CSIS0396A/COMP2396A - Assignment 4

Due: 9th Nov, 2014 23:30

Introduction

This assignment tests your skills on writing simple **GUI** program, with **graphics** and **event handling** in Java.

As an extended version of **Assignment 2**, you are required to write a GUI to interact with the "simulated" forest, which holds the same animal hierarchy, moving and attacking rules, except that the movement of animals can be **under your control**. You may control it by dragging the mouse to move a specific animal object, and an animal can move continuously as you wish.

This assignment will be evaluated on both **functionality** and **program design**. You can get part of the full marks if you implement some of the features.

You are expected to use **GUI graphics** to display the interface for this assignment.

You are also required to write **JavaDoc** for all non-private classes and non-private class members. Programs without JavaDoc will not be marked.

Part I. Classes in your project.

For the requirements of the forest and the hierarchy of animals, please refer to the specification in the Assignment 2. Note that you should also make good use of inheritance and polymorphism when implement the "simulated" forest.

Your project is required to contain Java files for the following classes:

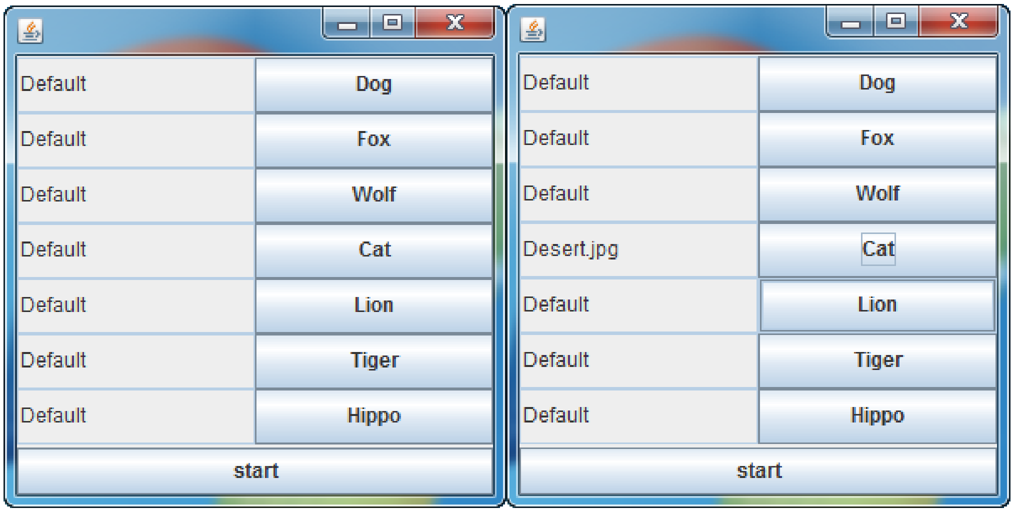
1) Animal class;
2) Subclasses of (1), namely Canine, Feline and Hippo;
3) Subclasses of (2) respectively, namely [Dog, Fox, Wolf] of Canine, [Cat, Lion, Tiger] of Feline;
4) AnimalGUI class, as the main program, with main() method inside.

You may need to implement additional classes.

Part II. The GUI interface.

When the program starts, it creates a window on your screen, with a table of size **15 × 15** inside as the forest. Each cell can only hold one animal. That is to say, if at some time, an animal A moves into the cell of another animal B, A will attack B before moving.

When the program starts, a setup menu should be shown which allows the user to select animals to be included in the current simulation run. User is allowed to select any animal using the setup menu. Each animal has a default image which will be shown in the "simulated" forest. On the other hand, user is allowed to select any image file for the selected animal in the current simulation run.



Default	Dog
Default	Fox
Default	Wolf
Default	Cat
Default	Lion
Default	Tiger
Default	Hippo

Default	Dog
Default	Fox
Default	Wolf
Desert.jpg	Cat
Default	Lion
Default	Tiger
Default	Hippo

Figure 1: Setup Menu (a VERY SIMPLE example)

The selected animals are then put into different cells of the 15x15 table randomly. To indicate which animal is in a particular cell, the cell is shown filled with an **icon** corresponding to that kind of animal.

To be precise, a 2-tuple (i, j) denotes the position of a cell, where i ∈ [0, 14], j ∈ [0, 14].

Part III. Animal moving.

Rules in animal moving are the same as Assignment 2, namely at one move, a Feline has 8 options, a Canine has also 8 options, and a Hippo has 4.

