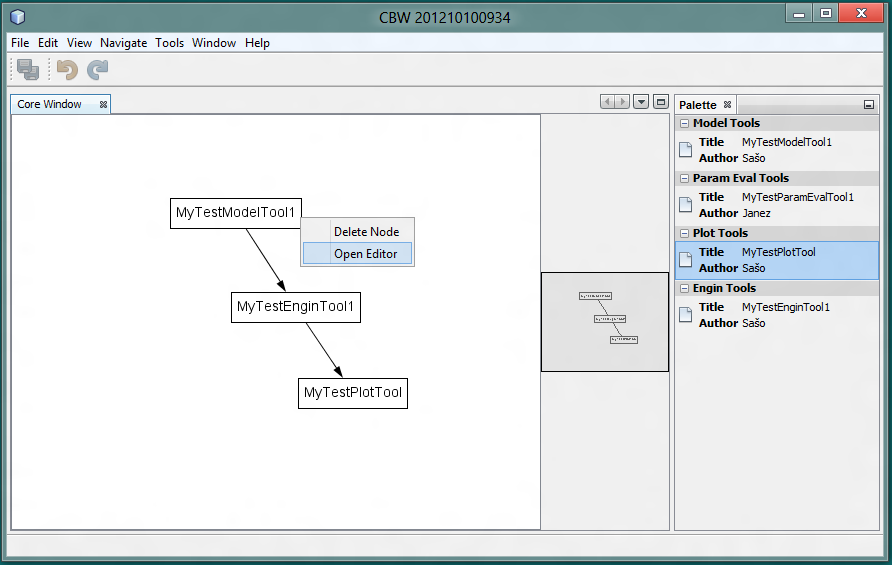
# COMPUTATIONAL BIOLOGY WORKSPACE (hitri vodič okolja in kreiranja novega modula)

## Okolje

Aplikacija uporablja NetBeans Platform, za njeno ogrodje. Za najboljšo kompatibilnost razvijanje je potrebno, da imate nameščeno različico **NetBeans IDE 7.2.1**. Aplikacija je tudi spisana za delovanje na Microsoft Windows operacijskem sistemu in zna imeti težave na drugih platformah (zaradi JavaFX knjižnjic, ki se nahajajo v ./release/modules/bin).

## Delovanje aplikacije

Osnovna dva dela aplikacije sta »Palette« in »Core window« (delovna površina), kamor vlečemo komponente s palete. Komponente oz. orodja, postanejo vozlišča (nodes), katere lahko povežemo med seboj. Komponente lahko povežemo, tako da držimo tipko kontrol in jih z miško povežemo.



Čeprav aplikacija dopušča povezovanje vseh orodji med seboj je potrebno biti pazljiv, saj so orodja odvisna od predhonikov (predhodnik plot orodja mora biti engine). Z desnim klikom na vozlišče se nam prikaže izpustni meni, ki nam ponudi brisanje vozlišča ali možnost dodatnega urejanja orodja.

Primer pravilnega povezovanja je viden na zgornji sliki. PlotTool dobi podatke z EngineTool-a, ta pa z ModelTool-a.

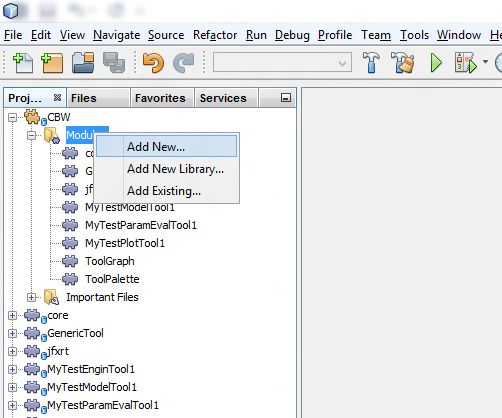
## Struktura palikacije

Aplikacija je razdeljena glavne module in razširitve. Glavni moduli so:

