Middlesex University London

Lecture 12

Inheritance, Polymorphism, Array List

TODAY:

- Inheritance
- Polymorphism
- Superclass
- Difference between overriding and overloading
- ArrayList

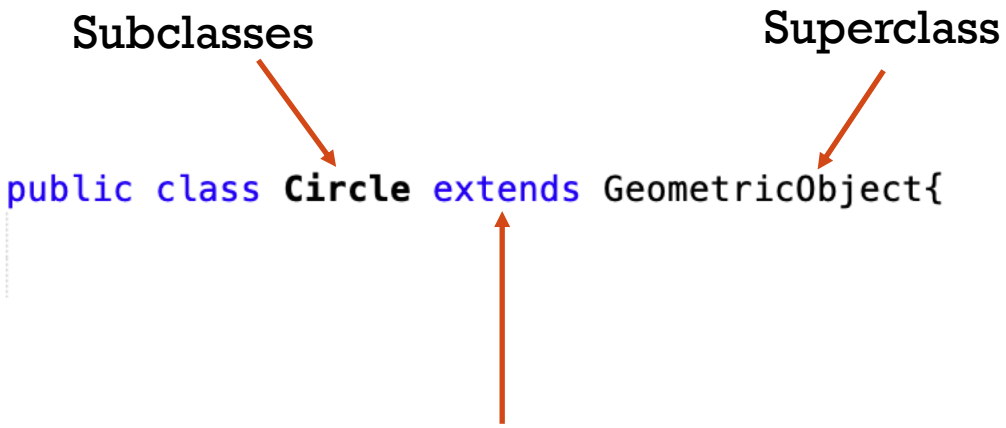
INHERITANCE VS POLYMORPHISM

- Inheritance and Polymorphism are additional aspects of Object Oriented Programming concepts
- Both are used for reducing complexity of the program and increase rentability, scalability, and reusability of the program.

INHERITANCE

- Inheritance is a process in which one class is allowed to inherit all the feature (parameters and methods) of another class
- Inheritance terminology:
- Superclass
  - A class whose feature are inherit
- Subclasses
  - A class that inherit the superclass feature

INHERITANCE



EXAMPLE:

```java
public class GeometricObject {

    public GeometricObject(String color, boolean filled) {
        this.color = color;
        this.filled = filled;
    }
    private String color;
    private boolean filled;
    private Date date;
    public GeometricObject (){

    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public boolean isFilled() {
        return filled;
    }

    public void setFilled(boolean filled) {
        this.filled = filled;
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

    public String toString() {
        return "GeometricObject{" + "color=" + color + ", filled=" + filled + ", date=" + date + '}';
    }
```

```java
public class Circle extends GeometricObject{


    private double radius;

    public Circle() {
    }

    public Circle(double radius) {
        this.radius = radius;
    }
    public Circle(double radius, String colour, boolean filler) {
        this.radius = radius;
        setColor(colour);
        setFilled(filler);
    }
    public double getRadius() {
        return radius;
    }

    public void setRadius(double radius) {
        this.radius = radius;
    }
    public double getPerimeter(){
        return 2* radius *Math.PI;
    }

    public double getDiameter(){
        return 2 * radius;
    }


}
```

```java
public class Rectangle extends GeometricObject{

    private double width;
    private double height;

    public Rectangle(){

    }
    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }


    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        this.width = width;
    }

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }
    public double getArea(){
        return width * height;
    }

    public double getPerimeter (){
        return 2 * (width + height);
    }

    public String toString() {
        return super.toString()+"Rectangle{" + "width=" + width + ", height=" + height + '}';
    }
```

```java
public class Main {
    public static void main(String[] args) {
        Circle c = new Circle (1);
        c.setColor("red");
        c.setDate(new Date ());
        c.setFilled(true);
        System.out.println(c);

        Rectangle r = new Rectangle ();
        r.setColor("blue");
        r.setDate(new Date ());
        r.setFilled(true);
        r.setHeight(10);
        r.setWidth(20);
        System.out.println(r);
    }

}
```

CALLING SUPERCLASS METHODS

In the Circle class:

```
public void printCircle(){
    System.out.println("Date of creating circle is:"
    + super.getDate());
}
```

Super key words

```
public class Main {
    public static void main(String[] args) {
        Circle c = new Circle (1);
        c.setColor("red");
        c.setDate(new Date ());
        c.setFilled(true);
        System.out.println(c);

        c.printCircle();

        Rectangle r = new Rectangle ();
        r.setColor("blue");
        r.setDate(new Date ());
        r.setFilled(true);
        r.setHeight(10);
        r.setWidth(20);
        System.out.println(r);
    }

}
```

QUESTION?

