Session 03
Classes and Objects

A. REQUIREMENTS

I. Write a class named Shape, which contains:

- 2 instance variables color(String) and filled(boolean).
- Two constructors:
  - No-argument constructor that initializes the color to "green" and filled to true
  - 2-arguments constructor that initializes the color and filled to the given values.
- Write **toString()** method that returns "A Shape with color of xxx and filled/Not filled".

II. Write the Circle class derived from Shape contains:

- An instance variable radius(double).
- Two constructors:
  - The no-arg constructor initializes the radius to 1.0.
  - A constructor with 3-args: color, filled, radius that initializes the radius to the given values and call super constructor to initializes the color and filled to the given values.
- Methods getArea() and getPerimeter().

Write a Test class to test Circle class.

III. Create packages named package1, package2.
Write a class named Woman (place in package2) that implements the interface named Human (place in package1). Human interface having:

- 2 constants: legNum (int) = 2, eyeNum (int) = 2
- 2 methods: void run(double speed), void sleep(int time, String type)

  Write a Test class to test Woman class

B. STEPS BY STEPS

I. Shape class

Shape.java

```
public class Shape {
    String color;
    boolean filled;

    Shape(){
        color = "White";
        filled = false;
    }

    public Shape(String color, boolean filled) {
        this.color = color;
        this.filled = filled;
    }

    @Override
    public String toString(){
        return "A Shape with color of " + color + " and " + (filled==true? "filled":
                                                                          "not filled");
    }

}
```

II. Circle class

Circle.java

```
public class Circle extends Shape {
    double radius;

    public Circle() {
        radius = 1.0;
    }

    public Circle(double radius, String color, boolean filled) {
        super(color, filled);
        this.radius = radius;
    }

    double getParimeter() {
        return radius*Math.PI*2;
    }

    double getArea() {
        return radius*Math.PI*Math.PI;
    }
}
```

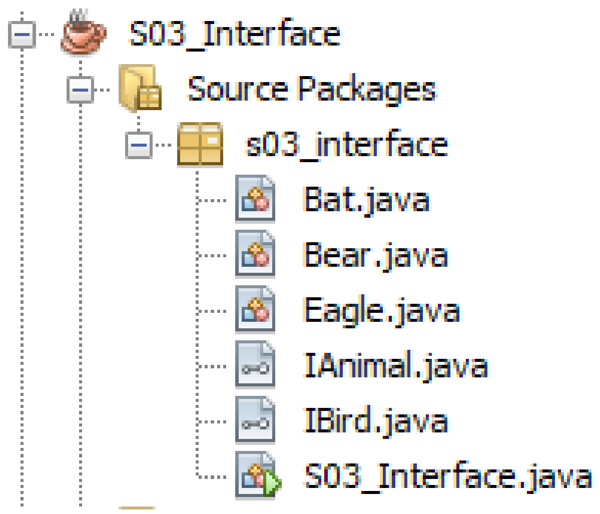
III. Test class

Test.java

```
public class Test {
    public static void main(String args[]) {
        Circle obj = new Circle();
        System.out.println("The first circle: - - - - - -");
        System.out.println("Parimeter: " + obj.getParimeter());
        System.out.println("Area:" + obj.getArea());

        System.out.println("The second circle: - - - - - -");
        Circle obj2 = new Circle(2.0, "Yellow", true);
        System.out.println("Perimeter: " + obj2.getParimeter());
        System.out.println("Area:" + obj2.getArea());
    }
}
```

IV. Interface


S03_Interface
Source Packages
s03_interface
Bat.java
Bear.java
Eagle.java
IAnimal.java
IBird.java
S03_Interface.java

IBird.java

```
package s03_interface;

/**
 *
 * @author KhanhVH@fe.edu.vn
 */
public interface IBird {
    void eat();
    void fly();
}
```

IAnimal.java

```java
package s03_interface;

/**
 *
 * @author KhanhVH@fe.edu.vn
 */
public interface IAnimal {
    void eat();
    void run();
}
```

Bear.java

```java
package s03_interface;

/**
 *
 * @author KhanhVH@fe.edu.vn
 */
public class Bear implements IAnimal {
    private String name;

    /**
     * Bear constructor
     * @param name is the name of Bear
     */
    public Bear(String name) {
        this.name = name;
        System.out.println("Bear named " + name + " was borned");
    }

    /**
     * override IAnimal's eat method
     */
    @Override
    public void eat() {
        System.out.println(name + " eats fishes");
    }

    /**
     * override IAnimal's run method
     */
    @Override
    public void run() {
        System.out.println(name + " is running");
    }
}
```