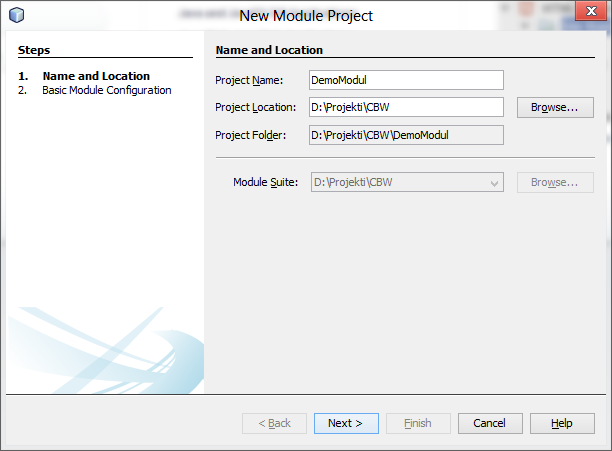
* **Core**: glavno okno, ki se odpre ob zagonu aplikacije in je ogrodje za ToolGraph in ToolPalette module.
* **ToolGraph**: je implementacija GraphScene razreda in omogoča dodajanje vozlišč in povezovanje le teh. Tako imenovana delovna površina.
* **ToolPalette**: je posebno okno v NetBeans platformi in se veže na obstoječe okno. V tem primeru se veže na delovno površino in vsebuje vse module, ki so najavljeni NetBeans Lookup servisu in so otroci razreda GenericTool.
* **GenericTool**: je paket za vse razširitvene module. Glavni razred je AbstractGenericTool, katerega dedujejo nato vsi ostali abstraktni razredi: AbstractEngineTool, AbstractPlotTool, AbstractModelTool in AbstractParamEvalTool (ki deduje AbstractModelTool) . Modul ima še razred ToolTopComponent, katerega mora uporabiti vsak razširitveni modul.
* **jfxt:** knjižnica JavaFX, za risanje grafov

Ostali moduli so razširitveni demo moduli, ki nakazujejo delovanje in uporabo.

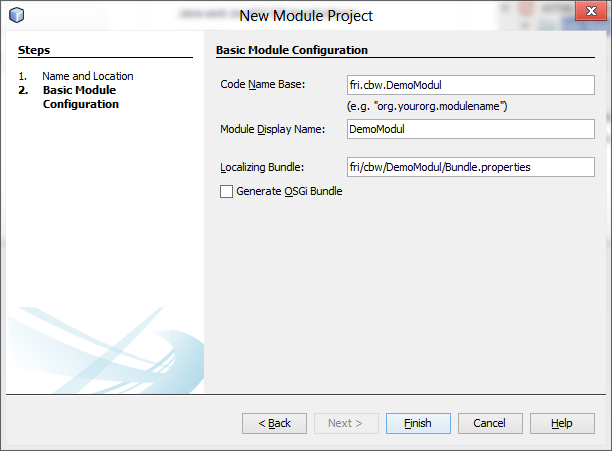
## Pisanje novega modula

Začnemo tako, da s čarovnikom ustvarimo nov NetBeans module. 

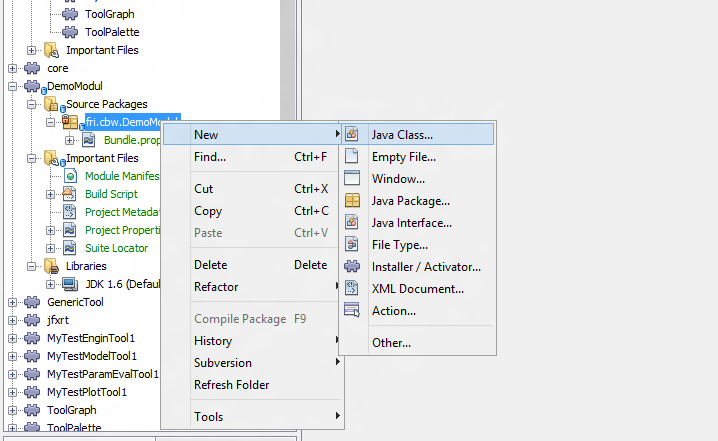
Izberemo ime modula in se prepričamo, da je lokacija projekta znotraj glavnega projekta,



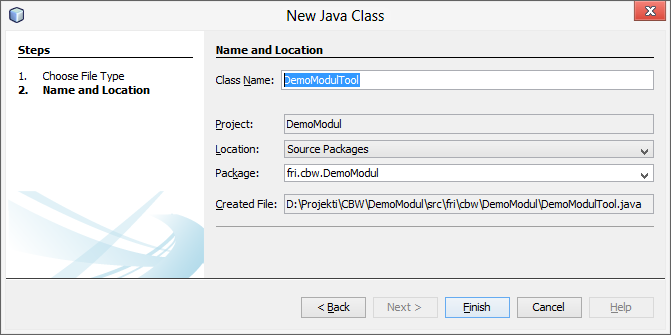
izberemo ime paketa in tako zaključimo s čarovnikom.



