Appendix C: Modelibra Swing

The objective of this appendix is to show that Modelibra may be used as a model for Java Swing [Swing] applets or applications. There are two Swing projects presented in this appendix. The first project is called `ModelibraRushHour` and it based on the Rush Hour game [ThinkFun]. The second project is a calendar utility called `ModelibraCalbadar`. If you replace the **en** letters in the cal**en**dar word with the **ba** letters you will get the new cal**ba**dar word. This is done to prevent you from considering this utility too seriously. The first project is presented to show how Modelibra is used as a model of a graphical game. The second project is introduced to show the use of Modelibra sessions, actions and undos.

After a hard work of learning how to use Modelibra and Wicket in the development of dynamic web applications, you may reward yourself by playing the Rush Hour game and by creating more challenging parking positions. If you are really into it, you may decide to create some intelligent algorithms for creating and resolving parkings, or you may try to design a software game for one of puzzles at [Puzzles].

ModelibraRushHour

The Rush Hour game has a few data windows and a graphical window for the game board. The game board has six rows and six columns with thirty six cells, where a cell is defined by its row and column. Several cars are positioned on the board horizontally or vertically. The cars are of different car model. They have different colors (in this book they have different shades of gray) and lengths (they cover two or three cells). The exit is on the right in the third row from the top. The goal of the game is to move the red car (the darkest car in this chapter) in the third row from the top out of the playing board. In order to do that you have to move other cars horizontally or vertically to make a free space for the red car to exit the board. You move a car one cell at the time by selecting it and clicking on the first available cell in the direction of your choice. Currently, the game has two areas, one for beginners and the other for intermediate users. New areas may be created by the game software. An area has multiple parkings, where each parking represents a new board with cars with more complex challenge in fun thinking.

The game uses Modelibra for the domain model of the Rush Hour game, ModelibraWicket for web views of the model, and Swing for game windows and the graphical view of the model. The default web application of the game model is used to create new areas and new parkings. The Swing part is used to present areas and area parkings to a user of the game in order to select a parking and play the game graphically. In this version of the game, rectangles with different colors and different lengths are used to represent cars. An advanced reader is invited to introduce car images and sound effects to make the game more appealing to young software developers.

Rush Hour Project

The Eclipse project is called `ModelibraRushHour` and is located in the Modelibra repository at JavaForge in the `trunk/App/ModelibraSwing` directory. The project is created from the `ModelibraWicketSkeleton` project. The domain model of the game is designed in ModelibraModeler. The model XML configuration is generated from ModelibraModeler. Based on the XML configuration the code is generated as usual with Modelibra projects. Thus the game software has standard Modelibra and Wicket packages. In addition, there are new Swing packages created by hand (Figure B.1).

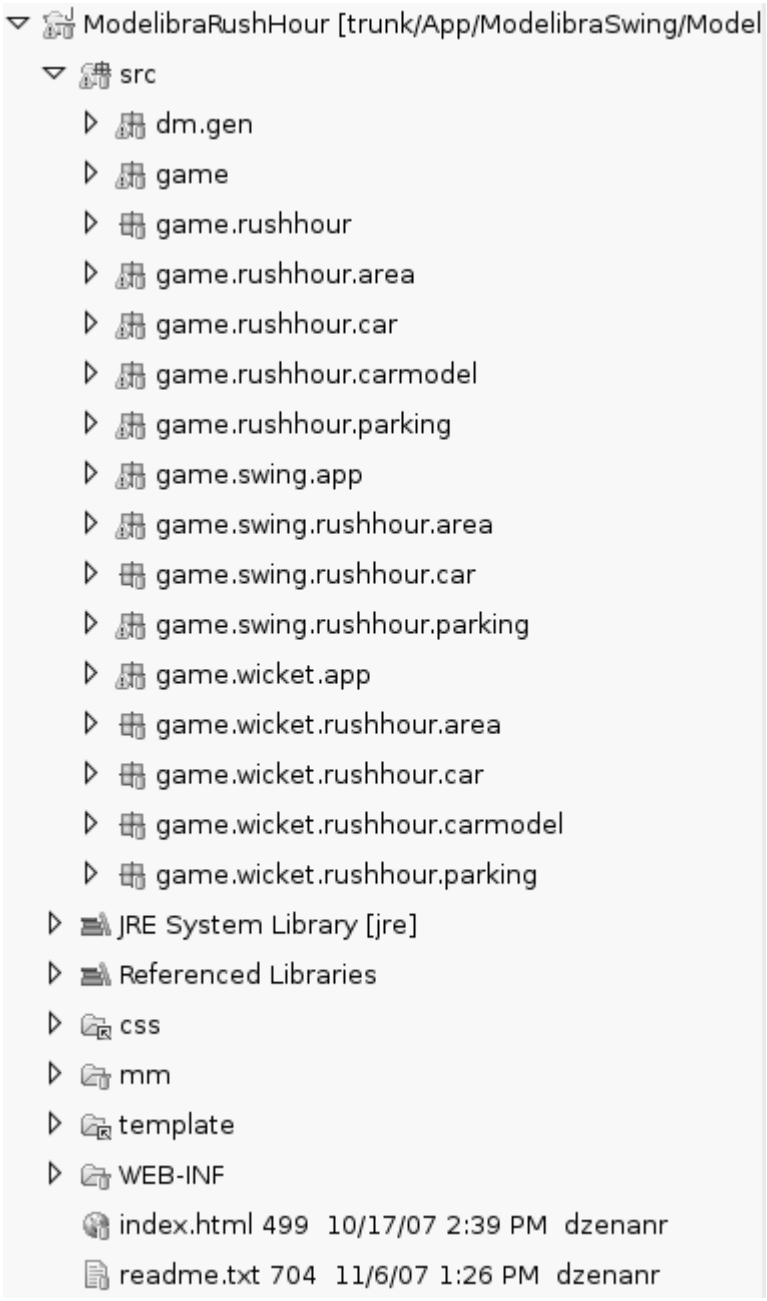

Figure B.1. Eclipse project

Rush Hour Model

The domain is called `Game`. The model is named `RushHour` (Figure B.2.). The model has four concepts: `Area`, `Parking`, `CarModel` and `Car`. Only the relationship between the `Parking` concept and the `Car` concept is internal (a lighter line in Figure B.2). The entry points into the model are the `Area`, `Parking` and `CarModel` concepts. The || sign in the upper left corner of the concept indicates an entry concept. An entry concept has also a darker border.

