Exercises 05

1. [HW] Given the code below. Discuss the output and explain why it is so?

```
class X {
    public X() {
        System.out.println("In constructor X.");
    }
}

class Y extends X {
    public Y() {
        System.out.println("In constructor Y.");
    }
}

public class Main {
    public static void main(String[] args) {
        Y y = new Y();
    }
}
```

2. [EX] Given the code below. Discuss the output and explain why it is so?

```
class A {
    static { System.out.println("In static init block of A"); }

    public A() { System.out.println("In constructor A.");     }

    { System.out.println("In instance init block of A"); }
}

class B extends A {
    static { System.out.println("In static init block of B");     }

    public B() { System.out.println("In constructor B.");     }

    { System.out.println("In instance init block of B");     }
}

public class Main2 {
    public static void main(String[] args) {
        B b = new B();

    }
}
```

3. [EX] Given the following code sample. Predict the output and explain why so?

```
public class OverrideDemo {
    public static void main(String[] args) {
        Child child = new Child();

        System.out.println(child.calcValue(5));
        System.out.println(child.calcValue(5, 10));
    }
}

class Parent {
    public int calcValue(int a) {
        System.out.println("Super");
        return a * 2;
    }
}

class Child extends Parent {
    public int calcValue(int a, int b) {
        System.out.println("Subclass");
        return (a + b) * 2;
    }
}
```

4. [EX] Given the following code sample. Predict the output and explain why so?

```
class A {
    static { System.out.println("AS"); }

    { System.out.println("A"); }

    public A() { System.out.println("AC"); }

    public A(int x) { System.out.println(x + " AC"); }
}
```

```
class GeneralType extends A {
    static { System.out.println("GTS"); }

    { System.out.println("GT"); }

    public GeneralType() {
        super(5);
        System.out.println("GTC");
    }
}
```

5. [HW] Discuss the types of inheritance. Give examples.
    - Single
    - Multilevel
    - Hierarchical
    - Multiple
    - Hybrid

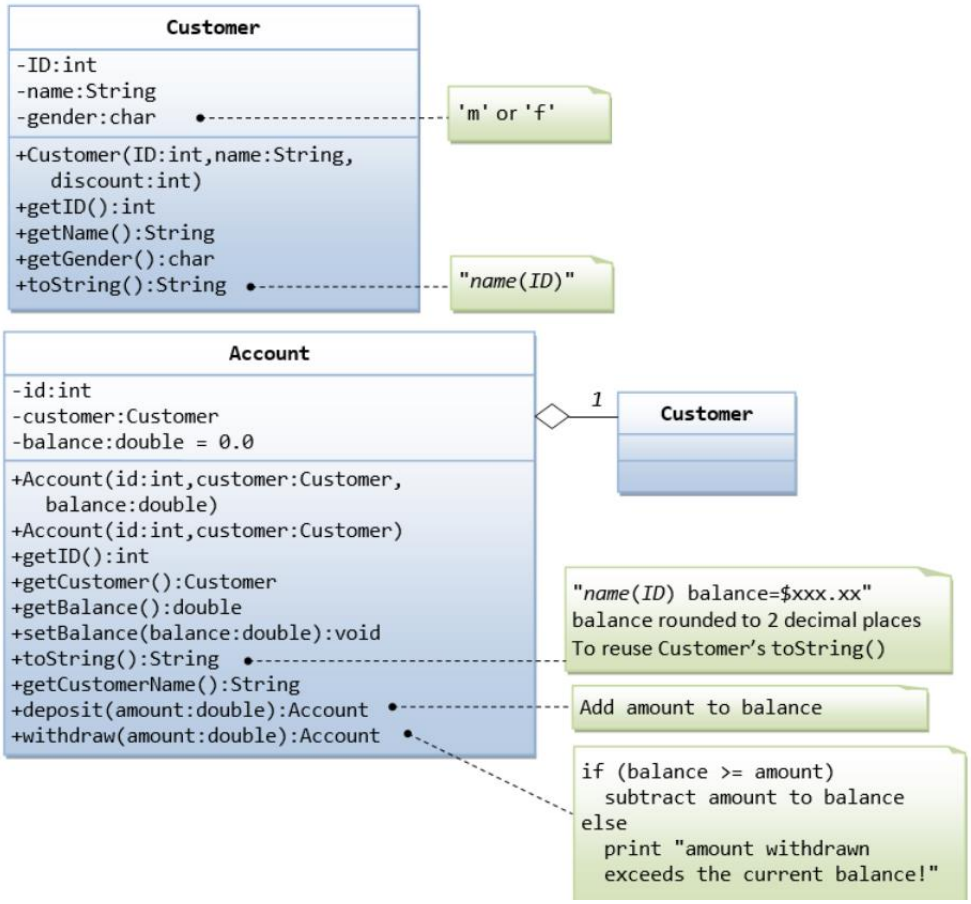
6. [EX] What is happening in the following code? What methods of ClassB are overriding, hiding superclass ClassB methods?

```
class ClassA {
    public void methodOne(int i) {}
    public void methodTwo(int i) { }
    public static void methodThree(int i) {}
    public static void methodFour(int i) {}
}
class ClassB extends ClassA {
    public static void methodOne(int i) { }
    public void methodTwo(int i) {}
    public void methodThree(int i) {}
    public static void methodFour(int i) {}
}
```