- What is the difference batwing ‘this’ and ‘super’ keywords ?

OVERRIDING METHODS

- To override method, the method must be defined in the subclass using the same signature as in its superclass. Subclass inherits methods from superclass and it could be modify.

In the Circle class:

```
    public String toString() {
        return super.toString()+ "Circle{" + "radius=" + radius + '}';
    }
```

OVERRIDING VS OVERLOADING

- **Overloading** means to define multiple methods with the same name but different signature.

```
public class GeometricObject {

    public void p(String s){
        System.out.println(s);

    }
```

```
public class Circle extends GeometricObject{

    // p method is overloud
    public void p(int i){
        System.out.println("This is the Circle number: "+i);

    }
```

- ***Overriding*** means to provide a new implementation of a method in the class.

```
public class GeometricObject {

    public String toString() {
        return "GeometricObject{" + "color=" + color + ", filled=" + filled +
    }
```

```
public class Circle extends GeometricObject{

    // to String is Override
    public String toString() {
        return super.toString()+ "Circle{" + "radius=" + radius + '
    }
```

POLYMORPHISM

- Polymorphism meaning = > “Many forms”
- A object of a subclass can be used whenever its superclass object is used. A variable of supertype can refer to to subtype object.

```
 */
public class Main {
    public static void main(String[] args) {
        displayObject (new Circle (1, "red", false));
        displayObject (new Rectangle (1, 1, "black", true));

    }

    public static void displayObject(GeometricObject object){
        System.out.println("Created on: "+ object.getDate()+
                "Colour is: "+object.getColor());
    }

}
```

ARRAYLIST

- ArryList class it's used to store an unlimited number of object. It is part of java.util package.

```
ArrayList <Integer> numbers = new ArrayList<Integer>(); //creating a list of Integers

ArrayList <String> words = new ArrayList<String>();//creating a list of String

ArrayList <Object> object = new ArrayList<Object>();//creating a list of Objects
```

ARRAYLIST – SOME METHODS

```
numbers.add(100);              //adding a new element to the end of the list
numbers.add(1, 120);           // adding a new element at the specific index space
System.out.println(numbers.contains(10));
numbers.get(0);                //return the element from this list at the specific index
numbers.size();                // return the number of elements in this list
numbers.remove(1);
numbers.clear();               //removes all elements from the list

numbers.add(11);
numbers.add(12);
```

LIST FROM ARRAYLIST

- Java provides the methods for creating a list from an Array, for:

sorting a list, shuffling and finding min and max element.

```
String [] names = {"Anna", "Olga", "Renata", "Katarina"};
ArrayList <String > names1 = new ArrayList <String>(Arrays.asList(names));


/* Sorting in asscending order*/
Collections.sort(names1);
for(String s : names1){
    System.out.println(s);
}

/* Sorting in decreasing order*/
Collections.sort(names1, Collections.reverseOrder());

for(String s : names1){
    System.out.println(s);
}

System.out.println("Max: "+Collections.max(names1));
System.out.println("Min: "+Collections.min(names1));

Collections.shuffle(names1);
```

HOMEWORK

- A part from finishing sobs:
- Chapter 11.6 (p. 446) The Object class and its toString method
- Chapter 11.8 (p. 447) Dynamic Binding
- Chapter 11.9 (p. 451) Casting Objects and the instanceof operator
- Chapter 11.10 (p. 455) The Object's equals Method

KEY THINGS YOU SHOULD UNDERSTAND AFTER WEEK 14

- Inheritance
- Polymorphism
- Superclass
- Difference between overriding and overloading
- ArrayList