt
prompt
prompt T_AADRESSi tyybi loomine .................
prompt
prompt

create type T_AADRESS as object(
  tn_maja varchar2(30),
  linn varchar2(30),
  indeks varchar2(5)
);
/


prompt
prompt
prompt Tabeli TARNIJA loomine ...................
prompt
prompt



create table tarnija(
  regkood number(10),
  nimi varchar2(30),
  aadress T_AADRESS,
  telefon varchar2(15),
  email varchar2(40),
  constraint tarnija_pk primary key (regkood)
);


prompt
prompt
prompt Tabeli RAAMAT loomine ...................
prompt
prompt


create table raamat(
  kood number,
  pealkiri varchar2(120),
  autor_enimi varchar2(30),
  autor_pnimi varchar2(30),
  i_aasta varchar2(4),
  tyyp varchar2(2),
  keel varchar2(30),
  laoseis number,
```

```
  hind number(8,2),
  tarnija_regkood number(10),
  constraint raamat_pk primary key (kood),
  constraint raamat_kuulub_tarnijale foreign key (tarnija_regkood) references tarnija
(regkood)
);


prompt
prompt
prompt Tabeli OST loomine ...................
prompt
prompt

rem+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
rem                                                                          +
rem    kuupaev tuleb sisestada sellisel kujul:  'pp-KUU-aaaa'                +
rem                                                                          +
rem+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


create table ost(
  kood number,
  aeg date,
  constraint ost_pk primary key (kood)
);

prompt
prompt
prompt Tabeli OSTUSISU loomine ...................
prompt
prompt


create table ostusisu(
  kogus number,
  ostu_kood number,
  raamatu_kood number,
  constraint ostu_raamatu_pk primary key (ostu_kood, raamatu_kood),
  constraint ostu_kuuluv_raamatukood foreign key (raamatu_kood) references raamat (kood),
  constraint ostu_kuuluv_ostukood foreign key (ostu_kood) references ost (kood)
);

COMMIT;

SET ECHO ON
```

Lisa 2 – tarnija andmete sisestuse skript

```
SET ECHO OFF

prompt
prompt
prompt Andmesisestus TARNIJA ...................
prompt
prompt

insert into tarnija (regkood, nimi, aadress, telefon, email) values
  ('1112223333', 'Peeter Muna', T_AADRESS('Toome 14', 'Tartu', '11564'), '56488656',
'sendmail@com.com');
insert into tarnija (regkood, nimi, aadress, telefon, email) values
  ('1345987237', 'Armant Ploom', T_AADRESS('Kuuse 12-12', 'Tapa', '15554'), '55892656',
'sebra@ram.com');
insert into tarnija (regkood, nimi, aadress, telefon, email) values
  ('3498592938', 'Ambrant', T_AADRESS('Kaasiku 45', 'Pärnu', '11111'), '56689456',
'potter@hot.ee');
insert into tarnija (regkood, nimi, aadress, telefon, email) values
  ('9283748535', 'Koorik', T_AADRESS('Elamu 23-1', 'Tallinn', '12364'), '5655556',
'koorik@aol.com');
insert into tarnija (regkood, nimi, aadress, telefon, email) values
  ('0943858356', 'Tartu Raamat', T_AADRESS('Emajõe 23-15', 'Tartu', '12222'), '56499885',
'priidik@msn.com');
insert into tarnija (regkood, nimi, aadress, telefon, email) values
  ('2349583905', 'Peetrike', T_AADRESS('Kase 14', 'Tallinn', '13978'), '55123556',
'peetrike@peetrike.ee');
insert into tarnija (regkood, nimi, aadress, telefon, email) values
  ('1245684596', 'Paistik', T_AADRESS('Tartu mnt. 16', 'Paide', '12264'), '53568974',
'karbik@kiri.ee');
insert into tarnija (regkood, nimi, aadress, telefon, email) values
  ('1134589496', 'Munamägi', T_AADRESS('Koralli 12', 'Tartu', '15667'), '6457852',
'saabas@com.com');
insert into tarnija (regkood, nimi, aadress, telefon, email) values
  ('3456920984', 'Kuuno Majakas', T_AADRESS('Vabaduse pst. 345', 'Tallinn', '11555'),
'56456875', 'raamat@lasn.ee');
insert into tarnija (regkood, nimi, aadress, telefon, email) values
  ('3593489238', 'Kapsaraud', T_AADRESS('Vase 6', 'Tallinn', '12456'), '56564556',
'kapsaraud@raud.ee');

COMMIT;