Zdaj ko imamo nov modul, lahko začnemo s programiranjem našega orodja. Najprej dodamo nov razred, ki bo dedovalo enega izmed abstraktnih razredov. S tem se bo prikazal tudi na ToolPalette in ga bo moč prenesti na ToolGraph.



V tem primeru bomo implementirali orodje za modeliranje,



zato bo razred dedoval AbstractModelTool.

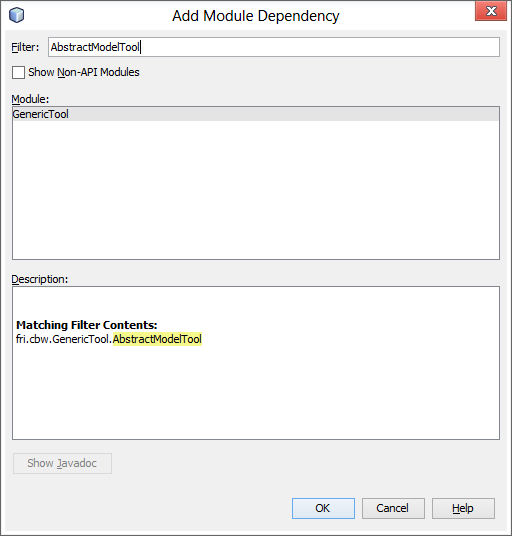
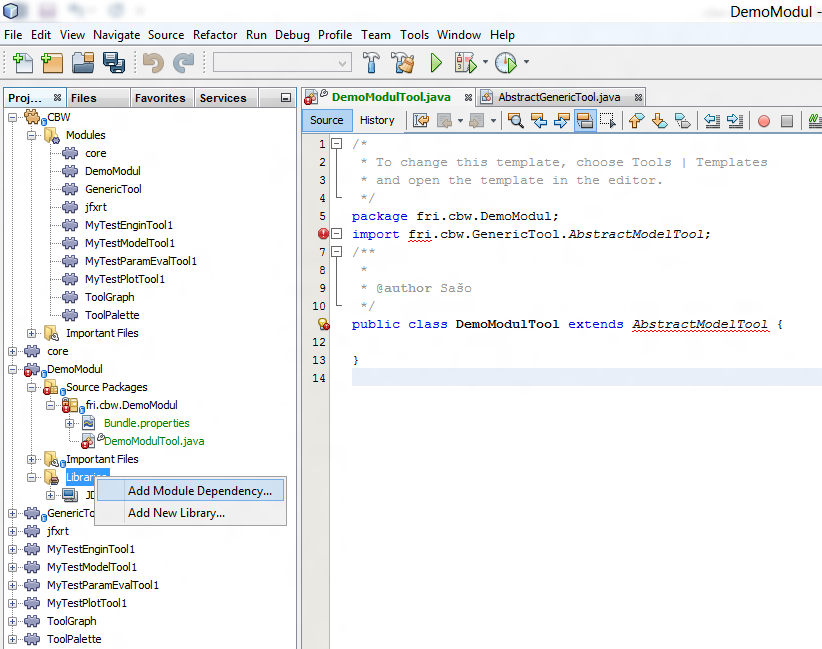
package fri.cbw.DemoModul;

import fri.cbw.GenericTool.AbstractModelTool;

public class DemoModulTool extends AbstractModelTool {

}

Tu že vidimo prve probleme modularnega razvijanja, saj se moduli med seboj ne vidijo dokler ne določimo odvisnosti med njimi. V NetBeans-ih to preprosto naredimo preko iskalnika »Add Module Dependency…«

Ko implementiramo še abstraktne metode nam ostane samo še da ga najavimo NetBenas service lookup-u. Modul ToolPalette namreč naredi seznam s pomočjo Lookup API, ki pa preišče vse module ki imajo najavljene servise določenega razreda.

package fri.cbw.DemoModul;

import fri.cbw.GenericTool.AbstractModelTool;

@ServiceProvider(service = AbstractModelTool.class)

public class DemoModulTool extends AbstractModelTool {

@Override

public String getName() {

return "Demo Model Tool";

}

@Override

public String getAuthor() {

return "Sašo";