The movement of animals can be in one of the following 2 modes:

1. Auto mode: animals are moving in the simulated forest automatically in the same manner as described in Assignment 2.
2. Manual mode: movements of animals are controlled manually by **dragging the mouse**. Specifically, if an animal A is currently at (i, j), and can move to (i + 1, j) as an option at one step, then the user **press the mouse** at cell (i, j) and drag it to the corresponding target cell. Note that the icon is not required to be with the cursor during dragging, but make sure it causes a move or attack when you release it at a proper position, and update the layout and icons in the forest.

User can switch between auto mode and manual mode through a button.

Part IV. Animal attacking.

Rules in attacking are also the same as Assignment 2.

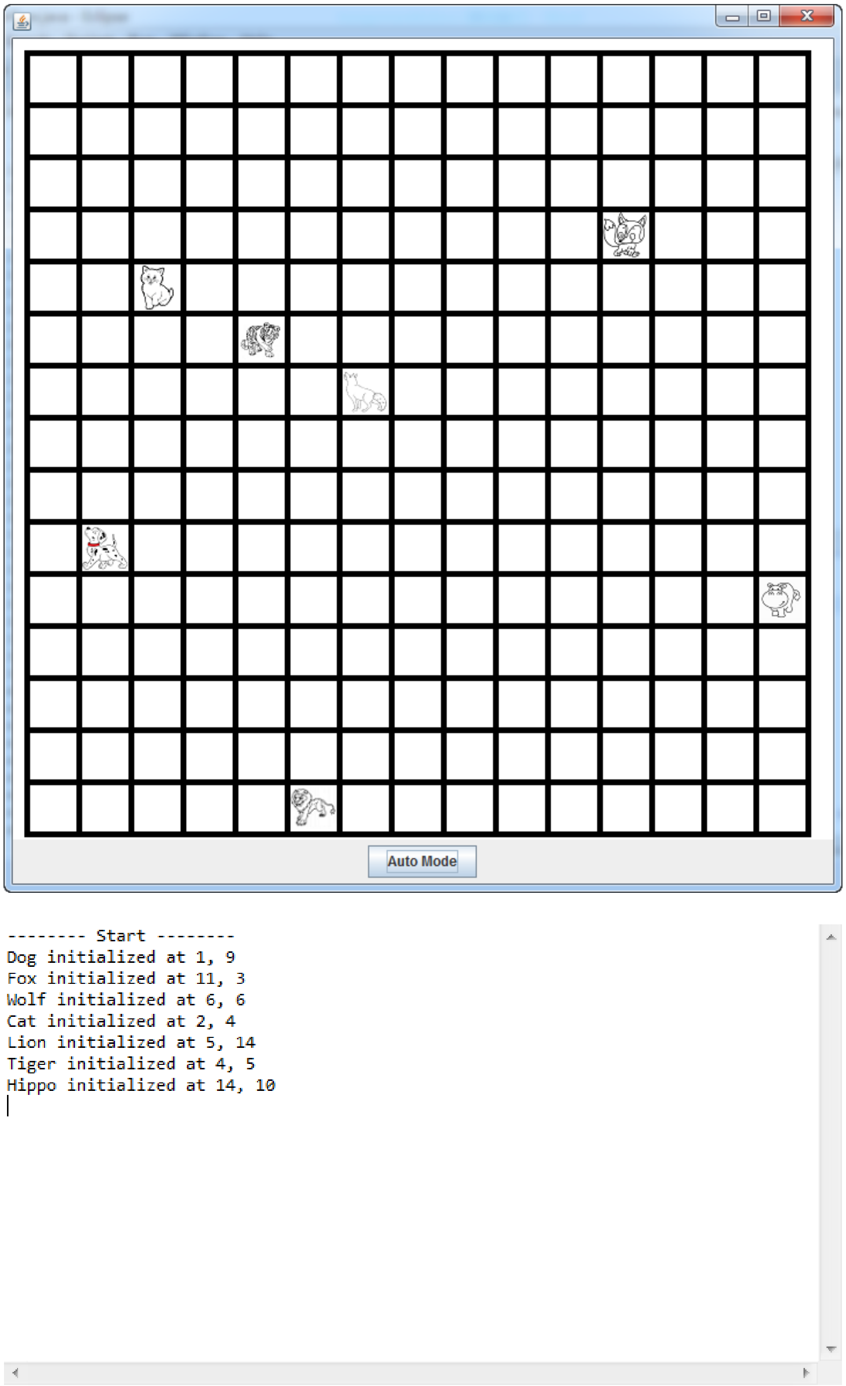
In Part III, if you drag the animal A onto another animal B, and A can just move to B at one move, the attack happens. Just like that in Assignment 2, if the attacker wins, it kills the defender and moves to the new cell, where the defender is removed. Otherwise the defender kills the invader and stays there. The attacker is removed.

