COMP-533 Model-Driven Software Development

Assignment 3
Video Slot Machine
GUI Design with Reusable Aspect Models

`(10% of final grade)`

November 27, 2013

Preparation

To solve this assignment you need to install TouchRAM. You can download the stand-alone application, which is distributed in form of a `.jar` file, from here: http://www.cs.mcgill.ca/~joerg/SEL/TouchRAM.html. The application should run on any machine that runs Java and has a relatively decent graphics card[1]. If graphic performance is low, try switching to windowed mode, changing the resolution or adjusting the OpenGL settings in the `Settings.txt` file. The QuickReference guide gives a very short overview of how to use TouchRAM.

You should also download the files *SlotMachine.ram* and *SlotMachineGUI.ram*, two RAM models that you are going to extend for this assignment. You must place these two files in the *models* folder in the TouchRAM application folder.

Finally, you can also download several pdf files with sample RAM models from the course webpage. The models include the design pattern RAM models (such as *Observer* and *Singleton*), *Utility* RAM models (such as *Map* and *Named*), the entire *Workflow* middleware, as well as the *NetworkCommand* RAM model and dependent *SocketCommunication*.

We would be very happy to receive constructive feedback on TouchRAM, i.e., how we could improve the tool in the future. If you are interested in working on the tool as part of a undergraduate or masters project / thesis, come and talk to me. In case you have trouble using the tool, please send Omar (or me) an email.

Problem Statement, Use Cases, and Requirements Specification

Same as for the final.

8 Partial Design

Fig. 1 shows a Reusable Aspect Model (RAM) that describes the design structure of the EGM that encodes the basic state of the slot machine[2]. The message view *systemStartup* describes the behaviour that is triggered from inside the constructor of the *SlotMachine* class to create and initialize all important objects as follows:

- First, the operation opens the file "icons.info", and reads from it a set of icon names (in form of Strings), each representing a distinct icon (we assume that there are no duplicate icon names in the file). It then instantiates for each icon name a corresponding *Icon* object and registers it with the *IconManager*.

[1] It seems like on the Macintosh, Java 1.7 is not supported. Make sure you run it under Java 1.6.

[2] The file `SlotMachine.ram` that is part of the handout for assignment 3 contains the same structural elements as the ones shown in Fig. 1.