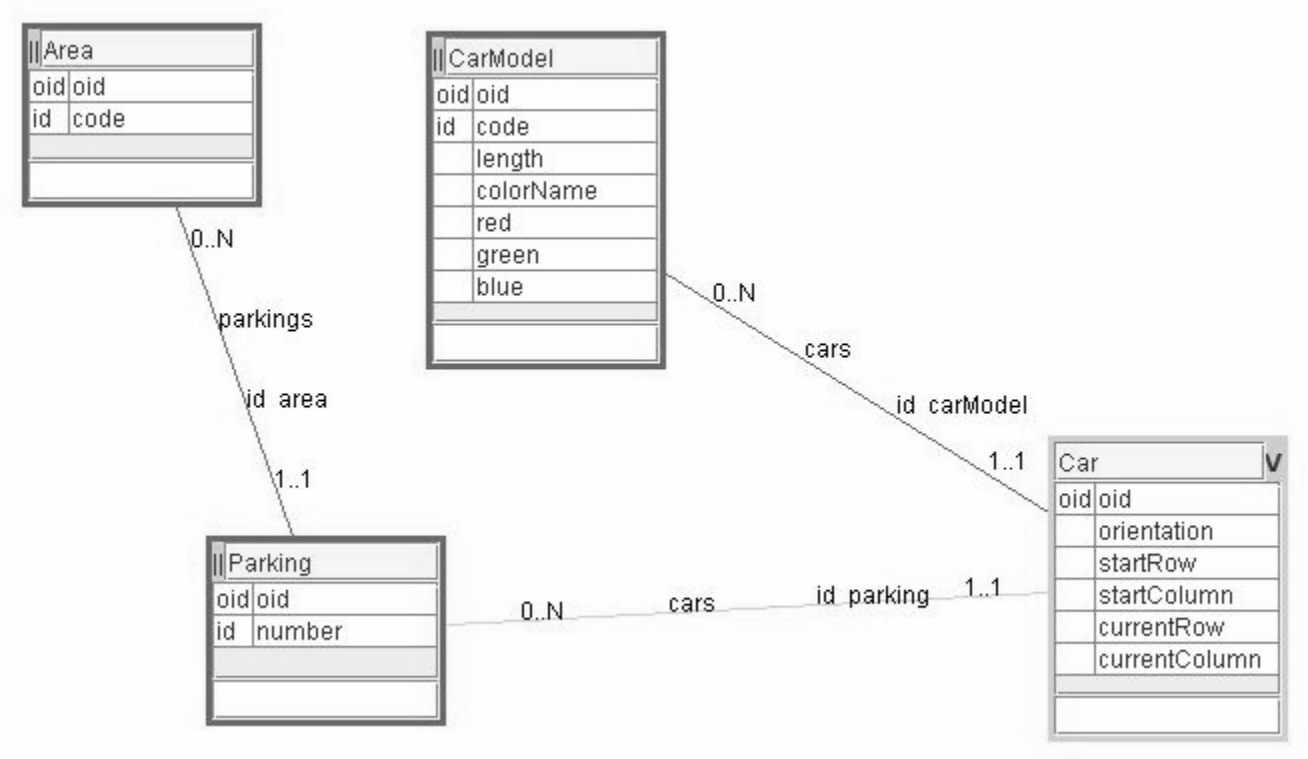

Figure B.2. Rush Hour model

An area has many parkings. A parkings has many cars. A car is defined by its car model. The same car model may be used to define many cars. Since the `Car` concept has two parents, it is an intersection concept (the V sign in the upper right corner of the concept).

Rush Hour Application

The main window of the Swing application (or applet since the game may be run either as an application or an applet) presents a choice of available areas (Figure B.3). The *Game* menu has the *Exit* menu item, while the *Help* menu has the *About...* and *Rules...* items. A click on the *Parkings* button for the selected area displays the new window with the list of available parkings (Figure B.4). The Parking 1 in the Beginner area is a simple car configuration to resolve (Figure B.5). This configuration is defined in a web page (Figure B.6).

Figure B.3. Rush Hour areas

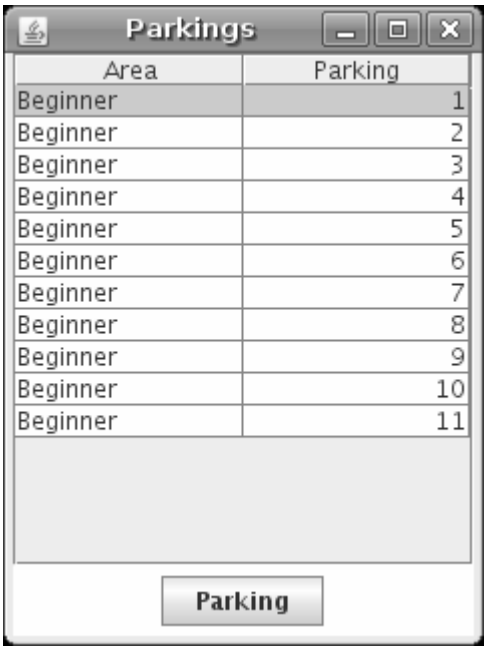

Figure B.4. Rush Hour parkings

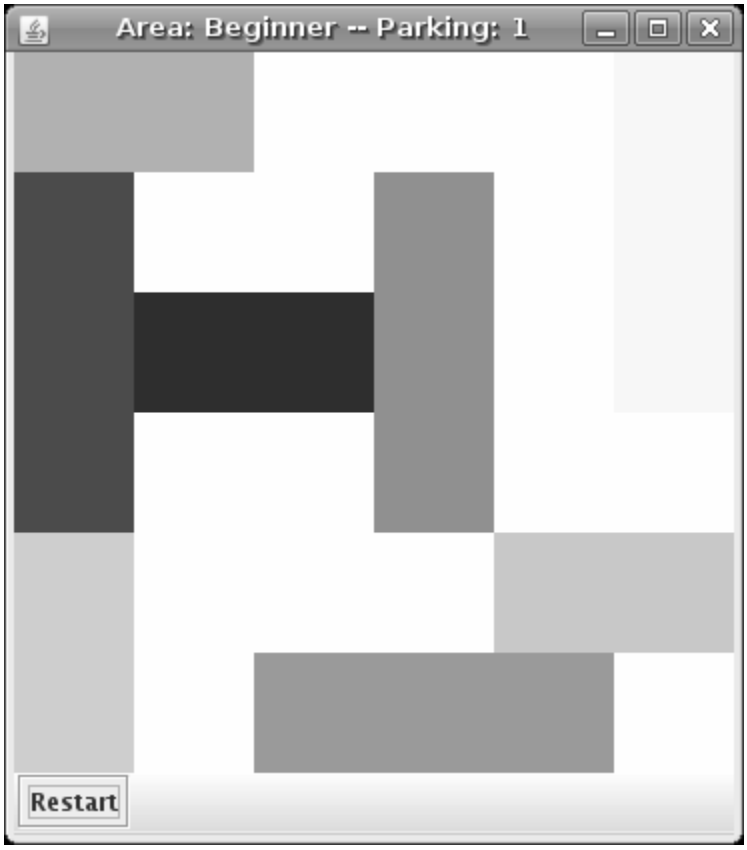

Figure B.5. Beginner area, Parking 1: graphical board

Cars

Number	Code	Length	Color Name	Red	Green	Blue	Orientation	Start Row	Start Column			
1	A	2	green	102	204	153	Horizontal	0	0			
1	B	2	orange	255	204	102	Vertical	4	0			
1	C	2	cyan	0	255	255	Horizontal	4	4			
1	O	3	light yellow	255	255	153	Vertical	0	5			
1	P	3	magenta	153	51	102	Vertical	1	0			
1	Q	3	gray blue	102	153	204	Vertical	1	3			
1	R	3	deep sky blue	0	191	255	Horizontal	5	2			
1	X	2	red	204	0	51	Horizontal	2	1			