7. [PW] Create a class MathDemo which has the following methods:
    a. int min(int a, int b)
    b. int max(int a, int b)
    c. int sum(int[] args)
    d. float mean(int[] args)
    e. int factorial(int n)
    f. What would be the advantage of defining these methods as static?

8. [EX] Inheritance
    a. Define a class **Person** with attributes **firstName**, **lastName**, **gender**.
        i. Provide **constructor**(s) and **getter/setter** methods.
        ii. Provide **toString**() method.
        iii. Provide **equals(Person p)** method.
    b. Define a class **Teacher** which extends **Person**. It also has **department** and **courses** attributes.
        i. Provide **constructor**(s) and **getter**/**setter** methods.
        ii. Provide **toString**() method.
        iii. Provide **equals(Teacher t)** method.
    c. Define a class **Student** which extends **Person**. It also has a **studentId** attribute.
        i. Provide **constructor**(s) and **getter**/**setter** methods.
        ii. Provide **toString**() method.
        iii. Provide **equals(Student s)** method.
    d. Define a class **PhdStudent** which extends **Student**. It also has **department** and **courses** attributes.
        i. Provide **constructor**(s) and **getter**/**setter** methods.
        ii. Provide **toString**() method.
        iii. Provide **equals(PhdStudent pStud)** method.
    e. Create a Main class to test all the above.

9. [PW] Consider the following class diagrams. Try to implement the classes and test them.



10. [PW] Take the Point and Circle classes designed and implemented in previous weeks. Override the toString() and equals(Object obj) methods of these classes.
    a. **Note**: Previously we have not talked about overriding them.

11. [PW] Create a **Rectangle** class and override the *equals()* method of the **Object** class such that, if the width and height of the provided rectangle are the same as the current object, then return true.

```
class Rectangle {
    int width, height;

    public Rectangle(int w, int h) {
        width = w;
        height = h;
    }

    public boolean equals(Object obj) {
        Rectangle rect = (Rectangle) obj;
        // your code here
    }
}
```

a. Test your code with the following statements:

```
Rectangle r1 = new Rectangle(5,10);
Rectangle r2 = new Rectangle(15,10);
Rectangle r3 = new Rectangle(5,10);

System.out.println(r1.equals(r2));
System.out.println(r1.equals(r3));
```

12. [PW] Now create a **Square** class that extends **Rectangle** and has a constructor with one argument – side. Test your code as given in the following example to see the power of OOP!

```
Object o1 = new Rectangle(5,10);
Object o2 = new Rectangle(15,15);
Object o3 = new Square(15);

System.out.println("Objects are identical: " + o1.equals(o2));
System.out.println("Objects are identical: " + o1.equals(o3));
System.out.println("Objects are identical: " + o2.equals(o3));
```