CS352: Advanced Software Engineering

Lab 2

Java Basics

Cairo University, Faculty of
Computers and Artificial Intelligence

This lab is a student-driven lab. The student will take responsibility of his own learning and go over the material to fully understand it. Seek external resources on OOP and Java as needed.

Objectives

1- **Very Important:** Learning OOP concepts. 2- Learning Java programming essentials

Part 1 – Classes and Objects

Important Concepts

Class: A class is a description of an aggregated set of data attributes and operations that represent a concept. For example, the class Car has the attributes: model, make, year, color and has the operations: speed, brake, turn right, turn left, etc. A class is a user-defined type.

Object: An object is an instance of a class. It is the actual implementation of one item that belongs to a certain class. For example, Ali's car is an instance of the class car and it is Nissan, Sunny, 2008, white, etc. and it can perform all the operations that are defined for the class Car.

Object Oriented Programming: Object-oriented programming is developing programs by creating objects and defining how they interact with each other rather than creating functions / methods.

Data Field / Attribute: It is a variable that defines one of the characteristics of objects that belong to a certain class. An attribute will have different values for different objects belonging to the same class, e.g., Ali's car color is white, but Mohammad's is red and Sayed's is metallic.

Method / Operation: Is a function that can be preformed on the objects of a certain class.

Reference Variable: This is a variable that points to an object. The variable does not have any useful data in itself, but it contains the address of the actual object in memory.

UML: A visual language used represent and model object oriented programs.

Access Specifiers: These are keywords that define who can access the class members (data fields and methods). The two important ones are **`private`** which means that only methods members of the same class can access this private class member and **`public`** which means that all methods in any other class can access this class member

Rectangle
`- length : double` `- width : double`

`+ setWidth(w : double) : void`
`+ setLength(len : double): void`
`+ getWidth() : double`
`+ getLength() : double`
`+ getArea() : double`

CS352: Advanced Software Engineering

Lab 2

Java Basics

Cairo University, Faculty of
Computers and Artificial Intelligence

Java Coding Style: These are a set of rules that make the code readable and easier to maintain. See the attached document.

Class Representation in UML: is as show in the figure. – is private and + is public.

Java Doc: Java Doc is a utility with Java that allows the developer to add comments in a certain format and generate Html documentation for his classes automatically from these comments. A summary of Java Doc commenting style is shown below.

Student Activities

Activity 1:

Create a class that represents an alarm clock that displays time in 12 hours format. Create a demo to show how objects of this class work. Follow the same steps in activity 1. The class will have these attributes:

- `hours`
- `minutes`
- `seconds`
- Any other attributes you like to add

It will have the following methods:

- `Clock (int h, int m, int s)`
- `setTime (int h, int m, int s)`
- `incrementSeconds (int addedSeconds)`
- `incrementMinutes (int addedMinutes)`
- `incrementHours (int addedHours)`
- `toString ()`
- Any other methods you like to add

CS352: Advanced Software Engineering

Lab 2

Java Basics

Cairo University, Faculty of
Computers and Artificial Intelligence

Part 2 –Aggregation, Inheritance, Abstract Classes, and Interfaces

Important Concepts

Java Packages: Are ready made libraries of classes provided by Java for the developer. They make the Java Application Programmable Interface (API). To invoke a package, use **import** statement.

* You must use proper coding style and generate documentation for your programs.

Coding Style is a set of rules for writing readable and organized code that is easy to understand and maintain.

Aggregation happens when a class owns an object of another class as a data field in this class.

Inheritance is deriving a new class from an existing class using **`extends`** keyword. The new class will have all the methods and data fields of the original class plus its own ones. The original class is called **parent** or **super class**. The new one is called **child** or **base class**.

Overriding happens when a child class re-implements a method that was already inherited from the parent class. Do not mix **overriding** with **overloading**.

`Object` is the parent (or grandparent) class of every class in Java. It give its children a basic form of some methods like **`toString ()`**, **`equals ()`**, **`clone ()`** and **`getClass ()`**.

Access modifiers are Java keywords that define the level of access of each data field or method. There are four of them: **`private`**, **`protected`**, **`public`** and default or package access. Read about the differences between them.

An **Abstract Class** is an incomplete class that is missing the body of one or more of its methods. Or it can be a complete class but has the keyword abstract in its header. An abstract class cannot be instantiated. It is only used as a parent class for children classes that extend it and provide full implementation for all incomplete methods.

