Lab: Polymorphism

This document defines the lab for "Java OOP" course @ Software University. Please submit your solutions (source code) of all below described problems in Judge.

1. Math Operation

Create a class **MathOperation**, which should have method **add()**. Method **add()** have to be invoked with **two, three** or **four Integers.**

You should be able to use the class like this:

Main.java

```
public static void main(String[] args) throws IOException {
    MathOperation math = new MathOperation();
    System.out.println(math.add(2, 2));
    System.out.println(math.add(3, 3, 3));
    System.out.println(math.add(4, 4, 4, 4));
}
```

Examples

Input	Output
	4 9 16

Solution

Class **MathOperation** should look like this:

```
public class MathOperation {

    public int add(int a, int b) {
        return a + b;
    }
    public int add(int a, int b, int c) {
        return a + b + c;
    }
    public int add(int a, int b, int c, int d) {
        return a + b + c + d;
    }

}
```

2. Shapes

Create class hierarchy, starting with abstract class **Shape**:

- **Fields:**
  - **perimeter : Double**
  - **area : Double**
- **Encapsulation for this fields**

- **Abstract methods:**
  - **`calculatePerimeter()`**
  - **`calculateArea()`**

Extend Shape class with two children:

- **Rectangle**
- **Circle**

Each of them needs to have:

- **Fields:**

  For Rectangle
  - **height : Double**
  - **width : Double**

  For Circle
  - **radius : Double**
- **Encapsulation for this fields**
- **Public constructor**
- **Concrete methods for calculations (perimeter and area)**

3. Animals

Create a class **`Animal`**, which holds two fields:

- **`name`**: **String**
- **`favouriteFood`**: **String**

Animal has one abstract method **`explainSelf()`: String.**
You should add two new classes - **Cat** and **Dog. Override** the **`explainSelf()`** method by adding concrete animal sound on a new line. (Look at examples below)

You should be able to use the class like this:

Main

```
public static void main(String[] args) {
    Animal cat = new Cat("Oscar", "Whiskas");
    Animal dog = new Dog("Rocky", "Meat");
    System.out.println(cat.explainSelf());
    System.out.println(dog.explainSelf());
}
```

Examples

Input	Output
	I am Oscar and my favourite food is Whiskas MEEOW I am Rocky and my favourite food is Meat DJAAF

Solution

```
public abstract class Animal {
    private String name;
    private String favouriteFood;

    protected Animal(String name, String favouriteFood) {
        this.setName(name);
        this.setFavouriteFood(favouriteFood);
    }

    public String explainSelf() {
        return String.format("I am %s and my favourite food is %s",
                this.getName(),
                this.getFavouriteFood());
    }
```

```
public class Cat extends Animal {
    public Cat(String name, String favouriteFood) {
        super(name, favouriteFood);
    }

    @Override
    public String explainSelf() {
        return String.format("%s%nMEEOW", super.explainSelf());
    }
}
```