}

@Override

public String getTopComponentName() {

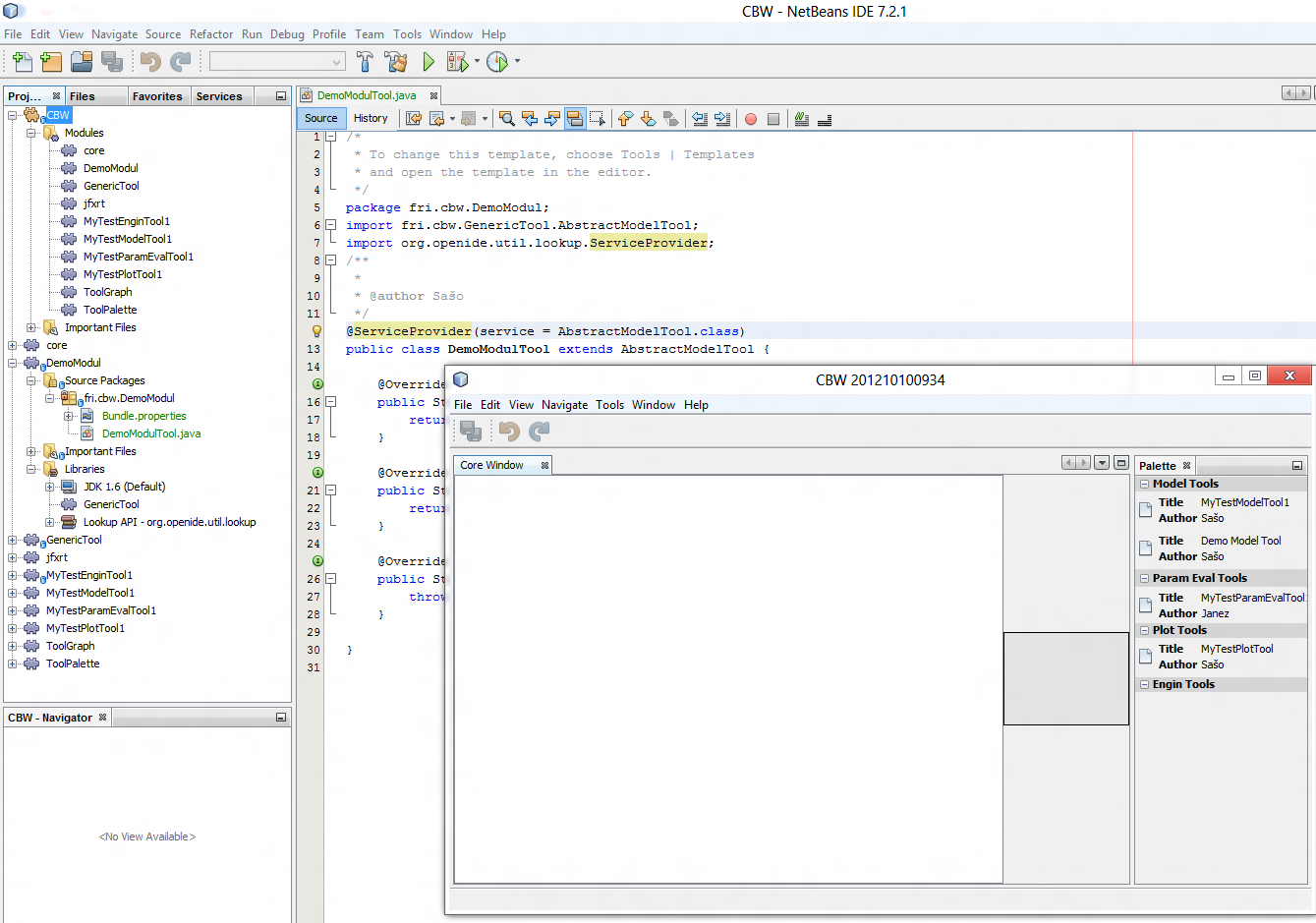
//Metodo bomo implementirali šele ko bomo naredili GUI za modul

throw new UnsupportedOperationException("Not supported yet.");

