10. OOP: Abstraction

Resources:

Abstract classes

Interfaces

Problems

1. Create an abstract class "Shape". It should have the following attributes: x and y. It should have the following methods: getArea and getPerimeter. Also create the constuctor, getters and setters.

   (a) Create a class "Circle". It should have one field called radius. Also create the constructor, getter and setter. The class Circle should inherit from class Shape.

   (b) Create a class "Rectangle". It should have the following fields: width and height. Also create the constructor, getters and setters. The class Rectangle should inherit from class Shape.

   (c) Create an interface "Translatable". It should have the following method: `translate(dx, dy)`

   (d) Create a class "TranslatableCircle". It should inherit from Circle and Translatable.

   The following code should be written in your main method.

   (e) Create an array of 4 shapes. This array should be initialized with both rectangles AND normal circles. Iterate over the array and print the area and perimeter of every shape.

   (f) Why do you not need to use the `instanceof` operator in this case?

   (g) Create a TranslatableCircle. Use the translate method to translate the circle and print the coordinates.

2. What does the code in the following snippet do? And why does it lead to this behavior?

```java
public abstract class Entity {

    private String name;

    public Entity(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

}

public class Main {

    public static void main(String[] args) {
        Entity entity = new Entity("bob");
        System.out.println(entity.getName());
    }

}
```

3. What does the code in the following snippet do? And why does it lead to this behavior?

```java
public abstract class Entity {

    private String name;

    public Entity(String name) {
        this.name = name;
    }

    public abstract String getName();

}

public class Monster extends Entity {

    public Monster(String name) {
        super(name);
    }

    @Override
    public String getName() {
        return super.getName();
    }

}

public class Main {

    public static void main(String[] args) {
        Monster monster = new Monster("bob");
        System.out.println(monster.getName());
    }

}
```

4. What are the differences between interfaces and abstract classes?

5. What is abstraction?