Interface at the beginning

At the beginning, different animal are initialized in different cells and shown with their icons.

Default icons are provided to you, each of size **40 × 40** pixels.

Meanwhile, some information should be printed in the console. See Figure 2.



```
-------- Start --------
Dog initialized at 1, 9
Fox initialized at 11, 3
Wolf initialized at 6, 6
Cat initialized at 2, 4
Lion initialized at 5, 14
Tiger initialized at 4, 5
Hippo initialized at 14, 10
```

Figure 2: initialization

Animal moving

After the initialization, animals are located in different cells respectively.

In auto mode, animals are moving around follow the rules as described in Assignment 2.

During each step in the manual mode, a user may press the left mouse button to specify the animal to be moved. Of course nothing happens when you press on an empty cell. But when there is an animal A, (holding the left button)

1) Cells that it can reach at one move will be painted with orange color;
2) If there are other animals under its attack range, their cells will be covered with red color.

If you just release the mouse at its original cell or other cells it cannot reach, nothing happens. See Figure 3.

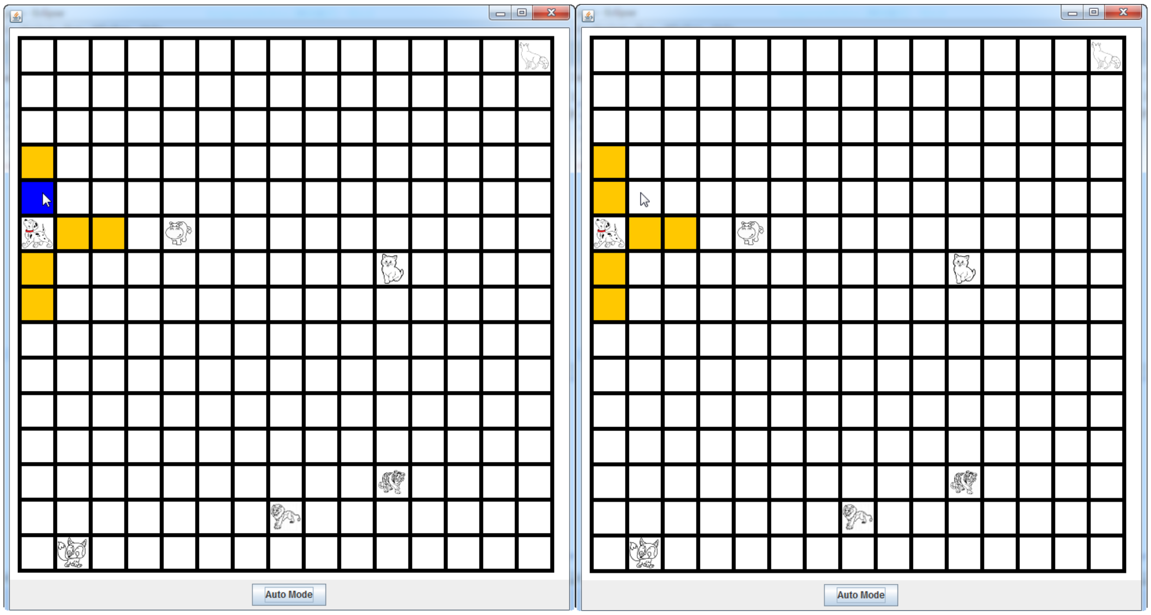


Figure 3: Press on the Dog cell and drag before release

But if you move the mouse (holding the left button) to one of the optional cell, the color of that cell will be painted with blue as a real-time process, even if there is another animal. And if you just release the left button at an empty cell it can reach, the animal A simply moves to the new cell with the icon.

See Figure 4 & 5.

