Systems Modeling

State modeling: Final assignment

Acknowledgements: This project is adapted from the one proposed at http://msdl.cs.mcgill.ca/people/hv/teaching/MSBDesign/201011/assignments/Statecharts/.

Practical Information

Due date: Tuesday, December 9th 2014, before 10:00 AM

Submission:

- Bitbucket (GIT) PRIVATE repository containing the eclipse project. You must TAG your repository in a way that we can check the evolution of your solution (i.e. add one TAG when you completely implement one requirement).
- A document describing, e.g. markdown file or a Word file in the root folder of your bitbucket repository, the solution (fully satisfied requirements, open issues – hopefully none, etc.), your conclusions and, of course, and the list of team members. NOTE: You must not change the code of the GUI (to check this requirement, we will use the original code).

Grant admin access to Amnir to your bitbucket repository (his username is simply "Amnir").

Introduction

In this assignment, you will specify a state model for the behavior of a Digital Watch (inspired by the 1981 Texas Instruments LCD Alarm Chronograph).

As for the in-class practice sessions, you must design the Statechart (to be submitted), and test it with Yakindu IDE. You are not required to write Java code, for this assignment you will be provided with an implementation for the GUI ready to be plugged to the statechart. You are allowed to add only internal events and variables but not to modify the interfaces "Buttons", "Display" nor "LogicUnit".

Behavior requirements

1. The time value should be updated every second, even when it is not displayed (as for example, when the chrono is running). However, time is not updated when it is being edited.
2. Pressing the top right button turns on the background light. The light stays on for as long as the button remains pressed. From the moment the button is released, the light stays on for 2 more seconds, after which it is turned off.
3. Pressing the top left button alternates between the chrono and the time display modes. The system starts in the time display mode. In this mode, the time (HH:MM:SS) and date (MM/DD/YY) are displayed.
4. When in chrono display mode, the elapsed time is displayed MM:SS:FF (with FF hundredths of a second). Initially, the chrono starts at 00:00:00. The bottom right button is used to start the chrono. The running chrono updates in 1/100 second increments. Subsequently pressing the bottom right button will pause/resume the chrono. Pressing the bottom left button resets the chrono to 00:00:00. The chrono will keep running (when in running mode) or keep its value (when in paused mode), even when the watch is in a different display mode (for example, when the time is displayed).
   Note: interactive simulation of a model containing time increments of 1/100 second is possible, but it is difficult to manually insert other events. Hence, while you are simulating your model, it is advisable to use larger increments (such as 1/4 second) for simulation purposes.
5. When in time display mode, the watch will go into time editing mode when the bottom right button is held pressed for at least 1.5 seconds.
6. When in time display mode, the alarm can be displayed and toggled between on or off by pressing the bottom left button. If the bottom left button is held for 1.5 seconds or more, the watch goes into alarm editing mode. This is not an example of good User Interface design, as going to editing mode will also toggle on/off and that may not be desired. It is however how the 1981 Texas Instruments LCD Alarm Chronograph works. The first time alarm editing mode is entered, the alarm time is set to 12:00:00. The alarm is activated when the alarm time is equal to the time in display mode. When it is activated, the screen will blink for 4 seconds, and then the alarm turns off. Blinking means switching to/from highlighted background twice per second. The alarm can be turned-off before the elapsed 4 seconds by a user interrupt (i.e.: if any button is pressed). After the alarm is turned off, activity continues exactly where it was left-off.

7. When in (either time or alarm) editing mode, briefly pressing the bottom left button will increase the current selection. Note that it is only possible to increase the current selection, there is no way to decrease or reset the current selection. If the bottom left button is held down, the current selection is incremented automatically every 0.3 seconds. Editing mode should be exited if no editing event occurs for 5 seconds. Holding the bottom right button down for 2 seconds will also exit the editing mode. Pressing the bottom right button for less than 2 seconds will move to the next selection (for example, from editing hours to editing minutes).

