Yerim Oh

March 22, 2023

[Class Abstraction: Abstract class vs. Interface]

There are 4 OOP principles - Inheritance, Polymorphism, Encapsulation, Abstraction

Abstraction
: Data abstraction is the process of hiding certain details and showing only essential lal information to the user

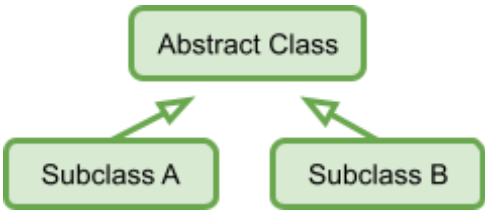
- It helps to focus on what the object does

There are 2 parts in abstraction in Java - Abstract Class and Interface

---

Abstract Class
: a restricted class that cannot be used to create objects

- It must be inherited from another class to access it
- It can be used only with inheritance
- mostly declared where two or more subclasses are also doing the same thing in different ways through different implementations



ex) get the area of a shape

Parent class

Shape.java | Square.java | Circle.java

```java
package creativeTeaching;

/**
 * abstract class; parent class
 *
 * @author Yerim Oh
 * @date March 22, 2023
 */

public abstract class Shape {
    int x;

    Shape (int new_x){
        x = new_x;
    }

    void print() {
        System.out.println("This shape has the side/radius of length " + x);
    }

    /**
     * calculate the area of the shape
     * @param length given length of the side or radius of a shape
     */
    abstract double area (int length);

    public static void main(String[] args) {
        Square square = new Square();
        square.print();
        System.out.println("This shape has the area " + square.area(5));
        Circle circle = new Circle();
        circle.print();
        System.out.println("This shape has the area " + circle.area(5));
    }
}
```

Concrete Methods (methods with body)

Abstract Method (method without body)

Child class

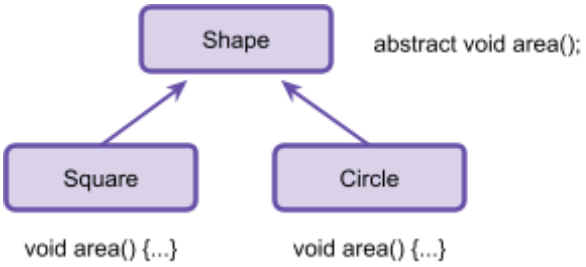
Shape.java | *Square.java | *Circle.java

```java
package creativeTeaching;

/**
 * child class
 *
 * @author Yerim Oh
 * @date March 22, 2023
 */

public class Square extends Shape {

    Square () {
        super(5);
    }

    /**
     * calculate the area of the square
     * @param length given side of a square
     * @return return the area of the square
     */
    @Override
    double area (int length) {
        double area = length*length;
        return area;
    }
}
```

Shape.java | *Square.java | *Circle.java

```java
package creativeTeaching;

/**
 * child class
 *
 * @author Yerim Oh
 * @date March 22, 2023
 */

public class Circle extends Shape {

    Circle () {
        super(5);
    }

    /**
     * calculate the area of the circle
     * @param length given radius of a shape
     * @return return the area of the circle
     */
    @Override
    double area (int length) {
        double area = length*length*Math.PI;
        return area;
    }
}
```

* Any concrete class that extends an abstract class
* Child class must override all abstract methods of the parent class



- The subclass provides implementations for all of the abstract methods in its parent class

Output

```
This shape has the side/radius of length 5
This shape has the area 25.0
This shape has the side/radius of length 5
This shape has the area 78.53981633974483
```

Interface

: a collection of abstract methods

- It must be implemented by concrete classes
- It can contain only Abstract methods (methods without body)
- All methods inside an interface are “public abstract” as default (do not have to declare)

`interface` = qualifier before a class makes it non-instantiable

A concrete class that has the same behaviors as described in an interface can implement the interface

- Implementing classes must declare the method definitions with public access modifiers because interfaces are public

ex) get the speed of the person walking

Interface

```java
package creativeTeaching;

/**
 * interface
 *
 * @author Yerim Oh
 * @date March 22, 2023
 */

interface Walk {

    /**
     * calculate the speed
     * @param d the distance the person walked in kilometers
     * @param t the amount of time the person walked in hours
     */
    float speed(int d, int t);
}
```

Concrete class implementing the interface

```java
package creativeTeaching;

/**
 * concrete class
 *
 * @author Yerim Oh
 * @date March 22, 2023
 */

class PersonWalking implements Walk {

    /**
     * calculate the speed
     * @param distance the distance the person walked in kilometers
     * @param time the amount of time the person walked in hours
     */
    @Override
    public float speed (int distance, int time) {
        float speed = (float)distance/time;
        return speed;
    }
}
```

Output

```java
package creativeTeaching;

/**
 * @author Yerim Oh
 * @date March 22, 2023
 */

class Yerim {
    public static void main (String[] args) {
        PersonWalking yerim = new PersonWalking();
        float yerim_speed = yerim.speed(5, 2);
        System.out.println("Yerim's walking speed is " + yerim_speed + "km per hour.");
    }
}
```

```
Yerim's walking speed is 2.5km per hour.
```