PRA2003 Programming Skills

Week 4: Graphical User Interfaces

Cameron Browne & Andreea Grigoriu

20 November 2018

This week, you will learn how to implement simple *graphical user interfaces* (GUIs) in Java using the Abstract Window Toolkit (AWT) and Swing tools. This includes implementing "listeners" based on mouse and keyboard events, such as mouse clicks.

Task 4.0. Understanding and playing with GUIs

Study this week's slides and familiarise yourselves with the Swing and AWT GUI components. Compile and run the examples (`SimpleFrame.java`, `TestGUI.java`, `Listener.java`), modify them, and see how GUIs work. Make sure you understand how listeners work.

If necessary, study the Java API for mouse and key listeners:
https://docs.oracle.com/javase/8/docs/api/java/awt/event/MouseListener.html
https://docs.oracle.com/javase/8/docs/api/java/awt/event/KeyListener.html

---

Description of Tasks 1-3

After finishing Task 4.0, move on to solving Tasks 4.1, 4.2 and 4.3. Each task this week involves modifying a file GUI.java available via EleUM. This program is incomplete but your first two tasks will complete it (note that it won't compile until you complete Task 4.2).

Task 4.1. TTDGameState Interface

Create the missing `TTDGameState` interface. This interface should contain the signatures of the three public methods in `TTDBoard` (i.e. not the constructor or `nextTurn()`). Recall from Week 1 for format and requirements of interfaces, and the implications/requirements for classes that implement them.

Note that the program will still not compile at this stage, but show it to your instructor before moving on to Task 4.2.

Task 2. Mouse Event Handling

Implement the `MouseEventTrapper` class. This class serves to trap mouse clicks, modify the state accordingly, and update the main panel. It keeps a reference to two objects: the main panel and the board (game state).

To complete this task, simply get the program to compile by implementing the event trapper. Add the appropriate constructor and necessary methods required by MouseListener. Most of them will be empty methods (since we do not need them). Provide a very simple mouse click handler:

```
public void mouseClicked(MouseEvent e)
{
   System.out.println("Click at:  (" + e.getX() + "," + e.getY() + ").");
}
```

Compile and run the program to make sure that clicking a mouse button generates a response. What X value do you see? What Y value do you see?

Task 3. Tic-Tac-Doge

First, download the images **doge-x.png** and **doge-o.png** from EleUM and place them in your project's base folder.

Now, finish the GUI for this simple game by modifying `MouseClicked(MouseEvent e)` in the `MouseEventTrapper`.

This method should first map the point coordinate values to a board row (0..2) and board column (0..2). Then, if the clicked cell is empty, it should:

1. Add the appropriate label in the main panel using `addDoge(...)`, and
2. Change the game state by invoking the board's `clickCell(...)` method.

Confirm that the GUI works and that the pieces are being drawn properly. Verify that clicking on a cell that already contains an **X** or **O** icon does change whose turn it is!

Assignment 4 **** Due 23:59 Monday 3 Dec. — Hard Deadline! ****

Using the code for Tic-Tac-Doge (TTD), add a graphical user interface to Emerald Mine. Images for the various world objects are available via the EleUM (or feel free to use your own graphics to customise the game).

In TTD, the width of each cell was 100 pixels and the height was 100 pixels. In Emerald Mine, reduce each dimension to 32 pixels. Therefore, a `World` with 20 rows and 20 columns should give a total resolution of 640 x 640 pixels, which should be okay for all screens.

Remove the TTD game state and replace it with a reference to the `World` that is initially created.

Rename the panel to `EmeraldMinePanel` and modify it so that the appropriate number of rows and columns are created to match those in the world. Replace the image icons with the relevant Emerald Mine images. This may require adding some getters to World (depending on how you implement it).

Instead of using a `MouseListener` and `MouseEventTrapper`, create a `KeyEventTrapper` class that implements `KeyListener`. The only method that is not empty is `keyPressed(KeyEvent e).` As in TTD, this method must:

1. Handle the keyboard event,
2. Invoke the world's `applyMove()` method to change the world accordingly, and
3. Set the icons and refresh the GUI as necessary.

Key press codes are handled via the `e.getKeyCode()` method: it will return an integer constant representing the key pressed. The key press codes of interest here are:
- KeyEvent.VK_UP
- KeyEvent.VK_RIGHT
- KeyEvent.VK_DOWN
- KeyEvent.VK_LEFT

These correspond to the *non-numpad* cursor keys.

Important Note: The `KeyEventTrapper` must be added as a key listener to the *frame*, not the panel! This is different than in TTD and requires a single-line change in the `GUI.init()` method.

Good news is that no change is required to `World.applyMove(...)`. It should work exactly as in the previous assignment.

Redrawing the world requires resetting the icon for the corresponding object in each cell, based on the current state of the world. This may require adding a getter to the `World`. To do this, add a method to `EmeraldMinePanel` that resets all icons:

```
public void redrawWorld(final World world)
{
    // Reset icons by looping over every object in the world
}
```

To reset an icon, see the code in `addDoge(..)` in TTD.

Note: The panel's `repaint()` method should only be called once, after all the label's icons have been (re)set.

As in TTD, most of the code to handle the key event, update the world, and update the GUI (except `redrawWorld`) is contained in the `KeyEventTrapper`. This is done using the appropriate method invocations, just as in TTD.

Honour code, coding style and deliverable

Please refer to Infosheet → Paragraph 7 → Honour policy.

You are welcome to expand your program to do extra things but they are not mandatory. Do not deviate from the specific requirements of the assignment!

If you use some code found online (or anywhere else, e.g. StackOverflow), clearly state it (with comment about where you found it) in your .java files. That means mention the author and the www source you found it. Be sure that you read the terms of the licence for the code that you are reusing. Most code that is found openly in the internet has specific licences that you should not violate.

Through courses portal, find Assignments→Assignment 4, submit a zipped file (.zip) that contains your source files (.java) for the assignment.

Please refer to the coding style guidelines uploaded to courses portal (under Course materials) on how to write readable programs.

Hard deadline for submission:

23:59 on Monday 3 December