<< < 1 > >>

Figure B.6. Beginner area, Parking 1: car configuration

Rush Hour Configuration

The domain configuration is generated as the specific (not reusable) XML configuration, since future changes of the model are not planned.

```
<?xml version="1.0" encoding="UTF-8"?>

<domains>
```

```
<domain oid="1174415547218">
     <code>Game</code>
     <type>Specific</type>

     <models>

          <model oid="1174415568899">
               <code>RushHour</code>
               <author>Dzenan Ridjanovic</author>
               <persistenceType>xml</persistenceType>
               <persistenceRelativePath>
                    data/xml/game/rushhour
               </persistenceRelativePath>
               <defaultLoadSave>true</defaultLoadSave>

               <concepts>

                    <concept oid="1174415643197">
                         <code>Parking</code>
                         <entitiesCode>Parkings</entitiesCode>
                         <entry>true</entry>

                         <properties>
                              <property oid="1174415718207">
                                   <code>areaOid</code>
                                   <propertyClass>java.lang.Long</propertyClass>
                                   <required>true</required>
                                   <reference>true</reference>
                                   <referenceNeighbor>area</referenceNeighbor>

                                   <essential>false</essential>
                              </property>
                              <property oid="1174416298442">
                                   <code>number</code>
                                   <propertyClass>
                                        java.lang.Integer
                                   </propertyClass>
                                   <required>true</required>
                                   <unique>true</unique>

                                   <essential>true</essential>
                              </property>
                         </properties>

                         <neighbors>
                              <neighbor oid="1174415728834">
                                   <code>cars</code>
                                   <destinationConcept>Car</destinationConcept>
                                   <inverseNeighbor>parking</inverseNeighbor>
                                   <internal>true</internal>
                                   <partOfManyToMany>false</partOfManyToMany>
                                   <type>child</type>
                                   <min>0</min>
                                   <max>N</max>
                              </neighbor>
                              <neighbor oid="1174415718207">
                                   <code>area</code>
                                   <destinationConcept>Area</destinationConcept>
                                   <inverseNeighbor>parkings</inverseNeighbor>
                                   <internal>false</internal>
                                   <partOfManyToMany>false</partOfManyToMany>
                                   <type>parent</type>
                                   <min>1</min>
```

```
                    <max>1</max>
                    <unique>true</unique>
               </neighbor>
          </neighbors>

     </concept>

     <concept oid="1174415652949">
          <code>Car</code>
          <entitiesCode>Cars</entitiesCode>

          <properties>
               <property oid="1174415732085">
                    <code>carModelOid</code>

                    <propertyClass>java.lang.Long</propertyClass>
                    <required>true</required>
                    <reference>true</reference>
                    <referenceNeighbor>
                         carModel
                    </referenceNeighbor>

                    <essential>false</essential>
               </property>
               <property oid="1174417282105">
                    <code>orientation</code>
                    <propertyClass>
                         java.lang.String
                    </propertyClass>
                    <maxLength>16</maxLength>
                    <required>true</required>
                    <defaultValue>Horizontal</defaultValue>

                    <essential>true</essential>
               </property>
               <property oid="1174427530720">
                    <code>startRow</code>
                    <propertyClass>
                         java.lang.Integer
                    </propertyClass>
                    <required>true</required>

                    <essential>true</essential>
               </property>
               <property oid="1174427533909">
                    <code>startColumn</code>
                    <propertyClass>
                         java.lang.Integer
                    </propertyClass>
                    <required>true</required>

                    <essential>true</essential>
               </property>
               <property oid="1174427542269">
                    <code>currentRow</code>
                    <propertyClass>
                         java.lang.Integer
                    </propertyClass>

                    <essential>false</essential>
               </property>
               <property oid="1174427547223">
                    <code>currentColumn</code>
                    <propertyClass>
```

```
                    java.lang.Integer
                </propertyClass>

                <essential>false</essential>
            </property>
        </properties>

        <neighbors>
            <neighbor oid="1174415728834">
                <code>parking</code>
                <destinationConcept>
                    Parking
                </destinationConcept>
                <inverseNeighbor>cars</inverseNeighbor>
                <internal>true</internal>
                <partOfManyToMany>false</partOfManyToMany>
                <type>parent</type>
                <min>1</min>
                <max>1</max>
                <unique>true</unique>
            </neighbor>
            <neighbor oid="1174415732085">
                <code>carModel</code>
                <destinationConcept>
                    CarModel
                </destinationConcept>
                <inverseNeighbor>cars</inverseNeighbor>
                <internal>false</internal>
                <partOfManyToMany>false</partOfManyToMany>
                <type>parent</type>
                <min>1</min>
                <max>1</max>
                <unique>true</unique>
            </neighbor>
        </neighbors>

    </concept>

    <concept oid="1174415685656">
        <code>CarModel</code>
        <entitiesCode>CarModels</entitiesCode>
        <entry>true</entry>

        <properties>
            <property oid="1174415881805">
                <code>code</code>
                <propertyClass>
                    java.lang.String
                </propertyClass>
                <maxLength>16</maxLength>
                <required>true</required>
                <unique>true</unique>

                <essential>true</essential>
            </property>
            <property oid="1174415885462">
                <code>length</code>
                <propertyClass>
                    java.lang.Integer
                </propertyClass>
                <required>true</required>

                <essential>true</essential>
            </property>
```

```
                <property oid="1174427457610">
                    <code>colorName</code>
                    <propertyClass>
                        java.lang.String
                    </propertyClass>
                    <maxLength>16</maxLength>

                    <essential>true</essential>
                </property>
                <property oid="1174415916291">
                    <code>red</code>
                    <propertyClass>
                        java.lang.Integer
                    </propertyClass>
                    <required>true</required>

                    <essential>true</essential>
                </property>
                <property oid="1174415918089">
                    <code>green</code>
                    <propertyClass>
                        java.lang.Integer
                    </propertyClass>
                    <required>true</required>

                    <essential>true</essential>
                </property>
                <property oid="1174415920168">
                    <code>blue</code>
                    <propertyClass>
                        java.lang.Integer
                    </propertyClass>
                    <required>true</required>

                    <essential>true</essential>
                </property>
            </properties>

            <neighbors>
                <neighbor oid="1174415732085">
                    <code>cars</code>
                    <destinationConcept>Car</destinationConcept>
                    <inverseNeighbor>carModel</inverseNeighbor>
                    <internal>false</internal>
                    <partOfManyToMany>false</partOfManyToMany>
                    <type>child</type>
                    <min>0</min>
                    <max>N</max>
                </neighbor>
            </neighbors>

        </concept>

        <concept oid="1174415701627">
            <code>Area</code>
            <entitiesCode>Areas</entitiesCode>
            <entry>true</entry>

            <properties>
                <property oid="1174416223456">
                    <code>code</code>
                    <propertyClass>
                        java.lang.String
                    </propertyClass>
```

```
                                        <maxLength>16</maxLength>
                                        <required>true</required>
                                        <unique>true</unique>

                                        <essential>true</essential>
                                   </property>
                              </properties>

                              <neighbors>
                                   <neighbor oid="1174415718207">
                                        <code>parkings</code>
                                        <destinationConcept>
                                             Parking
                                        </destinationConcept>
                                        <inverseNeighbor>area</inverseNeighbor>
                                        <internal>false</internal>
                                        <partOfManyToMany>false</partOfManyToMany>
                                        <type>child</type>
                                        <min>0</min>
                                        <max>N</max>
                                   </neighbor>
                              </neighbors>

                         </concept>

                    </concepts>

               </model>

          </models>

     </domain>

</domains>
```

Rush Hour Classes

The `GameConfig` class obtains the domain configuration from the specific XML configuration of the project for the `Game` domain.

```
package game;

import org.modelibra.config.Config;
import org.modelibra.config.DomainConfig;

public class GameConfig extends Config {

     private DomainConfig domainConfig;

     public GameConfig() {
          super();
          domainConfig = getDomainConfig("Game", "Specific");
     }

     public DomainConfig getDomainConfig() {
          return domainConfig;
     }

}
```

The `GenGame` class constructs the `RushHour` model.

```
package game;

import org.modelibra.Domain;
import org.modelibra.config.DomainConfig;

import game.rushhour.RushHour;

public abstract class GenGame extends Domain {

     private RushHour rushHour;

     public GenGame(DomainConfig domainConfig) {
          super(domainConfig);
          rushHour = new RushHour(this);
     }

     public RushHour getRushHour() {
          return rushHour;
     }

}
```

The `Game` class inherits everything from the `GenGame` class. Here you can add some specific code at the model level.

```
package game;

import org.modelibra.config.DomainConfig;

public class Game extends GenGame {

     public Game(DomainConfig domainConfig) {
          super(domainConfig);
     }

}
```

The `GenRushHour` class constructs model entry concepts.

```
package game.rushhour;

import game.rushhour.area.Areas;
import game.rushhour.carmodel.CarModels;
import game.rushhour.parking.Parkings;

import org.modelibra.IDomain;
import org.modelibra.Model;

public abstract class GenRushHour extends Model {

     private Parkings parkings;

     private CarModels carModels;

     private Areas areas;

     public GenRushHour(IDomain domain) {
          super(domain);
          parkings = new Parkings(this);
          carModels = new CarModels(this);
          areas = new Areas(this);
```

```
    }

    public Parkings getParkings() {
        return parkings;
    }

    public CarModels getCarModels() {
        return carModels;
    }

    public Areas getAreas() {
        return areas;
    }

}
```