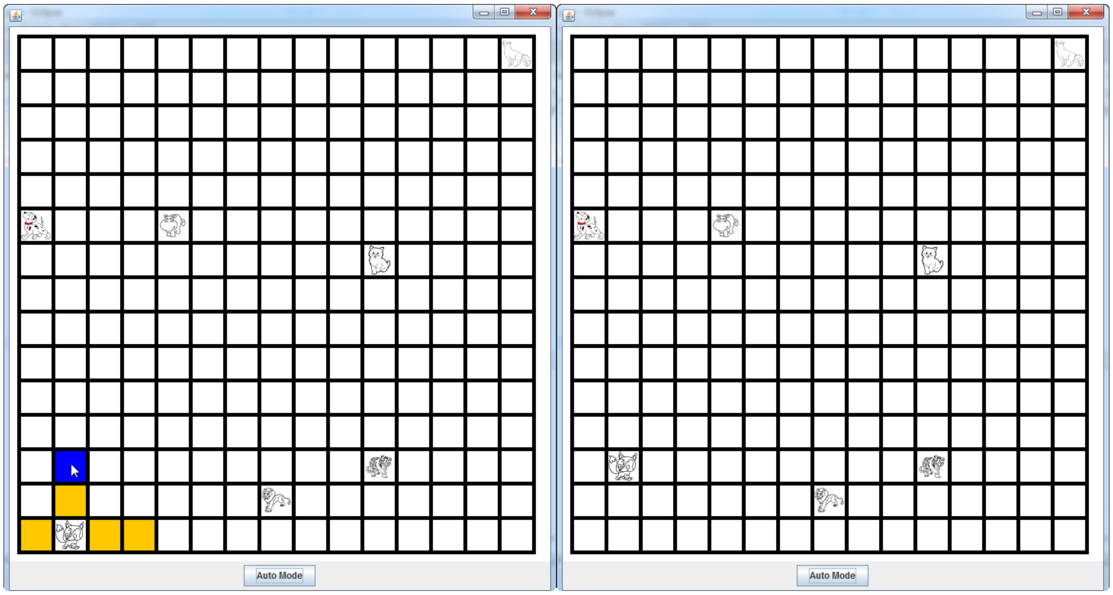
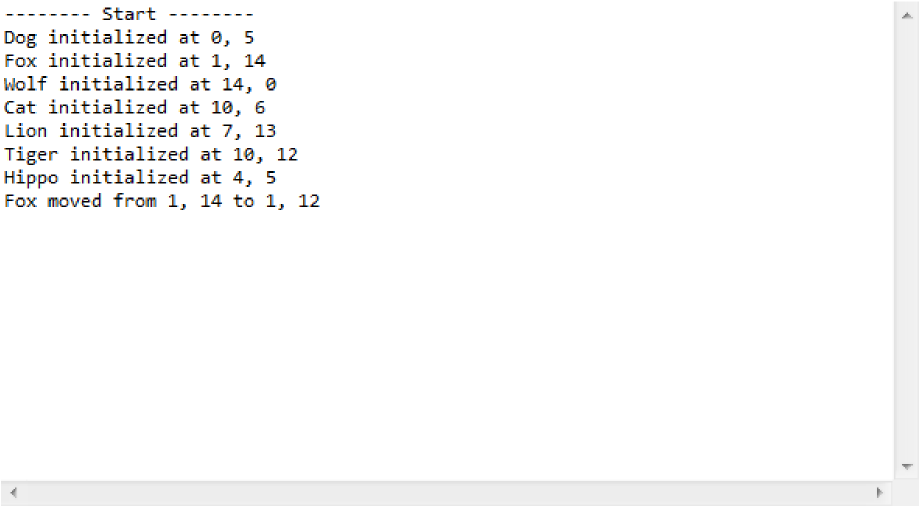


Figure 4: Press on the Fox cell, drag and release at another place.



```
-------- Start --------
Dog initialized at 0, 5
Fox initialized at 1, 14
Wolf initialized at 14, 0
Cat initialized at 10, 6
Lion initialized at 7, 13
Tiger initialized at 10, 12
Hippo initialized at 4, 5
Fox moved from 1, 14 to 1, 12
```

Figure 5: Print information.

Animal attacking

At the previous step, if you release the left button on a cell where another animal stays (make sure the animal A can reach there), A will attack that one.

See Figure 6 & 7 & 8.

