National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS 212: Object Oriented Programming

Class: BESE-7AB

Lab 07: Polymorphism

Date: April 14, 2017

Instructor:

Dr. Muhammad Muneeb Ullah

Learning Objectives

The learning objective of this lab is to understand and practice the concept of polymorphism, a very powerful feature of OOP which helps in code extensibility.

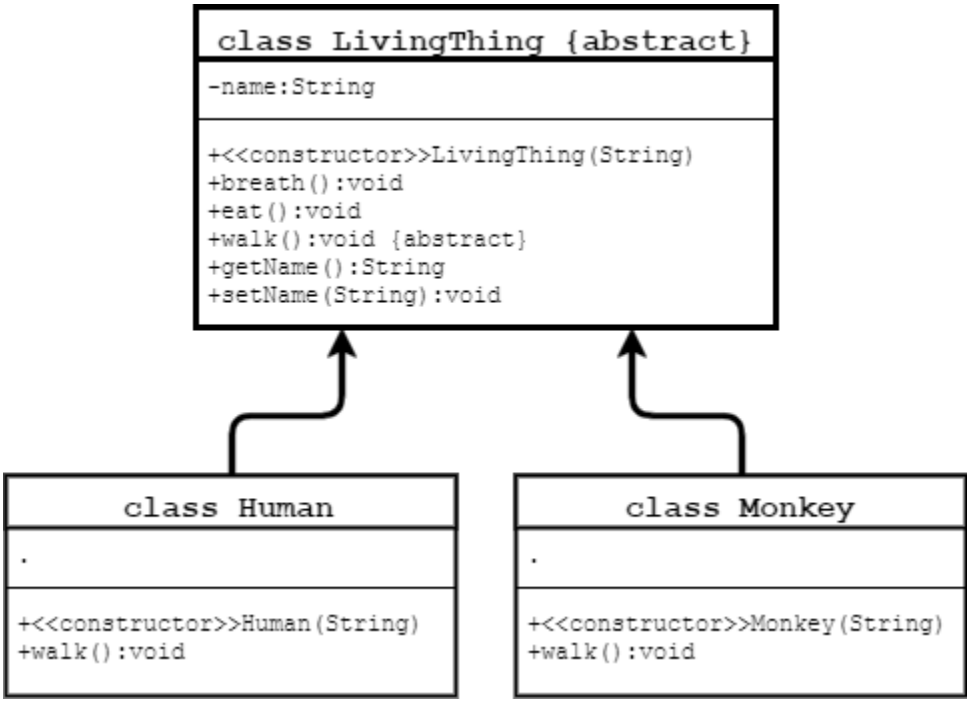
Warm-up.

Consider there is a hair dresser, a surgeon and an actor. If you ask each of them to 'cut': the hair dresser starts cutting hair, the surgeon makes an incision, and the actor stops acting out the scene. This is polymorphism; same name but different behavior.

Now your task is to give a simple real life example of polymorphism.

Task #1.

Write an abstract class LivingThing.java followed by two concrete classes, Human.java and Monkey.java, extending the abstract class.



Once done defining classes use the following client class to test it .

```
package myabstractclassproject;

public class Main {
	public static void main( String[] args) {
		// Create Human object instance
		// and assign it to Human type.
		Human human1 = new Human( "Will Rodman");
		human1.walk();

		// Create Human object instance
		// and assign it to LivingThing type.
		LivingThing livingthing1 = human1;
		livingthing1.walk();

		// Create a Monkey object instance
		// and assign it to LivingThing type.
		LivingThing livingthing2 = new Monkey( "Caesar");
		livingthing2.walk();

		// Display data from human1 and livingthing1.
		// Observe that they refer to the same object instance.
		System.out.println( "human1.getName() = " + human1.getName());
		System.out.println( "livingthing1.getName() = " +
livingthing1.getName());

		// Check of object instance that is referred by x and
		// y is the same object instance.
		boolean b1 = ( human1 == livingthing1);
		System.out.println( "Do human1 and livingthing1 point to the
same object instance? " + b1);
	}
}
```

Running the test should result in the following output .