The `RushHour` class inherits everything from the `GenRushHour` class. The specific `getCars` method is added to retrieve all cars for all parkings. There is no need to start with the `Area` entry concept, since `Parking` is also the entry concept.

```
package game.rushhour;

import game.rushhour.car.Car;
import game.rushhour.car.Cars;
import game.rushhour.parking.Parking;
import game.rushhour.parking.Parkings;

import org.modelibra.IDomain;

public class RushHour extends GenRushHour {

    public RushHour(IDomain domain) {
        super(domain);
    }

    /* ============================== */
    /* ======= SPECIFIC CODE ======= */
    /* ============================== */

    private Cars cars;

    public Cars getCars() {
        if (isInitialized()) {
            if (cars == null) {
                setInitialized(false);
                Cars allCars = new Cars(this);
                Parkings parkings = getParkings();
                for (Parking parking : parkings) {
                    Cars cars = parking.getCars();
                    for (Car car : cars) {
                        allCars.add(car);
                    }
                }
                cars = allCars;
                setInitialized(true);
            }
        }
        return cars;
    }

}
```

The `RushHourDb` is a new class that will be used in the main Swing window to access the model of the

game. In the class constructor, the game configuration is created first. Then, the domain configuration is obtained from the game configuration. Next, the game domain is constructed based on the domain configuration. The persistent game is created based on the game domain. Finally, The game model is found in the game domain. This game model will be used to get areas in order to display them in the main window of the game application (or applet).

```
package game.rushhour;

import game.Game;
import game.GameConfig;
import game.PersistentGame;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.modelibra.config.DomainConfig;

public class RushHourDb {

     private static Log log = LogFactory.getLog(RushHourDb.class);

     private Game game;

     private PersistentGame persistentGame;

     private RushHour rushHour;

     public RushHourDb() {
          super();
          try {
               GameConfig gameConfig = new GameConfig();
               DomainConfig domainConfig = gameConfig.getDomainConfig();
               game = new Game(domainConfig);
               persistentGame = new PersistentGame(game);
               rushHour = game.getRushHour();
          } catch (Exception e) {
               log.error("Error in RushHourDb.constructor: " + e.getMessage());
          }
     }

     public Game getGame() {
          return game;
     }

     public RushHour getRushHour() {
          return rushHour;
     }

     public void close() {
          if (persistentGame != null) {
               persistentGame.close();
          }
     }

}
```

The game application or applet is run by using the `Start` class from the `game.swing.app` package. In the Eclipse project this class is selected and in the pop-up menu the *Run As* menu item is chosen to decide if the application or the applet will be executed.

```
package game.swing.app;

import game.swing.rushhour.area.AreasWindow;
```

```
import javax.swing.JApplet;

public class Start extends JApplet {

     AreasWindow areasWindow;

     public void init() {
          areasWindow = new AreasWindow();
     }

     public void start() {
          areasWindow.setTitle("Modelibra Rush Hour");
          areasWindow.setVisible(true);
     }

     public void stop() {
          areasWindow.setVisible(false);
     }

     public void destroy() {
          areasWindow.dispose();
     }

     public static void main(String[] args) {
          Start app = new Start();
          app.init();
          app.start();
     }

}
```

In both cases the `init` method will be called to construct an object of the `AreasWindow` class from the `game.swing.rushhour.area` package. In the default constructor of this class an object of the `RushHourDb` class will be created and the `areas` entities will be obtained from the model of the Rush Hour database.

```
package game.swing.rushhour.area;

import game.rushhour.RushHourDb;
import game.rushhour.area.Area;
import game.rushhour.area.Areas;

...

public class AreasWindow extends JFrame implements ListSelectionListener {

     private RushHourDb rushHourDb;

     private Areas areas;

     private Area currentArea;

     ...

     public AreasWindow() {
          super();

          rushHourDb = new RushHourDb();
          areas = rushHourDb.getRushHour().getAreas();
          areaTableModel = new AreaTableModel(areas);

          ...
```

```
    }

    public RushHourDb getRushHourDb() {
        return rushHourDb;
    }

    ...

}
```

The Swing code is not explained here, since this is not an objective of the book. A reader is invited to learn about Swing in one of free Java Swing books at [Books].

ModelibraCalbadar

The Modelibra Calendar utility has the main window that displays the current month with days (dates) and the current date for the current year. By clicking on a date you may add a note. You may create or access a calendar for a year by changing the content of the year field. You may also save a calendar in an XML data file anywhere on a disk.

The utility uses Modelibra for the domain model of a calendar, ModelibraWicket for web views of the model, and Swing for windows. The default web application of the calendar model may be used to navigate through a large amount of data. An advanced reader is invited to introduce new calendar features to make the utility more appealing to software developers.

Calendar Project

The Eclipse project is called `ModelibraCalbadar` and is located in the Modelibra repository at JavaForge in the `trunk/App/ModelibraSwing` directory. The project is created from the `ModelibraWicketSkeleton` project. The domain model of the calendar utility is designed in ModelibraModeler. The reusable XML configuration is generated from ModelibraModeler. Based on the reusable XML configuration the code is generated as usual with Modelibra projects. Thus the utility software has standard Modelibra and Wicket packages. In addition, there are new Swing packages created by hand.They all start with the `util.swing` prefix.

By default the calendar utility is shown in English. However, there are also French and Bosnian versions. By changing the `lang` property, in the `Start.properties` file of the `util.swing.app` package, from `en` to `fr` or `ba`, and by executing the application, you get the new international version.

Calendar Model

The domain is called `Util`. The model is named `Calbadar` (Figure B.7.). The model is hierarchical and has five concepts: `Calendar`, `Year`, `Month`, `Day` and `Note`. All relationships are internal. The entry point into the model is the `Calendar` concept. Thus, all data will be saved in one file called `calendar.xml`.

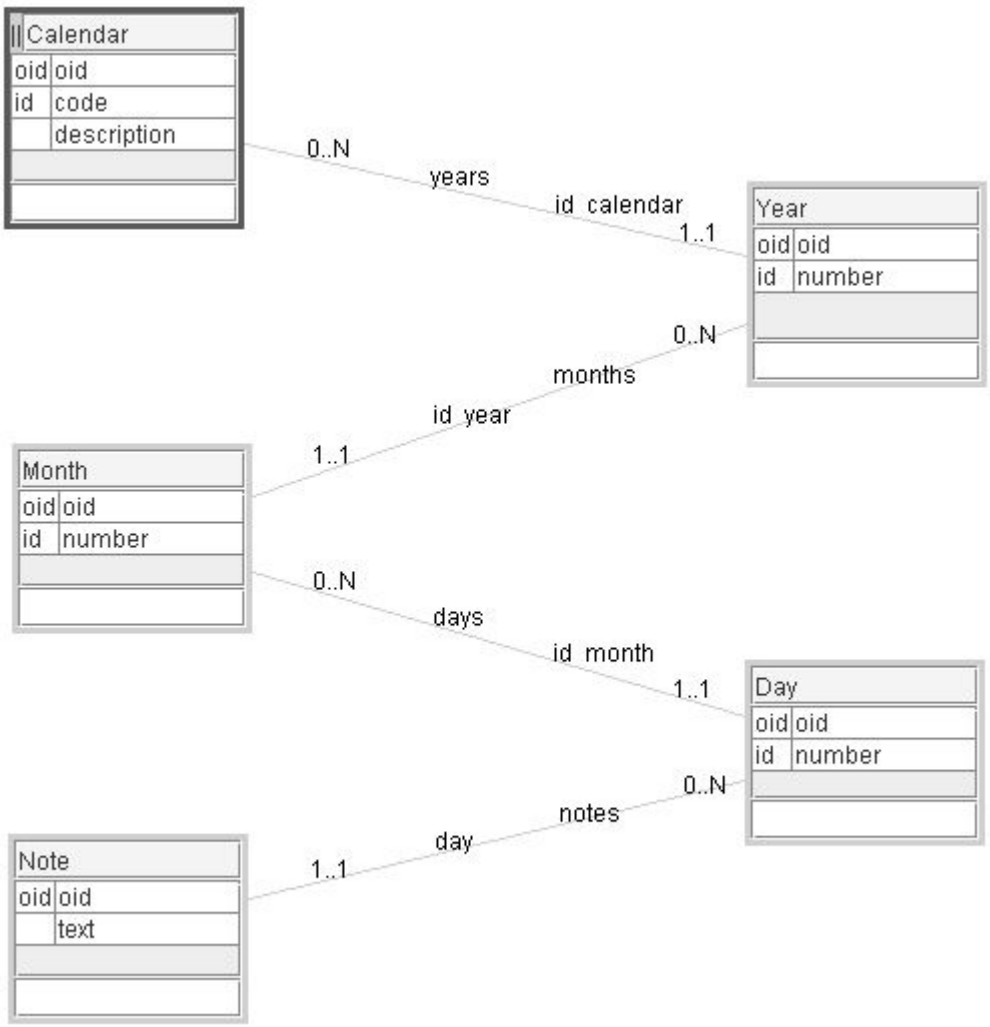

Figure B.7. Calbadar model

A calendar has many years. A year has 12 months. A month has at most 31 days. A day may have several notes.