CS352: Advanced Software Engineering

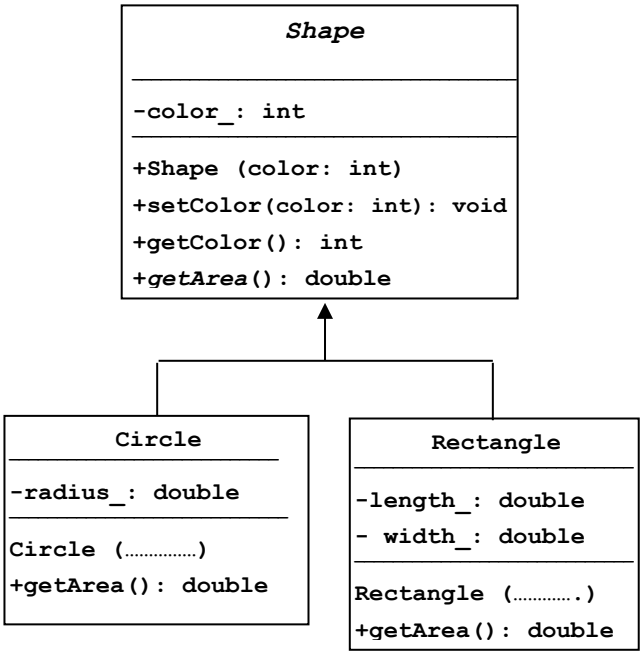
Lab 2

Java Basics

Cairo University, Faculty of
Computers and Artificial Intelligence

An **Interface** is the extreme case of an abstract class. It has only method headers and not method bodies. It cannot not have data fields. Basically, an interface is a way for defining how a certain behavior should be implemented by unrelated classes whose developers like to add to them this behavior.

Polymorphism means having many forms. Java supports a few types of polymorphism. One of them is overriding. Also, a reference variable is polymorphic because it can reference objects of types different from its own, as long as those types are subclasses of its type. It is the object's type, rather than the reference type that determines which method is called at runtime. The same applies to interfaces, where you can create a reference variable whose type is an interface. His variable can point to an object from a class that implements the interface.



Student Activities

Activity 2: Interfaces and Abstract Classes

Objective: to understand the differences between abstract classes and interfaces.

Suppose you have the following code:

```
abstract class Time {
    public abstract int getMinutes();
}

class Days extends Time {
      private int days;
      public Days(int days) {
            this.days = days;
      }
      public int getMinutes() {
            return days * 24 * 60;
      }
}

class HoursMinutes extends Time {
      private int hours;
      private int minutes;
```

Cairo University, Faculty of Computers and Artificial Intelligence


```
        public HoursMinutes(int hours, int minutes) {
                this.hours = hours;
                this.minutes = minutes;
        }
        public int getMinutes() {
                return hours * 60 + minutes;
        }
}
```

And also the following code that is similar to the previous one but using interface instead of abstract class.

```
interface Time {
        int getMinutes();
}

class Days implements Time {
        private final int days;
        public Days(int days) {
                this.days = days;
        }
        public int getMinutes() {
                return days * 24 * 60;
        }
}

class HoursMinutes implements Time {
        private final int hours;
        private final int minutes;
        public HoursMinutes(int hours, int minutes) {
                this.hours = hours;
                this.minutes = minutes;
        }
        public int getMinutes() {
                return hours * 60 + minutes;
        }
}

public class Demo {
        public static void main(String args[]) {
          Time t1 = new Days(10);
          Time t2 = new HoursMinutes(15, 59);
          System.out.println(t1.getMinutes());
          System.out.println(t2.getMinutes());
      }

}
```

Cairo University, Faculty of
Computers and Artificial Intelligence

Now what's the difference between using abstract classes and interfaces in the example above if you want to add to Time a method: `public int getSeconds();`?

Graded Lab Task

Apply the Java concepts that you learned (OOP, polymorphism, interfaces, packaging, …) to design your own ***Geometric*** library architecture and implement this package using the java programming language. This library handle drawing **2D shapes** with different styles. The designed library must handle the following functions:

- Draw ***Ellipse, Rectangle, Square***
- The shapes are moveable. You can use methods such as methods goUp(), goDown(), goLeft() and goRight() in your design for each point/shape.
- Help the user to calculate the area and the perimeter for those shapes.
- Make your shapes resizable by factors (e.g. 50%, 200%, …)
- The library is extensible to 3D shapes. Apply the basic skeleton code that can be easily
  implemented later.

Sketch a class diagram, and write the corresponding source code for that application using Java programming language. You can keep the bodies of the moving and drawing operations empty for the time limit of the lab.