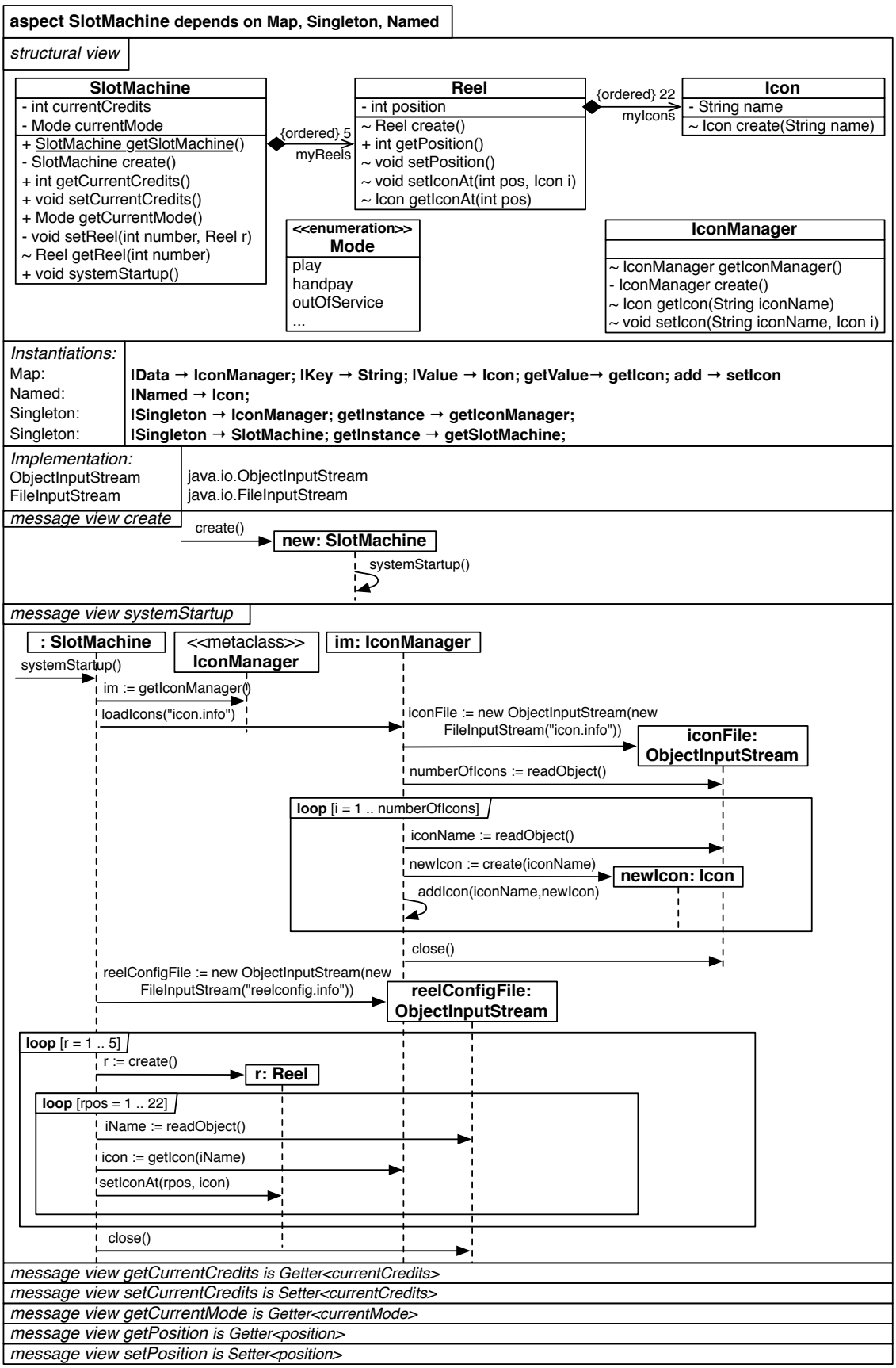

Figure 1: The SlotMachine Reusable Aspect Model

- Then, the operation opens the file "reelsConfig.info", and reads from it 5 times 22 icon names. Within each of the 5 iterations, a reel is instantiated and the 22 reel positions are initialized with the icons that corresponds to the read icon names.[3]

9 GUI Design

Assignment Summary

Your task is to design a GUI for the EGM that can display the 5 reels at their current position (i.e., for each reel, the icon associated with the current position of the reel, together with the two previous icons and the two following icons). Whenever the reel position changes, the GUI should immediately update accordingly. The additional structure and behaviour needed for the GUI must be designed in a separate RAM model called *SlotMachineGUI* that extends the *SlotMachine* RAM aspect shown in Fig. 1. All your design must be done within the *SlotMachineGUI* model. You are not allowed to modify the *SlotMachine* model.

Tools to Use / Documents to Handin

You are to use TouchRAM to design the structural extensions in the *SlotMachineGUI* model. Since TouchRAM unfortunately does not support editing of message views yet, you have to use some other UML or drawing tool to create the message views for *all* operations that you define. Only if an operation simply accesses attributes of the object (for example, like getters and setters do), or if the behaviour of the operation is provided by some other aspect's message views, then you do not have to define your own message view for that operation.

To complete the assignment you must hand in your modified copy of the `SlotMachineGUI.ram` file, plus a pdf / paper copy depicting the message views for all operations that you added to the *SlotMachineGUI* RAM model.

Assignment Details

Some design decisions have already been made for you.

- The pictures that are to be used to graphically represent the icons are stored in files using the PNG format with the names `iconname.png`, where *iconname* corresponds to the names stored in the *icons.info* and *reelConfig.info* files used during system startup.
- A specific 3rd party GUI framework was chosen to handle the drawing of pictures on the display of the slot machine. In particular, two classes, *Canvas* and *Picture*, are provided to you by the framework[4]. The *Canvas* class is an abstraction of a display or drawing surface. The *Picture* class provides two operations: a constructor that can initialize a picture by reading its content from a file in PNG format, and a `draw` operation that displays the picture at a specified x and y position on a given canvas object.

To simplify your task, I outline the different design steps needed to accomplish the GUI design in the following subsections.

9.1 Initializing the Graphical Representation of Icons

The *SlotMachineGUI* model already contains the two implementation classes provided by the GUI framework, *Canvas* and *Picture*. It extends the *SlotMachine* RAM model shown in Fig. 1. You can explore both models by opening SlotMachineGUI in TouchRAM. Double-clicking on the SlotMachine dependency will open the SlotMachine model[5].

You must now use TouchRAM to extend the structural view of the *SlotMachineGUI* model to associate a picture with each icon of the slot machine. Then, you should specify message views in *SlotMachineGUI* that affect the system startup behaviour to initialize the pictures. As a reminder, augmenting a message view m with additional behaviour is done in RAM as follows:

[3] Note: the complete EGM would also have to instantiate and initialize the paytable, the log and the cumulative meters during system startup, but this is out of the scope of this assignment.