To help clarify the requirements, you can find a working solution in the zip file (to run the demo use "java –jar digitalwatch.jar", note that the file "watch.gif" must be in the working directory).

Starting point

The zip file provided contains an Eclipse project with a specification of the interfaces to connect the Statechart with the GUI and other related Java classes. As the entry point for starting the application, you must use the class org.yakindu.src.test.Main. The latter class sets up all the elements in the application (State machine, Timer Service, GUI controllers and views, etc.) and launches the simulation.

Concerning the architecture, the Digital watch is conceptually decomposed on three components exposing the following interfaces:

Interface Buttons:

This interface give a hook for the simulation to generate the events associated with the buttons in the watch. We distinguish the following events:

topRightPressed
topRightReleased
topLeftPressed
topLeftReleased
bottomRightPressed
bottomRightReleased
bottomLeftPressed
bottomRightReleased

Interface Display:

This interface exposes the functions associated with the watch display. Not that this interface defines only operations and no event. The list of operations supported is the following:

refreshTimeDisplay()	Redraw the time with the current internal time value. The display does not need to be cleaned before calling this function. For instance, if the alarm is currently displayed, it will be deleted

	before drawing the time.
refreshChronoDisplay()	See refreshTimeDisplay()
refreshDateDisplay()	See refreshTimeDisplay()
refreshAlarmDisplay()	See refreshTimeDisplay()
setIndiglo()	Turn on the display background light.
unsetIndiglo()	Turn off the display background light.
showSelection()	When in edition mode, this method shows the value of the currently selected element (e.g., hour)
hideSelection()	When in edition mode, this method hides the value of the currently selected element (e.g., hour) Use showSelection() and hideSelection() to provide an animation for the edition mode

Interface LogicUnit:

This interface exposes the behavior internal to the digital watch (e.g., time counting, date counting, alarm raising, etc.). The set of operations and their descriptions follows:

getTime()	Returns the current clock time.
getTimeLabelAsForShowing()	When in edition mode, this method returns the current time/alarm label to be displayed.
getTimeLabelAsForHiding()	When in edition mode, this method returns the current time/alarm label to be displayed, hiding the currently selected part (e.g., hour) The methods getTimeLabelAsForShowing() and getTimeLabelAsForHiding() are internally used for implementing the animation for edition mode. YOU SHOULD NOT USE THEM!
increaseTimeByOne()	Increase the time by one second. Note how minutes, hours, days, month and year will be modified appropriately, if needed (for example, when increaseTimeByOne() is called at time 11:59:59, the new time will be 12:00:00).
getDate()	Returns the current date.
getDateLabelAsForShowing()	When in edition mode, this method returns the current date label to be displayed.
getDateLabelAsForHiding()	When in edition mode, this method returns the current date label to be displayed, hiding the currently selected part (e.g., year) The methods getDateLabelAsForShowing() and getDateLabelAsForHiding() are internally used for implementing the animation for edition mode. YOU SHOULD NOT USE THEM!

getChrono()	Returns the current time label for Chrono mode.
increaseChronoByOne()	Increase the chrono time by 10 milliseconds. All the other parts, i.e., seconds, minutes, and hours will be updated accordingly.
resetChrono()	Reset the time count for the chrono.
setAlarm()	Toggles on/off the alarm on the watch.
getAlarm()	Checks whether the alarm is set or not. This method is internally used. YOU SHOULD NOT USE IT!
startTimeEditMode()	Initializes the internal variables for edition mode, by selecting the leftmost digit group on the time label.
startAlarmEditMode()	Initializes the internal variables for edition mode, by selecting the leftmost digit group on the alarm label.
increaseSelection()	Increases the currently selected digit group's value by one.
selectNext()	Select the next digit group, looping back to the leftmost digit group when the rightmost digit group is currently selected. If the time is currently being edited, it also selects the groups on the date label.

Additionally, the interface LogicUnit raises the event “startAlarm” when the alarm is set and, of course, when the time reaches the programmed time for the alarm.