Calendar Application

The main window of the Swing application (or applet since the utility may be executed either as an application or an applet) presents a calendar for the current month with the current day (date) indicated by a darker background (Figure B.8). To change a year enter a four digit number, then press the keyboard *ENTER* key. If the entered year is not in the model, it will be created, together with its months and days. In this way you may create quite a lot of data in the XML data file and test the performance of Modelibra with respect to the XML persistency. To change a month and display it, enter a month number from 1 to 12, then press the *ENTER* key. You can also display the previous or the next month by clicking on the corresponding button. If you click on a day cell you may add or edit a note. You may view monthly notes and select a subset of them (*View* menu). By default, the data are saved in the `calendar.xml` file that is located within the project in the data directory defined by the `persistenceRelativePath` element in the reusable XML configuration. However, you may select a different directory where the data file will be saved and later on opened (*Calendar* menu). Finally any action can be undone (*Edit* menu, *Undo* menu item).

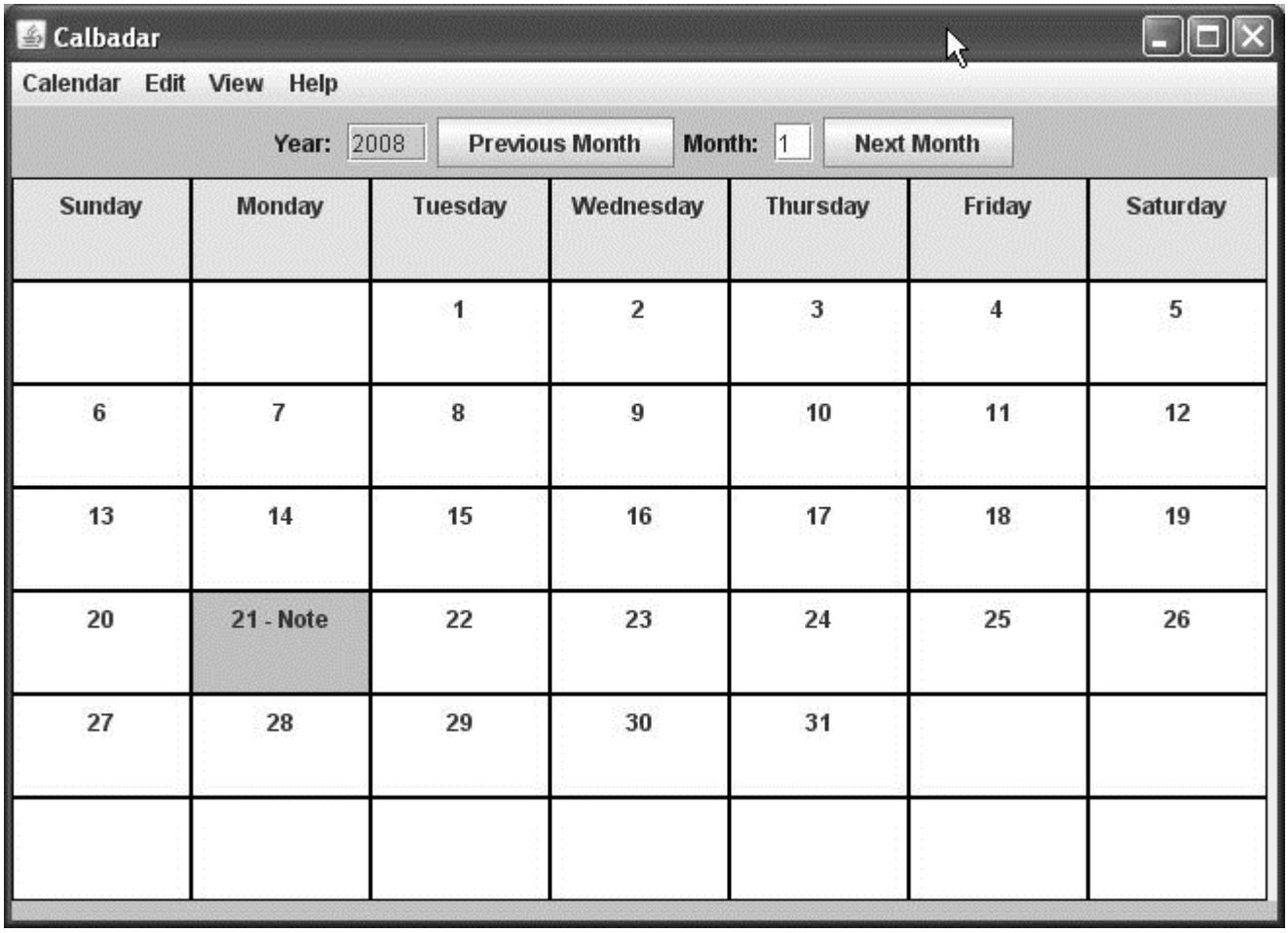

Figure B.8. Calendar main window

Calendar Configuration

The domain configuration is generated from the domain model in ModelibraModeler as the reusable XML configuration.

```
<?xml version="1.0" encoding="UTF-8"?>

<domains>

     <domain oid="1200866723313">
          <code>Util</code>
          <type>Reusable</type>

          <models>

               <model oid="1200867392284">
                    <code>Calbadar</code>
                    <author>Dzenan Ridjanovic</author>
                    <persistenceType>xml</persistenceType>
                    <persistenceRelativePath>
                         data/xml/util/calbadar
                    </persistenceRelativePath>
                    <defaultLoadSave>true</defaultLoadSave>

                    <concepts>

                         <concept oid="1200867415246">
```

```
        <code>Calendar</code>
        <entitiesCode>Calendars</entitiesCode>
        <entry>true</entry>

        <properties>
            <property oid="1200867570220">
                <code>code</code>
                <propertyClass>java.lang.String</propertyClass>
                <maxLength>16</maxLength>
                <required>true</required>
                <unique>true</unique>

                <essential>true</essential>
            </property>
            <property oid="1200867574949">
                <code>description</code>
                <propertyClass>java.lang.String</propertyClass>
                <maxLength>510</maxLength>

                <essential>false</essential>
            </property>
        </properties>

        <neighbors>
            <neighbor oid="1200867480808">
                <code>years</code>
                <destinationConcept>Year</destinationConcept>
                <inverseNeighbor>calendar</inverseNeighbor>
                <internal>true</internal>
                <partOfManyToMany>false</partOfManyToMany>
                <type>child</type>
                <min>0</min>
                <max>N</max>
            </neighbor>
        </neighbors>

    </concept>

    <concept oid="1200867420806">
        <code>Year</code>
        <entitiesCode>Years</entitiesCode>

        <properties>
            <property oid="1200867579686">
                <code>number</code>
                <propertyClass>java.lang.Integer</propertyClass>
                <required>true</required>
                <defaultValue>2008</defaultValue>
                <unique>true</unique>

                <essential>false</essential>
            </property>
        </properties>

        <neighbors>
            <neighbor oid="1200867484479">
                <code>months</code>
                <destinationConcept>Month</destinationConcept>
                <inverseNeighbor>year</inverseNeighbor>
                <internal>true</internal>
                <partOfManyToMany>false</partOfManyToMany>
                <type>child</type>
                <min>0</min>
                <max>N</max>
```

```
                    </neighbor>
                    <neighbor oid="1200867480808">
                         <code>calendar</code>
                         <destinationConcept>Calendar</destinationConcept>
                         <inverseNeighbor>years</inverseNeighbor>
                         <internal>true</internal>
                         <partOfManyToMany>false</partOfManyToMany>
                         <type>parent</type>
                         <min>1</min>
                         <max>1</max>
                         <unique>true</unique>
                    </neighbor>
               </neighbors>

          </concept>

          <concept oid="1200867422904">
               <code>Month</code>
               <entitiesCode>Months</entitiesCode>

               <properties>
                    <property oid="1200867584000">
                         <code>number</code>
                         <propertyClass>java.lang.Integer</propertyClass>
                         <required>true</required>
                         <unique>true</unique>

                         <essential>true</essential>
                    </property>
               </properties>

               <neighbors>
                    <neighbor oid="1200867488792">
                         <code>days</code>
                         <destinationConcept>Day</destinationConcept>
                         <inverseNeighbor>month</inverseNeighbor>
                         <internal>true</internal>
                         <partOfManyToMany>false</partOfManyToMany>
                         <type>child</type>
                         <min>0</min>
                         <max>N</max>
                    </neighbor>
                    <neighbor oid="1200867484479">
                         <code>year</code>
                         <destinationConcept>Year</destinationConcept>
                         <inverseNeighbor>months</inverseNeighbor>
                         <internal>true</internal>
                         <partOfManyToMany>false</partOfManyToMany>
                         <type>parent</type>
                         <min>1</min>
                         <max>1</max>
                         <unique>true</unique>
                    </neighbor>
               </neighbors>

          </concept>

          <concept oid="1200867423533">
               <code>Day</code>
               <entitiesCode>Days</entitiesCode>

               <properties>
                    <property oid="1200867588433">
                         <code>number</code>
```

```
                    <propertyClass>java.lang.Integer</propertyClass>
                    <required>true</required>
                    <unique>true</unique>

                    <essential>true</essential>
               </property>