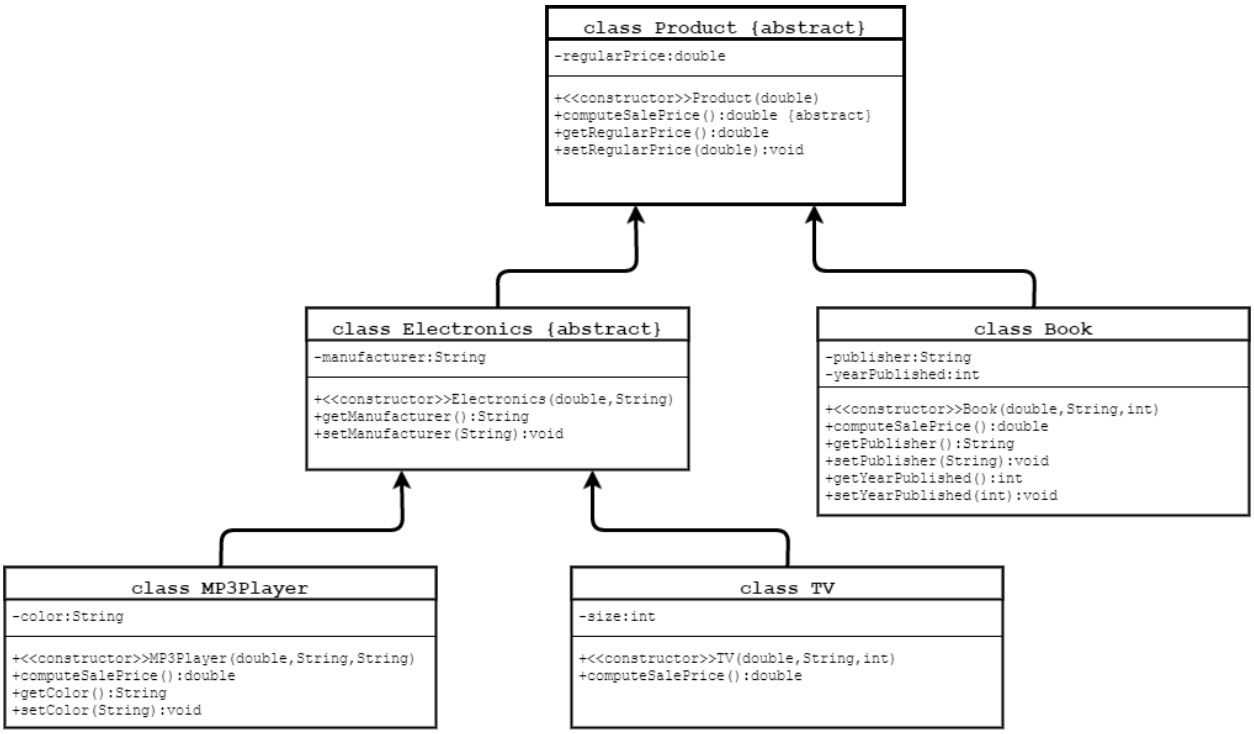
```
Human Will Rodman walks...
Human Will Rodman walks...
Monkey Caesar also walks...
human1.getName() = Will Rodman
livingthing1.getName() = Will Rodman
Do human1 and livingthing1 point to the same object instance? true
```

Bonus. What happens when you create a LivingThing object in the Main class? For example, using the statement,

```
LivingThing z = new LivingThing();
```

Task #2.

Your task is to write `MyOnlineShop` program by referring to the UML class diagram below.