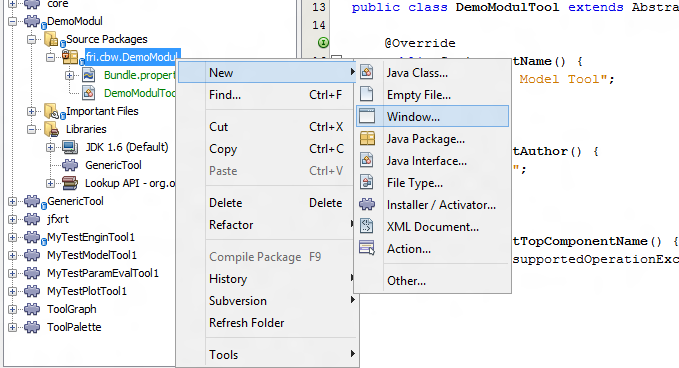
}

}

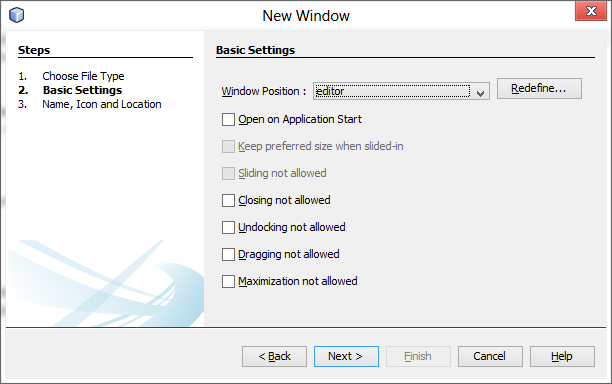
Registracijo naredimo na 2 načina. Lahko naredimo datoteko z imenom servisa v paktu META-INF.services (kot je to narejeno v MyTestModelTool1) ali pa preko notacij (kot je to narejeno v MyTestEnginTool1). V tem primeru ga bomo registrirali preko notacije in to z @ServiceProvider(service = AbstractModelTool.class), ki pa bo javljala napako dokler modulu ne dodamo odvisnosti za Lookup API. Zdaj lahko že zaženemo aplikacijo in naš DemoModul je že na paleti.



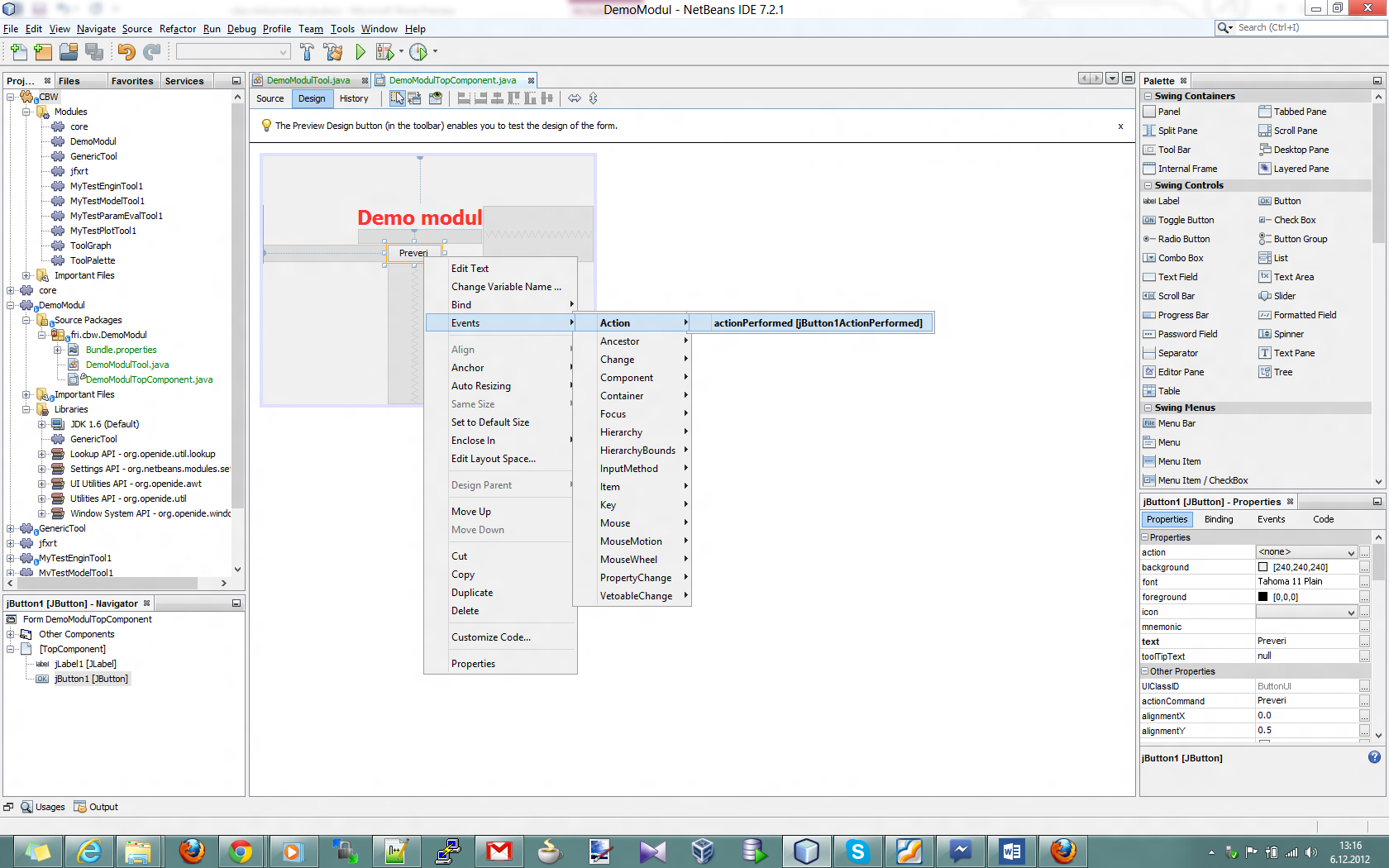
Zdaj pa orodje potrebuje še grafični urejevalnik. NetBeans nam tu zopet olajša zadevo s čarovnikom »New Window«.



