ELEC 279 - Winter 2020
Introduction to Object-Oriented Programming
Lab 2 - Week 3

Simple Classes in JAVA

Welcome to ELEC 279 Lab Session where you will learn object-oriented programming (OOP) by practicing and work as a team. Throughout the lab sessions, you will be working in the *pair programming model* [1] - usually two (2) programmers per workbench and in rare case, a group of three (3) programmers will be permitted at the approval of a *Graduate Teaching Assistant* coordinating the lab. Please, arrive early to the lab and get settled to start working on the lab - arriving 15 minutes after the start of a lab is considered late and could affect your participation point for the lab.

Pair Programming Model: In order to promote an equal learning environment, we will be evaluating your participation and results in the labs, which means every member of your group must participate. In the pair programming model, each of the two team members can either be the *driver* or the *navigator* and their roles are:

- *the driver:* the person typing at the keyboard
- *the navigator:* the person with the "eagle eyes," watching for mistakes and/or typos.

One of you can only assume one role at a time, and you can take turns to interchange the role during each lab session. An example is, for Lab 1 Task 1, one person assumes the *driver* role while the other person serves as the *navigator*. And for Task 2, the *navigator* for Task 1 becomes the *driver*, and vice versa. This is the model we will adopt for this course throughout the lab sessions in this semester.

Lab Grading and Assessment: Note that your TAs are here in this Lab to help you learn. Feel free to ask questions as the TAs will be going round to make sure all members of the group are participating. For each Lab assessment, YOU MUST demonstrate your results for each **Task** to get credit for that **Task** - do not wait until the end to show all the tasks, demonstrate each task to the TAs as you progress. Remember, the total Labs is worth 15% of your final grade.

1 Introduction

In this lab you will learn about basic classes and encapsulation. Classes are the blueprints for objects. They specify the instance variables (i.e., attributes) and behaviors of an object. There are two main components of a class:

1. Attributes or Instance Variables
    - is some object or collection of objects

[1] https://tinyurl.com/ydf9mwos

- are instantiated when an object is created
- are declared outside of methods
- are accessible by any method of that class
- are typically set to private

2. Methods
   - is a function attached to the object
   - Constructors are important methods that are used to instantiate an object
   - methods can be public or private ie accessed by anyone or only by the class
   - provide the main functionality of an object

2 Getting Started in Eclipse

This Lab has included test code that you must satisfy to get full marks. There are a couple steps that must be completed first to set up the lab in Eclipse. Like last lab **let one member be the driver and the other the navigator.**

1. Open Eclipse
2. Start by creating a project called **"InLab2"**
3. Select **File→New→Java Project** to create a new project
4. Name the project **"InLab2"**
5. Click finish

3 Classes in Java and Eclipse

Object-Oriented Programing is based on the creation of classes of objects. Java and Eclipse make it vary easy to start writing classes right away. We will start by creating the main class for this lab. It contains test code to evaluate the classes you will be making.

1. Open Eclipse
2. Now we must create the main class for this project
3. Select **File→New→Class** to add a new class
4. Under **Name** enter **"InLab2"**
5. Click finish
6. Copy the text from **"InLab2x.java"** on onq to your newly created class

For this lab you must make two classes. The first class will be called "Activity" and is used to store dates. We will walk you through the basics of setting up this class in Java using Eclipse.

1. Create a new class in you project and call it **"Activity"**
2. As we wish to use this class in other files ensure that it is public
3. Within the class brackets add three private attributes:
   (a) private int day;
   (b) private int month;
   (c) private int year;
4. It is good practice to make your attributes private. This prevents anyone from changing an object's information willy nilly. Additionally, through the use of encapsulation techniques we can check that all changes are acceptable. In our case we will have to ensure that the date is always a valid one. so that people can't input more than the number of days in a month or more than the number of months in a year.
5. Our next step is to add a constructor
6. After your attributes write down **public Activity(){}** this will be our default constructor
7. Within this constructor set all of our attributes to 1
8. Add another constructor, **public Activity(int dayIn, int monthIn, int yearIn){}** this will be used to initialize our instance variables (i.e. attributes).
9. Following this set the year then the month then the day to the input date. Ensure that you check that the date entered is valid.
10. ***Stop Here*** Ask the TAs to evaluate your code to get graded for this portion of the lab. (1 mark)
11. At this time it is recommended that you **switch from navigator to driver and vise-versa.**
12. Now add 3 methods to get the value of each of our attributes. These methods are called accessors as they access our attributes.
    (a) public int getDay(){}
    (b) public int getMonth(){}
    (c) public int getYear(){}
13. Fill in these methods to return their matching attribute value.
14. Now we will add a mutator method that sets the date to something else.

15. At this time it is recommended that you **switch from navigator to driver and vise-versa.**
16. Write a method **public void setDate(int dayIn, int monthIn, int yearIn){}**
17. Update the attributes with these values only if it is a valid date. Otherwise print an error statement using: System.out.println()
18. Last add a method to print the contents of our object
19. Call the method **public void printDate(){}** and have it print **"day.month.year"**
20. Now run the project

Stop Here Ask the TAs to evaluate your code's performance to get graded for this portion of the lab.(1 mark)

4 Expanding on Our Class

Having a class that stores the date can be very useful, but we can add more to our class! you will now create a class **"Stock"**. That will pass the test code *testStock()*.

1. At this time it is recommended that you **switch from navigator to driver and vise-versa.**
2. Go to your main method in **"InLab3"**
3. add the line *testStock();* to the method. this will now run the test code for our new class.

With that done lets get to building our stock class. It is a class that describes the daily value of a stock for each of the hours the market is open. It will have all of the attributes from our previous class (day, month, year) and two new ones. These new attributes will be the stock name and a list of 7 values of the stock through the day.

1. First create a class and name it **"Stock"**
2. Copy over the contents of **"Activity"** to **"Stock"** and remove the constructors
3. With the other attributes add two more.
   (a) **private String name;**
   (b) **private int[] values = new int[7];**
4. Add a new default constructor **public Stock(){}**
5. As before set the date to 1s, but then set the name to the empty String "", and all of the elements of values to 0.
6. At this time it is recommended that you **switch from navigator to driver and vise-versa.**

7. Now add a constructor that will initialize all the attributes
8. add the constructor **public Stock(int dayIn, int monthIn, int yearIn, String nameIn, int[] valIn){}**
9. Check that the date is valid, and that valIn is of length 7 with no negative number. We cant have a stock that is worth less than nothing.
10. If the arguments are all valid create set all the attributes accordingly. otherwise set all attributes to the default and print an error message.
11. Now we must add accessors for our new attributes
12. First create the method **public String getName(){}** and have it return the name
13. Then create **public int getValue(int ind){}**
14. This method must return the value at int or print an error statement and return -1 if an out of bounds index is used.
15. Now we will add mutators for our new attributes
    (a) **public void setName(String nameIn){}** changes the name atribute
    (b) **public void setValue(int val, int ind){}** changes the value at index ind and prints an error statement if val <0 or index is not valid
16. At this time it is recommended that you **switch from navigator to driver and vise-versa.**
17. Add a print method **public void printStock(){}** that displays all attributes of the class in a print statement.
18. lastly add a method **public int getMean(){}** that returns the average value of the stock on that day. HINT: consider loops or built in functions
19. Now you are done the class Stock run the program

Stop Here Ask the TAs to evaluate your code's performance to get graded for this portion of the lab.(1 Mark)

End of lab