Once done with the class definitions use the following tester class to confirm its working.

```
package myonlineshop;

public class Main {

    public static void main(String[] args) {

        // Declare and create Product array of size 5
        Product[] pa = new Product[5];

        // Create object instances and assign them to
        // the type of Product.
        pa[0] = new TV( 1000, "Samsung", 30);
        pa[1] = new TV( 2000, "Sony", 50);
        pa[2] = new MP3Player( 250, "Apple", "blue");
        pa[3] = new Book( 34, "Sun press", 1992);
        pa[4] = new Book( 15, "Korea press", 1986);

        // Compute total regular price and total
        // sale price.
        double totalRegularPrice = 0;
        double totalSalePrice    = 0;
```

```
        for (int i=0; i<pa.length; i++){

            // Call a method of the super class to get
            // the regular price.
            totalRegularPrice += pa[i].getRegularPrice();

            // Since the sale price is computed differently
            // depending on the product type, overriding (implementation)
            // method of the object instance of the sub-class
            // gets invoked.  This is runtime polymorphic
            // behavior.
            totalSalePrice += pa[i].computeSalePrice();

            System.out.println("Item number " + i +
                    ": Type = "          + pa[i].getClass().getName() +
                    ", Regular price = " + pa[i].getRegularPrice() +
                    ", Sale price = "    + pa[i].computeSalePrice());
        }

        System.out.println("totalRegularPrice = " + totalRegularPrice);
        System.out.println("totalSalePrice = "    + totalSalePrice);
    }
}
```

The test should result in the following output.

```
Item number 0: Type = myonlineshop.TV, Regular price = 1000.0, Sale price = 800.0
Item number 1: Type = myonlineshop.TV, Regular price = 2000.0, Sale price = 1600.0
Item number 2: Type = myonlineshop.MP3Player, Regular price = 250.0, Sale price
= 225.0
Item number 3: Type = myonlineshop.Book, Regular price = 34.0, Sale price = 17.0
Item number 4: Type = myonlineshop.Book, Regular price = 15.0, Sale price = 7.5
totalRegularPrice = 3299.0
totalSalePrice = 2649.5
```

Task #3 (Optional: for those students who would like to practice even more!).

(***CarbonFootprint*** *Interface: Polymorphism*) Using interfaces, as you learned in this chapter, you can specify similar behaviors for possibly disparate classes. Governments and companies worldwide are becoming increasingly concerned with carbon footprints (annual releases of carbon dioxide into the atmosphere) from buildings burning various types of fuels for heat, vehicles burning fuels for power, and the like. Many scientists blame these greenhouse gases for the phenomenon called global warming. Create three small classes unrelated by inheritance—classes `Building`, `Car` and `Bicycle`. Give each class some unique appropriate attributes and behaviors that it does not have in common with other classes. Write an interface `CarbonFootprint` with a `getCarbonFootprint` method. Have each of your classes implement that interface, so that its `getCarbonFootprint` method calculates an appropriate carbon footprint for that class (check out a few websites that explain how to calculate carbon footprints). Write an application that creates objects of each of the three classes, places references to those objects in

`ArrayList<CarbonFootprint>`, then iterates through the Array-List, polymorphically invoking each object's `getCarbonFootprint` method. For each object, print some identifying information and the object's carbon footprint.

Hand in

Hand in the source code from this lab at the appropriate location on the LMS system. You should hand in a single compressed/archived file named Lab_7_<Your CMS_ID. Your_NAME >.zip (without angle brackets) that contains ONLY the following files.

1) All completed java source files representing the work accomplished for this lab: LivingThing.java; Human.java; Monkey.java; Product.java; Electronics.java; MP3Player.java; TV.java; Book.java; (**Optional**) Building.java, Car.java and Bicycle.java and CarbonFootprint.java. The files should contain author in the comments at the top.

2) A plain text file named **README.TXT** that includes a) author information at the beginning, b) a brief explanation of the lab, and c) any comments, or suggestions.

To Receive Credit

1. By showing up on time for lab, working on the lab solution, and staying to the end of the class period, only then you can receive full credit for the lab assignment.

2. Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.

3. The lab time is not intended as free time for working on your programming/other assignments. Only if you have completely solved the lab assignment, including all challenges, and have had your work checked off for completeness by your TA/Lab Engineer should you begin the programming/other assignments.