SCC110: Software Development Term 2. Java Programming.

Lancaster University

Problem Set 4: Encapsulation and Integration

Aims

This week's lab aims to:

- Let you practice protecting your classes with public and private.
- Give you experience of using and writing accessors, mutators and constructors.
- Teach you how to create and use instances of classes written by others.

Exercise 1: The Person Object Again!

Last week you created a Person class, to practice writing your own classes. We saw how to define a class, and how to instantiate (make an instance of) that class. However, your Person class had no protection with public/private. ***It was not encapsulated***. We saw in this week's lectures that we should take ownership and responsibility for the classes we create… so let's quickly go back and fix that before we move on.

Following the guidelines in this week's lecture:

- Protect all the instance variables of your Person class so that they cannot be directly accessed from outside the class.
- Write **accessor** (get) methods for your **name** and **age** instance variables.
- Write a **constructor** method for your Person class. This should initialize a Person's age to zero (as it always is when a Person is created!), but allow the name to be initialized via a parameter to the constructor.
- Update your **main**() method in the Driver class to use your new constructor and accessor methods.
- Try to access your private instance variables directly and see what happens… 😊
- Write **JavaDoc** comments for your Person class. Remember that we use JavaDoc to document every **class** and **method**. Generate the JavaDoc web pages for your Person class to see how easy it is.

SCC110: Software Development Term 2. Java Programming.

Lancaster University

Exercise 2: Using classes written by others

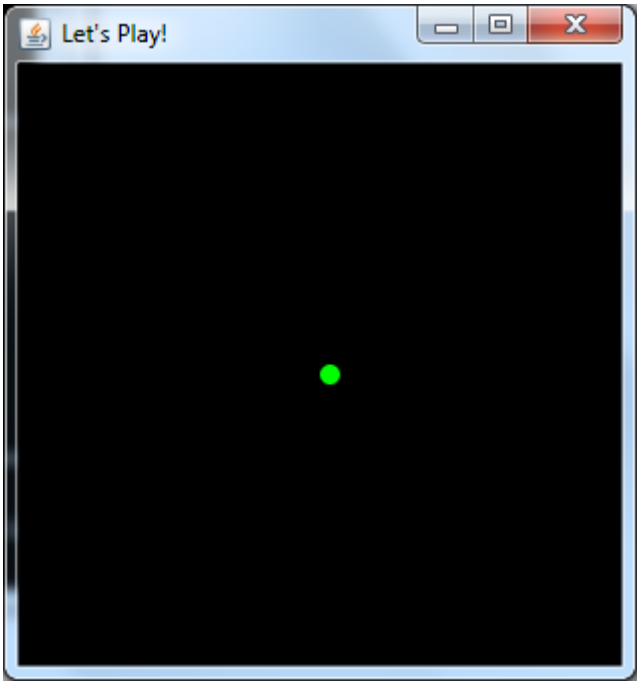
By now you should have the hang of creating instances of classes and calling methods on those instances. So let's look at a slightly more realistic example… I've written a few classes for you that lets you write simple, animated 2D computer games. :-D

On Moodle you will find a file called **GameArena.zip**. **Download** it and **extract** it into wherever you store your Java programs. You'll find three classes in the GameArena.zip file: **GameArena**, **Ball** and **Rectangle**. You'll find these classes are well documented:

- Browse into the doc folder, and double click index.html.
- Notice my documentation is the same format as we saw in the lecture – because I wrote it using JavaDoc.
- Browse through the available constructors and methods to see what these classes can do.
- Take a quick look at the Java code for the Ball and Rectangle classes. These should feel very familiar to you (compare to your own Person class for example).

I'm making the Ball and Rectangle classes your responsibility. Feel free to add new instance variables and methods to these classes. Don't remove any of the methods or instance variables already defined these classes though... else the software may not be able to operate.

You should however consider the GameArena class off limits. That one is my responsibility… :)



Lancaster University

Task 1: Draw a Ball on the Screen

Time to write a program that uses the classes you have been given…

- Write a new class containing a main method. Name it as you wish.
- Inside your main method, create an instance of the GameArena class.
- Also create an instance of the Ball class. Ensure this Ball is your favourite colour…
- Work out from the documentation how to make your ball appear somewhere near the middle of the GameArena window.

  HINT: Read all the method documentation, to make sure you understand how to use the GameArena class… Start by opening the **index.html** file in a web browser.

Task 2: Movement

Balls that don't move are a bit like a broken pencil…. Pointless!

- Add a while loop to the end of your main method that loops forever.
- Inside your loop change the position of the ball such that it moves horizontally across the screen. You should reset its location back to zero if it goes off the end of the window.

Task 3: Bouncing

Time to add the world's simplest "physics engine" (the bit of code in a computer game that simulates realistic movement).

- Update your code so that your ball can move both horizontally and vertically.
- Write code such that your ball bounces off the four sides of the window.

HINT:

This is much easier than you think. It's possible in just a few lines of code… Break the problem down. Consider:

- What you need to do each time around the loop
- Think about how you can detect if your ball reaches the edge of the window
- How you can reverse your ball's x or y direction when it does…

Task 4: Lots of Balls…

Update your program to handle multiple Balls all moving at once. You might want to make use of an array to make your code scalable… Write your code so that it is easy to define the number of balls to generate.

Note: If you've written your program in an object oriented way, this is really easy… if you weren't object oriented, you'll start to find you code becomes much more complex. If you find your program is suddenly starting to feel complex, perhaps consider how augmenting the Ball class might help…

Lancaster University

SCC110: Software Development Term 2. Java Programming.

Additional Tasks

I've noticed some of you have programmed before in Java and/or like a bit of a programming challenge…. Why not try some of the following ideas… these are purely optional exercises designed to stretch you and are not compulsory for the course.

For each task, try to be as object oriented as you can. Remember: always encapsulate your code in the appropriate class, and think where the **BEST** place to add code is. Make sure you show your code to me or a teaching assistant when you're done, so we can give you some feedback on how well you have structured your code.

- Try to develop your program such that you can launch a Ball on a random trajectory when it is created. You might find the random() method in Java's Math class useful for this… lookup the JavaDoc online to see how it works.

- Allow the user to define the number of balls that are generated using a command line parameter to your program. HINT: That's what that String array is for that is provided as a parameter to you main() method…

- Simulate gravity! Remember that Sir Isaac Newton taught us that objects accelerate downwards at about 9.8 metres per second per second… Now you wouldn't want to break the laws of Physics now would you? ☺

- Get creative!