          </properties>

          <neighbors>
               <neighbor oid="1200867502217">
                    <code>notes</code>
                    <destinationConcept>Note</destinationConcept>
                    <inverseNeighbor>day</inverseNeighbor>
                    <internal>true</internal>
                    <partOfManyToMany>false</partOfManyToMany>
                    <type>child</type>
                    <min>0</min>
                    <max>N</max>
               </neighbor>
               <neighbor oid="1200867488792">
                    <code>month</code>
                    <destinationConcept>Month</destinationConcept>
                    <inverseNeighbor>days</inverseNeighbor>
                    <internal>true</internal>
                    <partOfManyToMany>false</partOfManyToMany>
                    <type>parent</type>
                    <min>1</min>
                    <max>1</max>
                    <unique>true</unique>
               </neighbor>
          </neighbors>

     </concept>

     <concept oid="1200867424382">
          <code>Note</code>
          <entitiesCode>Notes</entitiesCode>

          <properties>
               <property oid="1200867596402">
                    <code>text</code>
                    <propertyClass>java.lang.String</propertyClass>
                    <maxLength>1020</maxLength>
                    <required>true</required>

                    <essential>true</essential>
               </property>
          </properties>

          <neighbors>
               <neighbor oid="1200867502217">
                    <code>day</code>
                    <destinationConcept>Day</destinationConcept>
                    <inverseNeighbor>notes</inverseNeighbor>
                    <internal>true</internal>
                    <partOfManyToMany>false</partOfManyToMany>
                    <type>parent</type>
                    <min>1</min>
                    <max>1</max>
               </neighbor>
          </neighbors>

     </concept>
```

```
                    </concepts>

                </model>

            </models>

        </domain>

</domains>
```

The specific XML configuration is generated together with other files generated by the main method of the `DmGenerator` class in the `dm.gen` package. The only XML element added is the `session` element with the `true` value. In Modelibra, the default value of the `session` element is `false`. A reader is invited to change maximal cardinalities of some neighbors from N to a specific number.

```
<?xml version="1.0" encoding="UTF-8"?>

<domains>

    <domain oid="1200866723313">
        <code>Util</code>
        <type>Specific</type>

        <models>

            <model oid="1200867392284">
                <code>Calbadar</code>
                <extension>true</extension>
                <extensionDomain>Util</extensionDomain>
                <extensionDomainType>Reusable</extensionDomainType>
                <extensionModel>Calbadar</extensionModel>
                <session>true</session>

                <concepts>

                    ...

                </concepts>

            </model>

        </models>

    </domain>

</domains>
```

Calendar Classes

After the code generation, some specific code was added to specific classes of concepts. This was done to ease the application programming with the model and its concepts. The `Calendar` class creates a year with its months and provides some utility `get` and `is` methods.

```
package util.calbadar.calendar;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.modelibra.IModel;
import org.modelibra.type.EasyCalendar;
```

```
import util.calbadar.year.Year;

public class Calendar extends GenCalendar {

     private static Log log = LogFactory.getLog(Calendar.class);

     public Calendar(IModel model) {
          super(model);
     }

     /* ============================== */
     /* ======= SPECIFIC CODE ======= */
     /* ============================== */

     public void createYear() {
          createYear(EasyCalendar.getOne().getCurrentYear());
     }

     public Year createYear(int number) {
          Year year = new Year(this);
          year.setNumber(number);
          year.createMonths();
          getYears().add(year);
          return year;
     }

     public Year getYear(Integer number) {
          return (Year) getYears().getYear("number", number);
     }

     public Year getYear(int number) {
          return getYear(new Integer(number));
     }

     public Year getCurrentYear() {
          int currentYearNumber = EasyCalendar.getOne().getCurrentYear();
          return getYear(currentYearNumber);
     }

     public int getCurrentYearNumber() {
          int number = 0;
          Year year = getCurrentYear();
          if (year != null) {
               number = year.getNumber().intValue();
          }
          return number;
     }

     public boolean isCalbadar() {
          boolean result = false;
          if (this.getCode().equals("Calbadar")) {
               result = true;
          }
          return result;
     }

}
```

The `Year` class creates its months and provides some utility `get` and `is` methods.

```
package util.calbadar.year;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
```

```
import org.modelibra.IModel;
import org.modelibra.type.EasyCalendar;

import util.calbadar.calendar.Calendar;
import util.calbadar.month.Month;

public class Year extends GenYear {

     private static Log log = LogFactory.getLog(Year.class);

     public Year(IModel model) {
          super(model);
     }

     public Year(Calendar calendar) {
          super(calendar);
     }

     /* ============================== */
     /* ======= SPECIFIC CODE ======= */
     /* ============================== */

     public void createMonths() {
          for (int m = 1; m < 13; m++) {
               createMonth(m);
          }
     }

     private void createMonth(int number) {
          Month month = new Month(this);
          month.setNumber(number);
          month.createDays();
          getMonths().add(month);
     }

     public Month getMonth(Integer number) {
          return (Month) getMonths().getMonth("number", number);
     }

     public Month getMonth(int number) {
          return getMonth(new Integer(number));
     }

     public Month getCurrentMonth() {
          Month month = null;
          if (isCurrent()) {
               int currentMonthNumber = EasyCalendar.getOne().getCurrentMonth();
               month = getMonth(currentMonthNumber);
          }
          return month;
     }

     public int getCurrentMonthNumber() {
          int number = 0;
          Month month = getCurrentMonth();
          if (month != null) {
               number = month.getNumber().intValue();
          }
          return number;
     }

     public boolean isCurrent() {
          boolean result = false;
          if (this.getNumber().intValue() == EasyCalendar.getOne()
```

```
                    .getCurrentYear()) {
               result = true;
          }
          return result;
     }

}
```