SET ECHO ON
```

Lisa 3 – raamatu andmete sisestuse skript

```
SET ECHO OFF

prompt
prompt
prompt Andmesisestus RAAMAT ...................
prompt
prompt

insert into raamat (kood, pealkiri, autor_enimi, autor_pnimi, i_aasta, tyyp, keel, laoseis,
hind, tarnija_regkood) values
  (raamatu_kood_seq.nextval, 'Kadri kasuema', 'Silvia', 'Rannamaa', '1976', 'KK', 'eesti', 12,
13.20, '3593489238');
insert into raamat (kood, pealkiri, autor_enimi, autor_pnimi, i_aasta, tyyp, keel, laoseis,
hind, tarnija_regkood) values
  (raamatu_kood_seq.nextval, 'Kadri', 'Silvia', 'Rannamaa', '1979', 'KK', 'eesti', 10, 11.00,
'3593489238');
insert into raamat (kood, pealkiri, autor_enimi, autor_pnimi, i_aasta, tyyp, keel, laoseis,
hind, tarnija_regkood) values
  (raamatu_kood_seq.nextval, 'Lennuk taevas', 'Sasa', 'Popov', '1999', 'PK', 'vene', 15, 9.30,
'1134589496');
insert into raamat (kood, pealkiri, autor_enimi, autor_pnimi, i_aasta, tyyp, keel, laoseis,
hind, tarnija_regkood) values
  (raamatu_kood_seq.nextval, 'Laste loomaaed', 'Steven', 'Rource', '2001', 'KK', 'eesti', 56,
125.00, '3456920984');
insert into raamat (kood, pealkiri, autor_enimi, autor_pnimi, i_aasta, tyyp, keel, laoseis,
hind, tarnija_regkood) values
  (raamatu_kood_seq.nextval, 'Nimed marmortahvlil', 'Karl', 'Ristikivi', '2002', 'KK',
'eesti', 112, 222.00, '2349583905');
insert into raamat (kood, pealkiri, autor_enimi, autor_pnimi, i_aasta, tyyp, keel, laoseis,
hind, tarnija_regkood) values
  (raamatu_kood_seq.nextval, 'Muinasjutte', '', 'Vennad Grimmid', '1980', 'KK', 'eesti', 12,
45.20, '2349583905');
insert into raamat (kood, pealkiri, autor_enimi, autor_pnimi, i_aasta, tyyp, keel, laoseis,
hind, tarnija_regkood) values
  (raamatu_kood_seq.nextval, 'Sentences', 'Phil', 'Dorsey', '2001', 'PK', 'inglise', 56,
65.55, '0943858356');
insert into raamat (kood, pealkiri, autor_enimi, autor_pnimi, i_aasta, tyyp, keel, laoseis,
hind, tarnija_regkood) values
  (raamatu_kood_seq.nextval, 'Koolimaja', 'Arnold', 'Tramm', '1976', 'PK', 'eesti', 3, 11.20,
'9283748535');
insert into raamat (kood, pealkiri, autor_enimi, autor_pnimi, i_aasta, tyyp, keel, laoseis,
hind, tarnija_regkood) values
  (raamatu_kood_seq.nextval, 'Mingid luuletused', 'Juhan', 'Liiv', '1999', 'KK', 'eesti', 23,
179.60, '3498592938');
insert into raamat (kood, pealkiri, autor_enimi, autor_pnimi, i_aasta, tyyp, keel, laoseis,
hind, tarnija_regkood) values
  (raamatu_kood_seq.nextval, 'Simon ja tammed', 'Oli', 'Keegi', '2003', 'KK', 'eesti', 10,
159.00, '1345987237');
insert into raamat (kood, pealkiri, autor_enimi, autor_pnimi, i_aasta, tyyp, keel, laoseis,
hind, tarnija_regkood) values
  (raamatu_kood_seq.nextval, 'Guten Abend!', 'Peter', 'Stahl', '1989', 'PK', 'saksa', 42,
133.20, '1112223333');
insert into raamat (kood, pealkiri, autor_enimi, autor_pnimi, i_aasta, tyyp, keel, laoseis,
hind, tarnija_regkood) values
  (raamatu_kood_seq.nextval, 'Naksitrallid', 'Eno', 'Raud', '1983', 'KK', 'eesti', 5, 89.20,
'1112223333');

COMMIT;

SET ECHO ON
```

Lisa 4 – ostu andmete sisestuse skript

```
SET ECHO OFF

prompt
prompt
prompt Andmesisestus OST ...................
prompt
prompt

insert into ost (kood, aeg) values
  (ostu_kood_seq.nextval, '27-JUL-99');
insert into ost (kood, aeg) values
  (ostu_kood_seq.nextval, '16-JUL-99');
insert into ost (kood, aeg) values
  (ostu_kood_seq.nextval, '27-JUL-99');
insert into ost (kood, aeg) values
  (ostu_kood_seq.nextval, '11-NOV-99');
insert into ost (kood, aeg) values
  (ostu_kood_seq.nextval, '12-JUL-99');
insert into ost (kood, aeg) values
  (ostu_kood_seq.nextval, '26-JUL-00');
insert into ost (kood, aeg) values
  (ostu_kood_seq.nextval, '13-JUL-00');
insert into ost (kood, aeg) values
  (ostu_kood_seq.nextval, '11-MAY-01');
insert into ost (kood, aeg) values
  (ostu_kood_seq.nextval, '15-APR-02');
insert into ost (kood, aeg) values
  (ostu_kood_seq.nextval, '17-AUG-03');
insert into ost (kood, aeg) values
  (ostu_kood_seq.nextval, sysdate);

COMMIT;

SET ECHO ON
```

Lisa 5 – ostusisu andmete sisestuse skript

```
SET ECHO OFF

prompt
prompt
prompt Andmesisestus OST ...................
prompt
prompt

insert into ostusisu (kogus, ostu_kood, raamatu_kood) values
  (1, 100001, 1002);
insert into ostusisu (kogus, ostu_kood, raamatu_kood) values
  (2, 100002, 1003);
insert into ostusisu (kogus, ostu_kood, raamatu_kood) values
  (1, 100003, 1001);
insert into ostusisu (kogus, ostu_kood, raamatu_kood) values
  (4, 100004, 1006);
insert into ostusisu (kogus, ostu_kood, raamatu_kood) values
  (7, 100005, 1004);
insert into ostusisu (kogus, ostu_kood, raamatu_kood) values
  (6, 100006, 1005);
insert into ostusisu (kogus, ostu_kood, raamatu_kood) values
  (8, 100007, 1007);


COMMIT;

SET ECHO ON
```