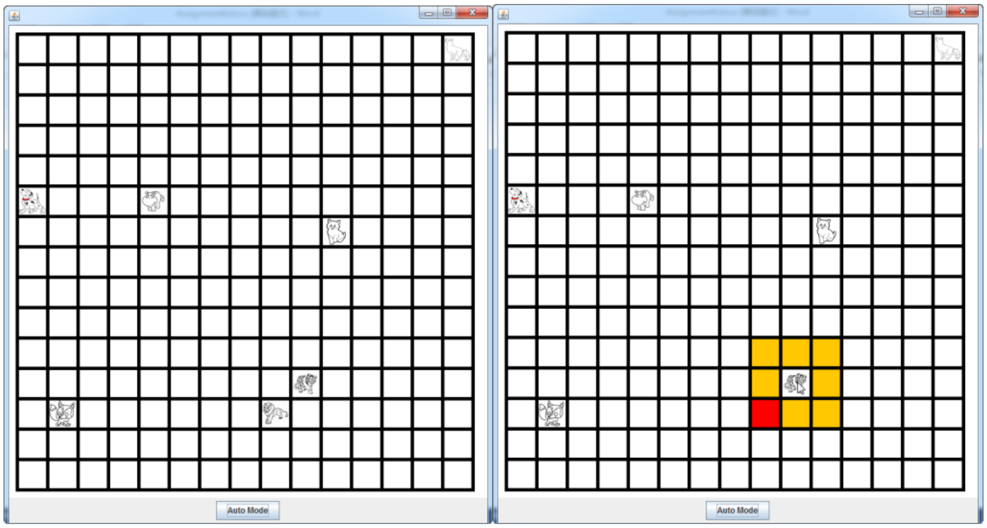


Figure 6: The Lion is in the Tiger’s attack range.

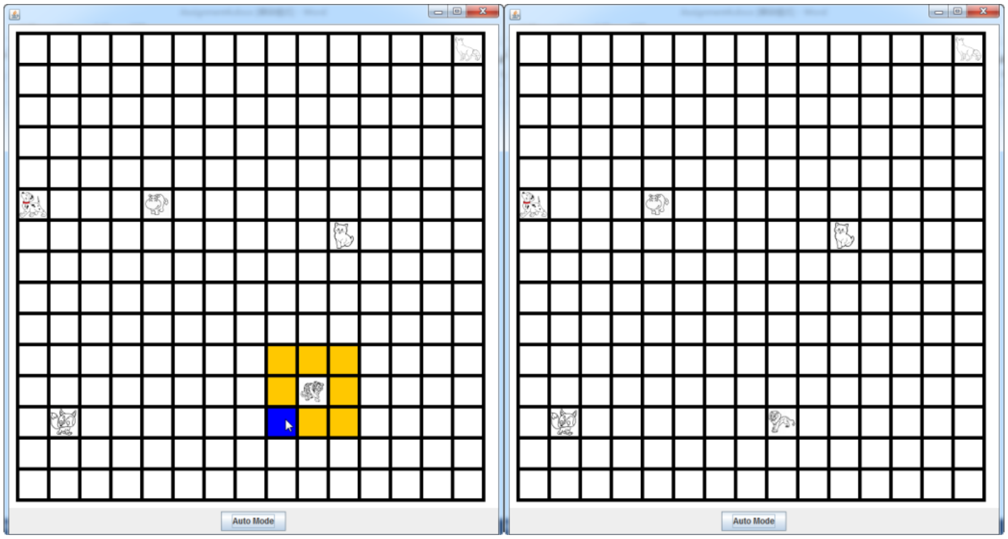
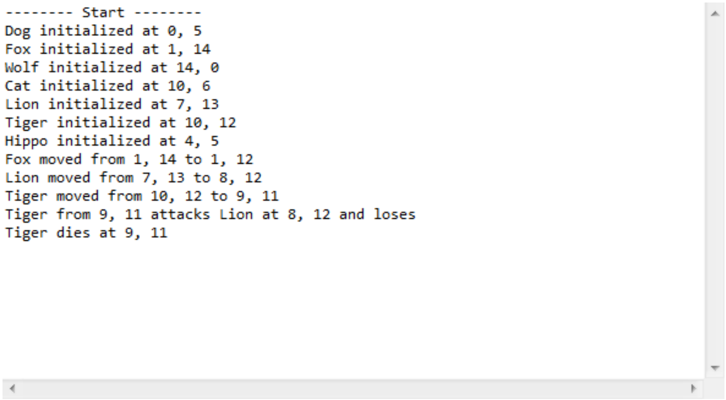


Figure 7: Drag the mouse and release on the Lion cell. Tiger attacks Lion and loses.