Eagle.java

```java
package s03_interface;

/**
 *
 * @author KhanhVH@fe.edu.vn
 */
public class Eagle implements IBird {
    private String name;

    /**
     * Eagle constructor
     * @param name is the name of Eagle
     */
    public Eagle(String name) {
        this.name = name;
        System.out.println("Eagle named " + name + " was borned");
    }

    /**
     * override IBird's eat method
     */
    @Override
    public void eat() {
        System.out.println(name + " eats doves");
    }

    /**
     * override IBird's fly method
     */
    @Override
    public void fly() {
        System.out.println(name + " is flying");
    }
}
```

Bat.java

```java
package s03_interface;

/**
 *
 * @author KhanhVH@fe.edu.vn
 */
public class Bat implements IAnimal, IBird {
    private String name;

    /**
     * Bat constructor
     * @param name is the name of Bat
     */
    public Bat(String name) {
        this.name = name;
        System.out.println("Bat named " + name + " was borned");
    }
    /**
     * override IAnimal and IBird eat method
     */
    @Override
    public void eat() {
        System.out.println(name + " eats mosquito");
    }

    /**
     * override IAnimal's run method
     */
    @Override
    public void run() {
        System.out.println(name + " is running");
    }

    /**
     * override IBird's fly method
     */
    @Override
    public void fly() {
        System.out.println(name + " is flying");
    }
}
```

S03_Interface.java

```
package s03_interface;
/**
 *
 * @author KhanhVH@fe.edu.vn
 */
public class S03_Interface {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        //use instance of Bat that implement
        //from both interfaces IAnimal and IBird
        Bat Sarah = new Bat("Sarah");
        Sarah.run();
        Sarah.fly();
        Sarah.eat();

        //use instance of Bear that implement from interface IAnimal
        IAnimal Jack = new Bear("Jack");
        Jack.run();
        Jack.eat();

        //use instance of Eagle that implement from interface IBird
        IBird Joe = new Eagle("Joe");
        Joe.fly();
        Joe.eat();
    }
}
```

C. MORE EXCERCISES

1. Create a superclass named Parent having 3 instance variables: number, name and salary

- Add 2 constructors:
  - No-arg that initializes default value to instance variables
  - 3-args that initializes given values to instance variables
- Add display() method to display object's information
- Create a subclass named Child derived from Parent
  - No-args call constructor from super class
  - Add main method and create 2 object from Child class (with different constructors)
  - Call display method of 2 object
- Add method named checkNum() to return odd or even number in Parent class
- Add method named checkNum() in Child class to override Parent's method to check positive number or not

2. You must create a class named Fraction to calculate some fraction operators of two fractions:

a) This class have three constructors

   i. The first constructor doesn't have any parameter. The numerator will be 0 and denominator will be 1.

   ii. The second constructor has one integer parameter so the numerator will equal with the parameter and denominator will be 1.

   iii. The third constructor has two integer parameters so the numerator will equal with the first parameter and denominator will equal with the second parameter.

b) Writing "set method" and "get method" for numerator and denominator and validate the value that user set for denominator when they use set method.

c) Writing a ToString method to display this fraction.

d) This class have two methods called "add"

   i. The first function has one parameter that typed Fraction. This function will calculate the sum of current fraction and the fraction provided by the parameter.

   ii. The second function has one integer parameter. This function will calculate the sum of current fraction and the integer parameter.

e) This class have two methods called “subtract”

   i. The first function has one parameter that typed Fraction. This function will calculate the subtraction of current fraction and the fraction provided by the parameter.

   ii. The second function has one integer parameter. This function will calculate the subtraction of current fraction and the integer parameter.

f) This class have two methods called “multiply”

   i. The first function has one parameter that typed Fraction. This function will calculate the multiplication of current fraction and the fraction provided by the parameter.

   ii. The second function has one integer parameter. This function will calculate the multiplication of current fraction and the integer parameter.

g) This class have two methods called “divide”

   i. The first function has one parameter that typed Fraction. This function will calculate the division of current fraction and the fraction provided by the parameter.

   ii. The second function has one integer parameter. This function will calculate the division of current fraction and the integer parameter.