CSE1720

Week 08, Lab 07

Winter 2014 ◆
Thursday, Feb 27, 2014 & Friday, Feb 28, 2013

YORK U
UNIVERSITÉ
UNIVERSITY

Lab 07 Exercise

- **Due Date:** Wednesday, March 5th, 2013, 11:59pm
- **Course Weight:** 2%
- **Topic**: Skills with subclasses and collections

Once you have completed the exercises, you need to submit your files. Use the following URL:
https://webapp.eecs.yorku.ca/submit/

Submit the following files (see following pages for instructions):

```
Lab07Answers.txt
Controller.java
ChaosWorld.java
```

Exercise #1

Examine the “game” code base. OK, its not really a game yet, but we’re getting there ☺

Run the app by invoking `AppDriver.java`

Examine the code, specifically looking in the class `SimulationRunnable` and the method `run()`. Examine the following statement:

```
ActionListener frameAdvancer = new Controller(pict);
```

Please note: the class `FrameAdvancer` has been renamed `Controller`.

Q.1 What is the advantage of declaring the variable `frameAdvancer` as `ActionListener` instead of a `Controller`?

Write your answer in the a file called `Lab07Answers.txt`

You may wish to prepare your answer after completing the other exercises (the advantage will become apparent by then)

Exercise #2

In Exercise #1, you ran the app. Examine the class `Controller`, specifically the constructor and the `actionPerformed` method.

- constructor: sets up a collection of 10 `BasicSprite` instances
- `actionPerformed` method: moves and redraws the sprites in the collection

Note that the `Sprite` interface has been modified to have two separate methods: `move` and `draw`. Familiarize yourself with each of the `Sprite` subclasses.

Behold the splendor of the beautiful delegation!

You can add whichever Sprite instances you like into the collection and they all will get moved and redrawn.

Modify the class `Controller` so that the game world consists of 40 sprites: 10 of each subclass.

Exercise #3

Whoops! Premature celebration! The splendor of the beautiful delegation is not so splendid after all… Why?

Alas our class does not yet make full use of delegation ☹

The **model** of the game world includes the number and types of `Sprite`s within it. (The model doesn't include the `Sprite` behaviours directly, since each `Sprite` encapsulates its own type of behaviours.) In the class `Controller`, the details about the game world **model** are enmeshed with the details about how the frames are updated and drawn (the **view**).

Modify the class `SimulationRunnable` to use the class `ControllerV2` instead of `Controller`. See that it behaves in the same way as `Controller` does (in its original version, before you changed it in exercise #2). The two classes have behaviours that are exactly the same, but `ControllerV2` uses delegation, whereas `ControllerV2` does not.

Exercise #4

When you examine `ControllerV2`, you should be able to see that all of the specifics about the **model** of the game world (e.g, the number and types of `Sprite`s within it) are now encapsulated within the class `BoringWorld`.

There are three subclasses of `GameWorldModel`: `BoringWorld`, `BoringReverseWorld` and `SkitteryWorld`. Modify `ControllerV2` to try out each one of the three different game worlds.

Exercise #5

Create a copy of one of the subclasses of `GameWorldModel`. Call it `ChaosWorld.`

Modify `ChaosWorld so` that the game world consists of 40 sprites: 10 of each subclass.

Set up your world to have 100 of each sprite subclass.

Set up your world to have 1000 of each sprite subclass.

Notice anything? Revert back to 100 of each sprite subclass.

Modify `ChaosWorld` to ensure that the dots are always drawn on top of the dogs (the dogs should not obscure the dots).

Hint: change the type of your collection to one that maintains ordering, and let the drawing exploit the ordering.