V tem primeru želimo da se odpre na poziciji »Editor« (osrednji del, o razporeditvi si lahko preberete več na <http://ui.netbeans.org/docs/ui/ws/ws_spec-netbeans_ide.html>) in ne želimo da se odpre ko se applikacija začne (saj ga odpremo šele ko želimo urejati orodje). Na naslednjem koraku dodamo še predpono in s čarovnikom smo zaključili. NetBeans nam zdaj odpre Swing urejevalnik preko katerega lahko z »drag and drop« komponentami ustvarimo naš vmesnik. Če smo pozorni nam je NetBeans čarovnik dodal tudi vse potrebne odvisnosti do knjižnjižnjic, da se nam ni potrebno s tem ukvarjati (npr. Window System API).



Preden se lotimo izdelavo vmesnika je dobro, da ga najprej naredimo kompatibilnega z CBW okoljem. Naš DemoModulTopComponent razred trenutno deduje TopComponent. To je potrebo spremeniti v ToolTopComponent, saj nam ta omogoča povezavo do delovne površine in tako povezav med tem in ostalimi moduli. Ko to naredimo pa popravimo še naš DemoModulTool razred in zdaj implementiramo še metodo getTopComponentName(). Dodamo ji samo return izjavo z vrednostjo »DemoModulTopComponent«. To vrednost najdemo v našem TopComponent razredu pod notacijo preferredID. Spremenimo še TopComponent anotacijo TopComponent.PERSISTENCE\_ALWAYS v TopComponent.PERSISTENCE\_NEVER, da se okno ponastavi ob izhodu aplikacije in zdaj lahko zaženemo aplikacijo. Dodamo naše orodje na delovno površino in z desnim klikom odpremo urejevalnik, ki je trenutno še prazen.



Za naš Demo primer sem na urejevalni dodal labelo in gumb, ki bo preveril ali je predhodno orodje tudi tipa AbstractModelTool.

Koda za gumb je sledeča:

(ne pozabite dodati odvisnotis: Visual Library API, ToolGraph, Dialogs API, Utilities API)

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

NotifyDescriptor nd;

try{

ToolWrapper tool = (ToolWrapper)getScene().findObject (getToolNode());

ToolWrapper prevTool = tool.getPrevNode(getScene());

AbstractModelTool gt = (AbstractModelTool) prevTool.getNodeGenericTool();

nd = new NotifyDescriptor.Message("Predhodni modul JE ModelTool");

}catch(ClassCastException e){

Logger.getAnonymousLogger().severe(e.getMessage());

nd = new NotifyDescriptor.Message("Predhodni modul NI ModelTool");

}catch(NullPointerException e){

Logger.getAnonymousLogger().severe(e.getMessage());

nd = new NotifyDescriptor.Message("Predhodni modul ne obstaja");

}finally {

DialogDisplayer.getDefault().notify(nd);

}

}

Funkciji getScene() in getToolNode() se nahajati v ToolTopComponent razredu in sti referenci na delovno površino in vozlišče orodja, ki ga trenutno urejamo. S tem lahko dobimo ToolWrapper, na delovni površini, ki ima metodo getPrevNode(). Ta vrne prvo predhodno orodje (ali null če ne bostaja). V ToolWrapper-ju pa se nahaja tudi samo orodje ki ga dobimo z metodo getNodeGenericTool.