The `Month` class creates its days and provides some utility `get` and `is` methods.

```
package util.calbadar.month;

import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.Iterator;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.modelibra.IModel;
import org.modelibra.PropertySelector;
import org.modelibra.type.EasyCalendar;

import util.calbadar.day.Day;
import util.calbadar.day.Days;
import util.calbadar.note.Note;
import util.calbadar.note.Notes;
import util.calbadar.year.Year;

public class Month extends GenMonth {

     private static Log log = LogFactory.getLog(Month.class);

     public Month(IModel model) {
          super(model);
     }

     public Month(Year year) {
          super(year);
     }

     /* ============================== */
     /* ======= SPECIFIC CODE ======= */
     /* ============================== */

     public void createDays() {
          GregorianCalendar calendar = new GregorianCalendar();
          calendar.set(Calendar.YEAR, this.getYear().getNumber().intValue());
          // First month is 0.
          calendar.set(Calendar.MONTH, this.getNumber().intValue() - 1);
          createDays(calendar);
     }

     private void createDays(GregorianCalendar calendar) {
          int noOfDays = calendar.getActualMaximum(Calendar.DAY_OF_MONTH);
          for (int i = 1; i <= noOfDays; i++) {
               createDay(i);
          }
     }

     private void createDay(int number) {
          Day day = new Day(this);
          day.setNumber(number);
          getDays().add(day);
     }
```

```
public int getFirstWeekFirstDayNumber() {
     int monthNumber = getNumber().intValue();
     GregorianCalendar calendar = new GregorianCalendar();
     calendar.set(Calendar.YEAR, this.getYear().getNumber().intValue());
     calendar.set(Calendar.MONTH, monthNumber - 1); // First month is 0.
     calendar.set(Calendar.DAY_OF_MONTH, 1); // Sets the first wek.
     return calendar.get(Calendar.DAY_OF_WEEK);
}

public Day getDay(Integer number) {
     return (Day) getDays().getDay("number", number);
}

public Day getDay(int number) {
     return getDay(new Integer(number));
}

public Day getCurrentDay() {
     Day day = null;
     if (isCurrent()) {
          int currentDayNumber = EasyCalendar.getOne().getCurrentDay();
          day = getDay(currentDayNumber);
     }
     return day;
}

public int getCurrentDayNumber() {
     int number = 0;
     Day day = getCurrentDay();
     if (day != null) {
          number = day.getNumber().intValue();
     }
     return number;
}

public boolean isCurrent() {
     boolean result = false;
     EasyCalendar easyCalendar = EasyCalendar.getOne();
     if (this.getYear().getNumber().intValue() == easyCalendar
               .getCurrentYear()) {
          if (getNumber().intValue() == easyCalendar.getCurrentMonth()) {
               result = true;
          }
     }
     return result;
}

public Notes getNotes() {
     Notes result = new Notes(getDay(1));
     Days days = this.getDays();
     Day day;
     Note note;
     for (Iterator x = days.iterator(); x.hasNext();) {
          day = (Day) x.next();
          Notes dailyNotes = day.getNotes();
          if (dailyNotes != null) {
               note = (Note) day.getNotes().first();
               if (note != null) {
                    result.add(note);
               }
          }
     }
     return result;
```

```
    }

    public Notes getNotes(String selection) {
        Notes selectedNotes = null;
        PropertySelector propertySelector = new PropertySelector("text");
        propertySelector.defineContain(selection);
        selectedNotes = getNotes().getNotes(propertySelector);
        return selectedNotes;
    }

}
```

The `Day` class sets a valid day number and provides some utility `get` methods.

```
package util.calbadar.day;

import java.util.Date;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.modelibra.IModel;
import org.modelibra.type.EasyCalendar;

import util.calbadar.month.Month;

public class Day extends GenDay {

    private static Log log = LogFactory.getLog(Day.class);

    public Day(IModel model) {
        super(model);
    }

    public Day(Month month) {
        super(month);
    }

    /* ============================== */
    /* ======= SPECIFIC CODE ======= */
    /* ============================== */

    public void setNumber(Integer number) {
        int no = number.intValue();
        if ((0 <= no) && (no <= 31)) {
            super.setNumber(number);
        }
    }

    public Date getDate() {
        int year = getMonth().getYear().getNumber().intValue();
        int month = getMonth().getNumber().intValue();
        int day = getNumber().intValue();
        return EasyCalendar.getOne().getDate(year, month, day);
    }

    public String getDateText() {
        Date date = this.getDate();
        return EasyCalendar.getOne().getDateString(date);
    }

}
```

There is no specific code in the `Note` class. However, there is the `getContainTextNotes` method in the

`Notes` class. This methods selects a subset of nodes of the same parent that contain the given text. This is used in the *View/Monthly Notes...* window.

```
package util.calbadar.note;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.modelibra.IModel;
import org.modelibra.PropertySelector;

import util.calbadar.day.Day;

public class Notes extends GenNotes {

     private static Log log = LogFactory.getLog(Notes.class);

     public Notes(IModel model) {
          super(model);
     }

     public Notes(Day day) {
          super(day);
     }

     /* ============================== */
     /* ======= SPECIFIC CODE ======= */
     /* ============================== */

     public Notes getContainTextNotes(String subText) {
          PropertySelector propertySelector = new PropertySelector("text");
          propertySelector.defineContain(subText);
          return getNotes(propertySelector);
     }

}
```

There is one new class in the model's package with the `CalbadarDb` name. An object of this class is created by the `Board` class in the `util.swing.calbadar.month` package. The `Board` class is used in the `MainWindow` class of the `util.swing.app` package to display the current month. The `Board` class constructs an object of the `CalbadarDb` class in order to load data from the default location and the default file. The default constructor of the `CalbadarDb` class goes through several standard steps to obtain the domain configuration and to create the domain with its model based on that configuration. In addition, a persistent domain is created to provide the default XML persistency of data. The four **`public`** methods are called by the `Board` class.

```
package util.calbadar;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.modelibra.config.DomainConfig;
import org.modelibra.exception.PersistencyException;
import org.modelibra.persistency.IPersistentEntities;
import org.modelibra.persistency.IPersistentModel;
import org.modelibra.persistency.PersistentEntities;
import org.modelibra.persistency.xml.XmlEntities;

import util.PersistentUtil;
import util.Util;
import util.UtilConfig;
import util.calbadar.calendar.Calendar;
import util.calbadar.calendar.Calendars;
```

```
import util.calbadar.day.Day;
import util.calbadar.month.Month;
import util.calbadar.year.Year;

public class CalbadarDb {

     private static Log log = LogFactory.getLog(CalbadarDb.class);

     private Util util;

     private PersistentUtil persistentUtil;

     private Calbadar calbadar;

     public CalbadarDb() {
          super();
          try {
               UtilConfig utilConfig = new UtilConfig();
               DomainConfig domainConfig = utilConfig.getDomainConfig();
               util = new Util(domainConfig);
               persistentUtil = new PersistentUtil(util);
               calbadar = util.getCalbadar();
          } catch (Exception e) {
               log.error("Error in CalbadarDb.constructor: " + e.getMessage());
          }
     }

     public Util getUtil() {
          return util;
     }

     public Calbadar getCalbadar() {
          return calbadar;
     }

     public void close() {
          if (persistentUtil != null) {
               persistentUtil.close();
          }
     }

     private void createCalbadarCalendar() {
          Calendar calendar = new Calendar(calbadar);
          calbadar.setInitialized(false);
          calendar.setCode("Calbadar");
          calendar.setDescription("Modelibra model -- Swing views.");
          calendar.createYear();
          calbadar.getCalendars().add(calendar);
          calbadar.setInitialized(true);
     }

     public Calendar getCalbadarCalendar() {
          Calendar calendar = null;
          Calendars calendars = calbadar.getCalendars();
          if (calendars != null) {
               calendar = calendars.getCalendar("code", "Calbadar");
          }
          return calendar;
     }

     public void loadData() {
          boolean defaultLoadSave = calbadar.getModelConfig().isDefaultLoadSave();
          if (!defaultLoadSave) {
               try {
```

```
                persistentUtil.load();
            } catch (PersistencyException e) {
                log.error("Error in CalbadarDb.loadData: " + e.getMessage());
            }
        }
    }

    public void createData() {
        createCalbadarCalendar();
    }

    public void emptyData() {
        Calendar calbadarCalendar = getCalbadarCalendar();
        calbadar.getCalendars().remove(calbadarCalendar);

    }

    private IPersistentEntities getPersistentCalendars() {
        IPersistentModel persistentCalbadar = persistentUtil
                .getPersistentModels().getPersistentModel("Calbadar");
        IPersistentEntities persistentCalendars = persistentCalbadar
                .getPersistentEntry("Calendars");
        return persistentCalendars;
    }

    public XmlEntities getXmlCalendars() {
        PersistentEntities persistentCalendars = (PersistentEntities)
            getPersistentCalendars();
        XmlEntities xmlCalendars = (XmlEntities) persistentCalendars
                .getStoreEntities();
        return xmlCalendars;
    }

    public static void main(String[] args) {
        CalbadarDb calbadarDb = null;
        try {
            calbadarDb = new CalbadarDb();
            calbadarDb.createData();
            Calendar calbadarCalendar = calbadarDb.getCalbadarCalendar();
            Year year = calbadarCalendar.getYear(2008);
            Month month = year.getMonth(1);
            Day day = month.getDay(21);
            calbadarDb.close();
        } catch (Exception e) {
            log.error("Error in CalbadarDb.main: " + e.getMessage());
            calbadarDb.close();
        }
    }

}
```

