SCC110: Software Development Term 2. Java Programming.

Lancaster University

Problem Set 5: Version control and working in a team!

Aims

This week's lab aims to:

- Gain more experience of working with Git
- Gain experience of working in programming teams.
- Provide more practice with Object Oriented thinking.
- Give you initial experience of building more complex objects.
- Gather feedback on the quality of your work from your lab assistants.

The Task

This week's tasks build on the version control principles we saw in the lectures, and the GameArena examples you worked on last week. If you feel there are some concepts from these sessions that you haven't understood, talk to one of the TAs so that you're building on strong foundations.

Lancaster University

Task 1: Writing a Simple Game Collaboratively

FOR THIS EXERCISE YOU WILL NEED TO WORK IN A SMALL GROUP OF IDEALLY TWO OR THREE PEOPLE (Definitely no more than FOUR).

Firstly, you are going to need a team. Feel free to create your own group, or we will allocate you to a group at the start of your lab session if you prefer.

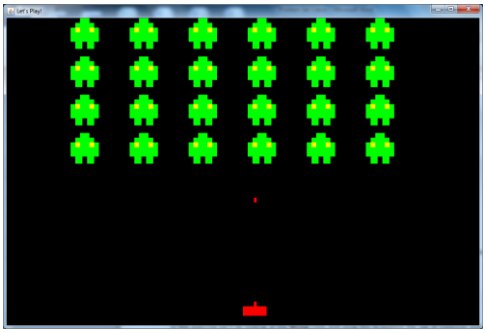
Last week you experimented with the **GameArena** class to draw and move primitive shapes on the screen. The week, we're going to build on these classes to create more complex shapes that you can move just as easily through the use of OO concepts.

Imagine you want to build a simple computer game based on the **GameArena** class and the **Rectangle** class. You plan to stick together several rectangles to create a representation of objects in your game and write Java classes to control these as characters in your game.

Do some brainstorming in your group to generate an idea for a simple game. You're free to imagine any game you want but it should have:

- A Goodie (the player, e.g. a person, a dog, a car, a big honking space invaders gun, etc.)
- Baddies (things that player interacts with, e.g. cats, alien spaceships, road hazards, etc).

e.g. As a test case, I built a very simple version of Space Invaders. This is **much** more complex than we expect from you, but notice how the aliens and base ship are all made up of multiple rectangles!



Working as a team:

- Elect one of your team members to host a github repository. That user should go to their homepage on github and log in (**http://www.github.com/<your username>**)
- Create a **private**, **MIT licensed** repository for your game. You'll need a good name of course, so get imaginative! You can do this by selecting the **Repositories** tab and clicking **New**.
- Invite the rest of your team as collaborators to the repository. You'll find this under the **Settings->Collaborators** page of your repository on github.
- **Add** all the **GameArena** files from last week's task into your repo. **Commit** and **push** your changes, as we saw in the lecture.
- Make sure that all your team-mates can **clone** the repository on their computer.

Lancaster University

For each type of character in your game (e.g. a Goodie or Baddie), assign each team member the following task. Make sure each team member creates a character:

- Create a Java class to represent that character in your game. Choose an appropriate name for the class.
- Add private instance variables to represent the x and y location of your character.
- Add private instance variables in to hold the rectangles that make up the character. Using an array here might be a good idea if you want to keep your code concise… especially if you plan to have a lot of Rectangles!
- Write a constructor that takes an x and y location as its parameters. Use this constructor to initialize your location instance variables and to create your rectangles. **Get creative here to make a shape that looks the part!**
- Write a public method called **addTo**. This method should take a GameArena as its sole parameter. When called, the method should add all the rectangles in your object to the GameArena provided.
- Write a public method called **move**. This method should move that character in the x and y direction **relative to its current position**. This method should update the instance variables representing the location of your character and in each of the Rectangle instances used to draw the character.

Write a main method:

- Create a separate class named after your game and write a main method in it.
- Create an instances of GameArena and the classes you just wrote to represent the characters in your game.
- Using the methods you have just written, add your characters to your GameArena window.
- Write a loop that calls the move method on your Goodie. Your Goodie should move based on the up, down, left and right cursor keys. **HINT**: If you're wondering how to get user input? Checkout the JavaDoc documentation for the GameArena class for an easy way to detect these key presses…
- Update your loop so your baddies move according to some simple rules…

Gasp in awe at your wonderful creation!

Note in particular how grouping the data and code related to a particular character into its own class has helped to keep your code contained as the complexity of your program grows… and how you could work independently as a team without needing to edit the same files at the same time.

SCC110: Software Development Term 2. Java Programming.

Lancaster University

Additional Tasks

If you want to do more work…

1. Make sure your code is in keeping with the OO concepts. Is there any code in your main() method that should really be inside your other classes? Remember: **Code that describes the behaviour of an object should be encapsulated inside that object's class…**

2. Any game of this style is likely to need accurate collision detection to determine when two graphical objects touch each other… Write a method for your Rectangle class that will detect a collision between two rectangles. Show how you can use this to accurately detect a collision between your game characters…

If you have already completed all tasks:

- **Reflect upon your work**. Do you feel it follows the OO principle of encapsulation? Is all the functionality cleanly wrapped into the relevant classes?

- **Ensure you have commented your code**. Pay particular attention to documenting your public methods. Use Javadoc comments and use the Javadoc tool to generate a set of web pages documenting your code.

- **Discuss your code with your lab assistants** and gather feedback on how your code can be improved.

- **Showcase your work.** Create a short video demo of your work. Let's build a showcase of all your projects in there! It doesn't matter if your project isn't going to be the next big hit on PlayStation 5 – let's all share in your successes anyway! Send me a link to your video via Teams or email.