Practical Exercise 1 – Basic Java and Object-oriented Programming

Overall Objective

By the end of this practical, you should:

- understand and be able to use the integrated development environment (IDE) for Java
- be able to create, edit, and run your Java programs

Background

You will need to know:

1. data types, values and variables
2. input/output
3. selection and iteration
4. methods
5. arrays
6. strings
7. objects and classes
8. inheritance and polymorphism
9. exception handlings
10. abstract classes and interfaces

Description

Part 1: Discussion

1 Type all of the following code, save the program as specified.
Compile the file. Why did it fail?
Fix it, recompile and run the program

(a) **saved as Test1.java**

```
public class Testing1 {
   public static void main(String args) {
      System.out.println("What's wrong with this program?");
   }
}
```

(b) **saved as Test2.java**

```
public class Test2 {
   public void main(String[] args) {
      System.out.println("What's wrong with this program?")
   }
}
```

2 What problem arises in compiling the following program?

```
class A {
   public A(int x) {}
}

class B extends A {
   public B() {}
}
```

3 Show the output of the following programs:

a)
```
public class Test {
   public static void main(String args[]) {
      A a = new A(3);
   }
}

class A extends B {
   public A(int n) {
      System.out.println("A's constructor is invoked.");
   }
}

class B {
   public B() {
      System.out.println("B's constructor is invoked.");
   }
}
```

b)
```
class Base {
   public void Base() {
      System.out.println("Base");
   }
}

public class In extends Base {
   public static void main(String argv[]) {
      In i = new In();
   }
}
```

c)
```
class Base {
   void sub() {
      System.out.println("Base");
   }
   void sub(int p) {
      System.out.println(20);
   }
}

class Severn extends Base {
   public static void main(String argv[]) {
      Severn s = new Severn();
      s.sub(15);
   }

   void sub(){
      System.out.println("Severn");
   }
}
```

4 Write a class named Account to model accounts. The UML diagram for the class is shown below.

Account
-balance : Double -id : Integer -annualInterestRate : Double
+Account() +Account(" id : Integer, " balance : Double, " annualInterest : Double) +deposit(" amount : Double) : void +withdraw(" amount : Double) : void +getBalance() : Double +getAnnualInterest() : Double

Write a client program to test the Account class. In the client program, create an Account instance with an account id of 1122, a balance of 20000, and an annual interest rate of 4.5%. Use the withdraw method to withdraw $2500, use the deposit method to deposit $3000, and print the balance and the monthly interest.

5 Declare Two classes according to the following instructions:

(a) Create a base class named `Rectangle`

- Declare 2 private object attributes: `length` and `width`, that contain in class `Rectangle`.
- Declare a public constructor that takes 2 parameters `(init_length, init_width)` that populates the `length` and `width` attributes.
- Declare a public method `area()` that returns that surface area of the rectangle.

(b) From the base class `Rectangle`, create a subclass named `Box`.

- Declare an additional object attribute named: `depth`
- Declare a constructor for the derived `Box` class that takes 3 parameters `(init_length, init_width, init_depth)` that populates the `length`, `width` and `depth` attributes.
- Declare an override method named `area()` that returns that surface area of the box.
- Declare a method named `volume()` that returns that volume of the box.

(Formula for computing:
area of a rectangle, area = length × width
area of a box , area = 6 × length × width
volume of a box, volume = length × width × height)

(c) Write a client program to test both `Rectangle` and `Box` classes.

- Create an instance that stores a 10 × 12 rectangle. Print the area of the rectangle.
- Create an instance that stores a 10 × 12 × 8 box. Print the area and volume of the box.

6 Given that :

```
public abstract class Lecturer {
   private String name;
   private int id;

   public Lecturer (String name, int id) {
      this.name = name;
      this.id = id;
   }

   public void setName(String name) {
      this.name = name;
   }

   public String getName() {
      return name;
   }

   public void setId(int id) {
      this.id = id;
   }

   public int getId() {
      return id;
   }

   public String toString() {
      return getName( ) + " " + getId( );
   }

   public abstract double salary();
}
```

From the base class `Lecturer`, create a subclass named `PartTimeLecturer` with the following requirements.

– Declare an additional `double` typed attribute named: `teachingHour`.
– Declare 3 constructors for the derived `PartTimeLecturer` class
  - The first constructor should be a default constructor.
  - The second constructor should take one parameter that populates the `id` attribute.
  - The third constructor should take three parameters that populate the `name`, `id` and `teachingHour` attributes.
- Declare a mutator method named `setTeachingHour( )` that sets a lecturer's teaching hours.
- Declare an accesor method named `getTechingHour( )` that returns an lecturer's teaching hours.
- Declare an override method named `toString( )` that returns a lecturer's `name`, `id` and `teachingHour`.
- Implement the abstract method named `salary( )` declared in class `Lecturer` that returns the part time lecturer's salary corresponding to the teaching hours. The salary can be determined using the formula:

  $salary = teachingHour \times 100.00;$

Write a client program to test both `Lecturer` and `PartTimeLecturer` classes.

7 Given

```
public class Vehicle {}

public interface VehicleMethods {
   public double getWeight();
   public int getSpeed();
}
```

(i) Write an abstract class `LandVehicle` that inherits from the class `Vehicle` and implements the `VehicleMethods` interface.

(ii) Write a subclass `Car` that inherits from `LandVehicle` and implements the two methods declared in the interface `VehicleMethods`. Assume there are no other attributes and methods in both class declarations. The constructors can also be omitted.

Assessment

This practical exercise will not be assessed. However, you are encouraged to finish all the exercises so that you can gain more familiarization and understanding on Java programming language.