[4] This means that you are not allowed to modify the *Canvas* or *Picture* classes

[5] *Note*: Since our tool currently does not support enumeration types, the *Mode* enumeration type in the TouchRAM model is simply encoded as a String.

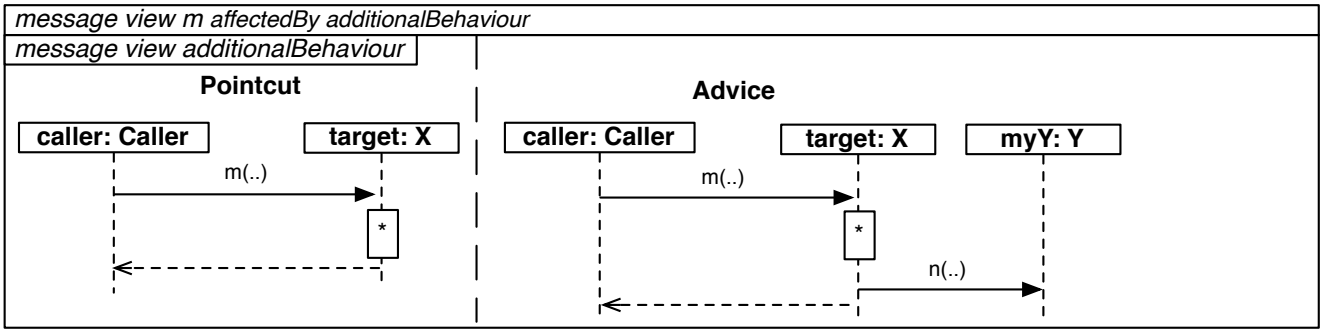

Figure 2: RAM Aspect Message Views

9.2 Creating the Reel Views

We are going to use a simplified version of the *Model-View-Controller* (MVC) design pattern to integrate the GUI with the slot machine base classes. In this step you must create the *ReelView* class that is capable of displaying the state of a reel. To accomplish this step, you need to use TouchRAM to add a *ReelView* class to the *SlotMachineGUI* RAM model. It should have a current x and y position, and a canvas on which it is displayed. Then, design the `draw` operation that takes as a parameter a reel and displays the 5 icons that correspond to the current reel position on the canvas at the position (x,y-2*iconHeight), (x,y-iconHeight), (x,y), (x,y+iconHeight) and (x,y+2*iconHeight).

9.3 Initializing the Reel Views

To initialize the GUI, you are to create a *SlotMachineGUI* class in the *SlotMachineGUI* RAM model that provides an `initializeGUI` operation that creates a canvas of width *displayWidth* and height *displayHeight*. Then it should create a reel view instance for each reel, and position it at (reelXPos,reelYPos), (reelXPos+iconWidth+horizontalBuffer,reelYPos), (reelXPos+2*(iconWidth+horizontalBuffer,reelYPos)), etc.

Adapt the system startup behaviour so that it creates a *SlotMachineGUI* instance and calls `initializeGUI` on it.

9.4 Updating the Views

Use the *Observer* RAM model to ensure that the reel views are redrawn whenever a reel is set to a new position. To depend on the *Observer* model, hit the "+" button on the top left of the TouchRAM drawing area, and then select the *Observer* model in the design patterns folder of the reusable model library. To establish the mappings, click on the "triangle" button next to "Observer", then click on "C+" to add a mapping for */Subject*. Then create a mapping for */modify* by clicking on "O+". Repeat the same procedure for establishing a mapping of */Observer* and */update*.

Generate the Complete Design

When you are done, you might want to click on "Weave All" to look at the final design class diagram that combines the structure of all RAM models (*SlotMachineGUI*, *SlotMachine*, *Observer*, *ZeroToMany*, *Map*, *Singleton* and *Named*) into one.

Hand-In

Please hand in the `SlotMachineGUI.ram` file and a pdf or paper copy of your message views until Tuesday December 3rd by sending an email to Omar.Alam@mail.mcgill.ca with the title "COMP-533 assignment 3 of yournames" and cc me as well (Joerg.Kienzle@mcgill.ca). If you don't get an acknowledgment for your email, send us another email (without attachment, but putting the handin somewhere where we can download it).

Remember that you are allowed to work in groups of 2, but not with a person you worked with for the final or for a previous assignment. If you work with someone, please hand in a single copy with both names.