Only the `Board` class will be completely presented in this appendix. The calendar board of the current month is initialized in the default constructor. The board can be filled by a different month by using the `fill` method. Default data can be created by the `createDefaultData` method, or data can be loaded from the default file by the `loadDefaultData` method. Data can also be loaded from a specific location by the `loadData` method, and saved to that or a different location by the `saveData` method.

```
package util.swing.calbadar.month;

import java.awt.Color;
import java.awt.GridLayout;
import java.util.Iterator;
```

```
import javax.swing.JPanel;

import org.modelibra.Session;
import org.modelibra.persistency.xml.XmlEntities;

import util.calbadar.CalbadarDb;
import util.calbadar.calendar.Calendar;
import util.calbadar.day.Day;
import util.calbadar.month.Month;
import util.calbadar.year.Year;
import util.swing.calbadar.day.Cell;
import util.swing.param.Para;

public class Board extends JPanel {

     public static final Color HEADER_COLOR = Color.YELLOW;

     public static final Color DAY_COLOR = Color.WHITE;

     public static final Color CURRENT_DAY_COLOR = Color.LIGHT_GRAY;

     public static final int BOARD_LENGTH = 7;

     private Cell[][] cells;

     private CalbadarDb calbadarDb;

     private Session session;

     private Month month;

     public Board() {
          super();
          init();
     }

     public CalbadarDb getCalbadar() {
          return calbadarDb;
     }

     public Month getMonth() {
          return month;
     }

     private void init() {
          cells = new Cell[BOARD_LENGTH][BOARD_LENGTH];
          this.setLayout(new GridLayout(BOARD_LENGTH, BOARD_LENGTH));
          for (int v = 0; v < BOARD_LENGTH; v++) {
               for (int h = 0; h < BOARD_LENGTH; h++) {
                    cells[v][h] = new Cell(this);
                    this.add(cells[v][h]);
                    if (v == 0) {
                         cells[v][h].setColor(HEADER_COLOR);
                         // Sunday, Monday, ...
                         cells[v][h].setDayName(Para.getSingleton().getText(
                                   "day" + h));
                    } else {
                         cells[v][h].setColor(DAY_COLOR);
                    }
               }
          }
     }
```

```
private void empty() {
     for (int v = 1; v < BOARD_LENGTH; v++) {
          for (int h = 0; h < BOARD_LENGTH; h++) {
               cells[v][h].setColor(DAY_COLOR);
               cells[v][h].setNumber(0);
               cells[v][h].setDayName("");
               cells[v][h].emptyNoteIndicator();
               cells[v][h].setDay(null);
          }
     }
}

public void fill(Month month) {
     this.month = month;
     empty();
     int noOfEmptyCells = month.getFirstWeekFirstDayNumber();
     Day day = null;
     int v = 1;
     int h = noOfEmptyCells - 1;
     for (Iterator x = month.getDays().getList().iterator(); x.hasNext();) {
          day = (Day) x.next();
          Integer number = day.getNumber();
          int no = number.intValue();
          cells[v][h].setNumber(no);
          cells[v][h].setDay(day);
          cells[v][h].updateNoteIndicator();
          if (month.isCurrent()) {
               if (no == month.getCurrentDayNumber()) {
                    cells[v][h].setColor(CURRENT_DAY_COLOR);
               }
          }
          if (++h == BOARD_LENGTH) {
               h = 0;
               v++;
          }
     }
}

private void fill(Calendar calendar) {
     try {
          Year currentYear = calendar.getCurrentYear();
          Month currentMonth = currentYear.getCurrentMonth();
          if (currentMonth != null) {
               fill(currentMonth);
          }
     } catch (Exception e) {
          System.out.println("Fill data error from Board.fill: "
                    + e.getMessage());
     }
}

public void loadDefaultData() {
     calbadarDb = new CalbadarDb();
     session = calbadarDb.getCalbadar().getSession();
     calbadarDb.loadData();
     Calendar calbadarCalendar = calbadarDb.getCalbadarCalendar();
     if (calbadarCalendar == null) {
          calbadarDb.createData();
          calbadarCalendar = calbadarDb.getCalbadarCalendar();
     }
     fill(calbadarCalendar);
}

public void createDefaultData() {
```

```
        calbadarDb = new CalbadarDb();
        session = calbadarDb.getCalbadar().getSession();
        calbadarDb.createData();
        Calendar calbadarCalendar = calbadarDb.getCalbadarCalendar();
        fill(calbadarCalendar);
    }

    public Session getSession() {
        return session;
    }

    public void loadData(String filePath) {
        try {
            XmlEntities calendars = calbadarDb.getXmlCalendars();
            if (filePath != null) {
                calendars.setDataFilePath(filePath);
            }
            calendars.load();
            Calendar calbadarCalendar = calbadarDb.getCalbadarCalendar();
            if (calbadarCalendar == null) {
                calbadarDb.createData();
                calbadarCalendar = calbadarDb.getCalbadarCalendar();
            }
            fill(calbadarCalendar);
        } catch (Exception e) {
            System.out.println("Load data error from Board.loadData: "
                    + e.getMessage());
        }
    }

    public void saveData(String filePath) {
        try {
            XmlEntities calbadars = calbadarDb.getXmlCalendars();
            if (filePath != null) {
                calbadars.setDataFilePath(filePath);
            }
            calbadars.save();
        } catch (Exception e) {
            System.out.println("Save data error from Board.saveData: "
                    + e.getMessage());
        }
    }

    public void emptyData() {
        calbadarDb.emptyData();
        session.getHistory().empty();
        empty();
    }

}
```

A user action can be undone by the *Undo* menu item in the *Edit* menu. The undo code is simple and it is provided in the `MainMenu` class of the `util.swing.app` package.

```
        menuEditUndo.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Session session = mainFrame.getBoard().getSession();
                session.getHistory().undo();
            }
        });
```

The `Session` class comes from the `org.modelibra.Session` package.

Web Links

[Books] Java Swing free books
http://www.javaolympus.com/freebooks/FreeJavaSwingBooks.jsp

[Puzzles] Puzzles
http://www.puzzles.com/

[Swing] Java Swing
http://java.sun.com/docs/books/tutorial/uiswing/

[ThinkFun] ThinkFun
http://www.thinkfun.com/