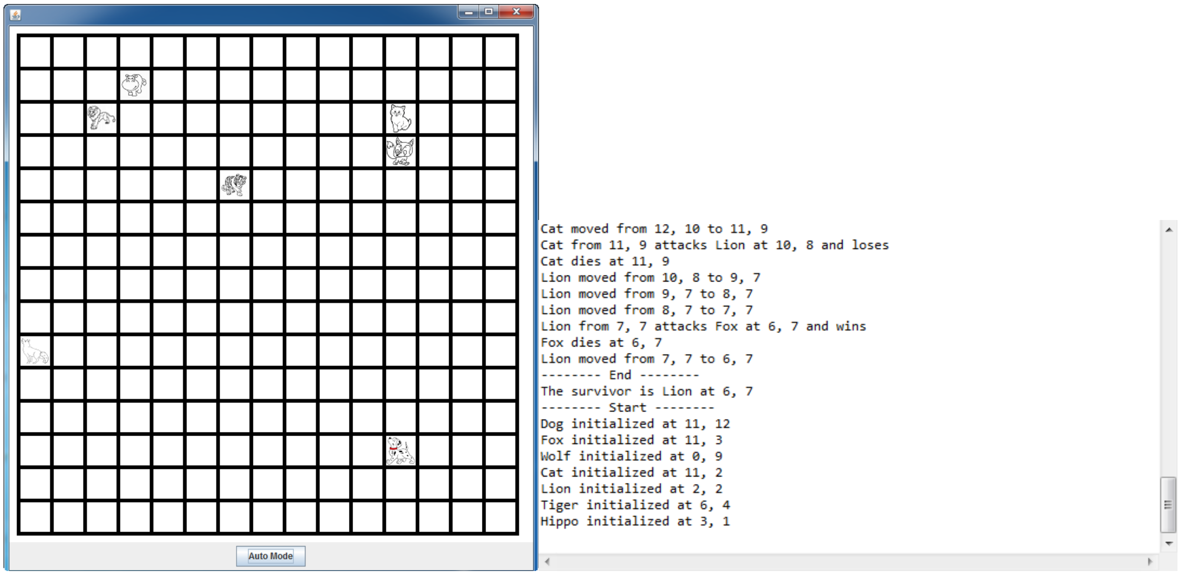
```
-------- Start --------
Dog initialized at 0, 5
Fox initialized at 1, 14
Wolf initialized at 14, 0
Cat initialized at 10, 6
Lion initialized at 7, 13
Tiger initialized at 10, 12
Hippo initialized at 4, 5
Fox moved from 1, 14 to 1, 12
Lion moved from 7, 13 to 8, 12
Tiger moved from 10, 12 to 9, 11
Tiger from 9, 11 attacks Lion at 8, 12 and loses
Tiger dies at 9, 11
```

Figure 8: Print information.

New round

The program should be able to start a new round (i.e. 7 animals created again randomly) when there is only one survivor in the table. It can be triggered either immediately after the final attack or with a simple click.

See Figure 9.



```
Cat moved from 12, 10 to 11, 9
Cat from 11, 9 attacks Lion at 10, 8 and loses
Cat dies at 11, 9
Lion moved from 10, 8 to 9, 7
Lion moved from 9, 7 to 8, 7
Lion moved from 8, 7 to 7, 7
Lion from 7, 7 attacks Fox at 6, 7 and wins
Fox dies at 6, 7
Lion moved from 7, 7 to 6, 7
-------- End --------
The survivor is Lion at 6, 7
-------- Start --------
Dog initialized at 11, 12
Fox initialized at 11, 3
Wolf initialized at 0, 9
Cat initialized at 11, 2
Lion initialized at 2, 2
Tiger initialized at 6, 4
Hippo initialized at 3, 1
```

Figure 9: Restart.

Marking

- **80% marks** are given to the **functionality**.
  - You may add additional classes, instant variables and methods to the project
  - You will get part of the full marks if you implement some of the features.
  - A program that can run normally without throwing exceptions during runtime gets higher marks.
  - Don't forget to print the information as required above during runtime in console.
- **20% marks** are given to the **program design.**
  - You should make good use of inheritance, polymorphism, layout, painter and event handler.
  - You can check it by avoiding code duplication as much as possible
  - **Economy is valuable in coding: the easiest way to ensure a bug-free line of code is not to write the line of code at all.**

Submission:

Please submit all source files (`*.java`) in a single compressed file (in `.zip` or `.7z`) to Moodle. **Late submission is not allowed**.

Do not submit .*class* file.

-- END --