

Java. Inheritance

IT Academy

Agenda

- Classes. Interfaces. Abstract Classes
- Inheritance in Java
- Polymorphism
- Java Classes. Examples



Abstract Classes

- A class must be declared **abstract** when it has one or more abstract *methods*.
- A method is declared abstract when it has a method heading, but **no body** – which means that an abstract method has no implementation code inside curly braces like normal methods do.
 - The derived class must provide a definition method;
 - The derived class must be declared abstract itself.
- A non abstract class is called a **concrete class**.

Abstract Classes

```
/* The Figure class must be declared as abstract because it  
contains an abstract method */
```

```
public abstract class Figure {
```

```
/* because this is an abstract method the body will be blank */
```

```
    public abstract double getArea();
```

```
}
```

```
public class Circle extends Figure {
```

```
    private double radius;
```

```
    public Circle (double radius) {
```

```
        this.radius = radius;
```

```
    public double getArea() {
```

```
        return (3.14 * (radius * 2));    } }
```

Classes. Abstract Classes

```
public class Rectangle extends Figure {
    private double length, width;
    //
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    //
    public double getArea() {
        return length * width;
    }
}
```

Interfaces

- An interface differs from an abstract class because an interface is **not** a class.
- An interface is essentially a **type** that can be satisfied by any class that implements the interface.
- Any class that implements an interface must satisfy 2 conditions
 - It must have the phrase "**implements** *Interface_Name*" at the beginning of the class definition;
 - It must implement **all** of the method headings listed in the interface definition.

Interfaces

```
public interface Dog {  
    public boolean Barks();  
    public boolean isGoldenRetriever();  
}  
  
public class SomeClass implements Dog {  
    public boolean Barks() {  
        // method definition here  
    }  
    public boolean isGoldenRetriever() {  
        // method definition here  
    }  
}
```

Inheritance in Java

- **Assignment** operator. What will be done ?

```
int num=1;  
double data=1.0;  
data=num; // num=data; ???
```

```
class Aclass {  
    int field1=10;  
}  
class Bclass extends Aclass {  
    int field2=20;  
}
```

```
Aclass a=new Aclass( );  
Bclass b=new Bclass( );  
a=b; // b=a; ??? // a = new Bclass( );
```


Inheritance in Java

```
public class ClassA {  
    public int i=1;  
    public void m1() {  
        System.out.println("ClassA, metod m1, i="+i); }  
    public void m2() {  
        System.out.println("ClassA, metod m2, i="+i); }  
    public void m3() {  
        System.out.print("ClassA, metod m3, runnind m4(): ");  
        m4(); }  
    public void m4() {  
        System.out.println("ClassA, metod m4");  
    } }  
}
```

Inheritance in Java

```
public class ClassB extends ClassA {  
    public double i=1.1;  
    public void m1() {  
        System.out.println("ClassB, metod m1, i="+i);  
    }  
    public void m4() {  
        System.out.println("ClassB, metod m4");  
    }  
}
```

- Automatically added **default constructor**.

Inheritance in Java

```
public class ApplAB {  
    public static void main(String[] args) {  
        System.out.println("The Start.");  
        ClassA a;  
        ClassA b;  
        a=new ClassA();  
        b=new ClassB();  
        System.out.println("Test ClassA.");  
        a.m1();  
        a.m2();  
        a.m3();  
        a.m4();  
    }  
}
```

Inheritance in Java

```
System.out.println("Test ClassB.");
b.m1();
b.m2();
b.m3();
b.m4();
ClassB b0;
b0=new ClassB();
System.out.println("Test_0 ClassB.");
b0.m1();
b0.m2();
b0.m3();
b0.m4();
System.out.println("The End."); } }
```

Inheritance in Java

The Start.

Test ClassA.

ClassA, metod m1, i=1

ClassA, metod m2, i=1

ClassA, metod m3, runnind m4(): ClassA, metod m4

ClassA, metod m4

Test ClassB.

ClassB, metod m1, i=1.1

ClassA, metod m2, i=1

ClassA, metod m3, runnind m4(): ClassB, metod m4

ClassB, metod m4

Test_0 ClassB.

ClassB, metod m1, i=1.1

ClassA, metod m2, i=1

ClassA, metod m3, runnind m4(): ClassB, metod m4

ClassB, metod m4

The End.

Polymorphism

```
package com.softserve.train;
```

```
public abstract class ACar {  
    private double maxSpeed;
```

```
    public double getMaxSpeed() {  
        return maxSpeed;  
    }
```

```
    public void setMaxSpeed(double maxSpeed) {  
        this.maxSpeed = maxSpeed;  
    }
```

```
    abstract void carRides();  
}
```

Polymorphism

```
public class BmwX6 extends ACar {
    public BmwX6() {}
    @Override
    public void carRides() {
        setMaxSpeed(200);
        System.out.println("Car Rides");
        workedEngine();
        workedGearBox();
    }
    public void workedEngine() {
        System.out.println("BmwX6: Engine Running on Petrol.");
        System.out.println("BmwX6: Max Speed: " +
            getMaxSpeed());
    }
}
```

Polymorphism

```
private void workedGearBox( ) {
```

```
    System.out.println("BmwX6: Worked GearBox.");
```

```
}
```

```
public void lightsShine( ) {
```

```
    System.out.println("BmwX6: Halogen Headlights.");
```

```
}
```

```
}
```

- inheritance of **private** fields and methods ?

Polymorphism

```
package com.softserve.train;
public class BmwX6mod extends BmwX6 {
    public BmwX6mod() {
        super();
    }
    @Override
    public void workedEngine() {
        System.out.println("BmwX6mod:
                            Engine Running on Diesel.");
        System.out.println("BmwX6mod: Max Speed: " +
                            getMaxSpeed());
    }
    @Override
    public void lightsShine() {
        System.out.println("BmwX6mod: Xenon Headlights.");
        super.lightsShine();
    }
}
```

Polymorphism

```
package com.softserve.train;
public class Appl {
    public static void main(String[ ] args) {
        ACar carX6 = new BmwX6( );
        ACar carX6mod = new BmwX6mod( );
        BmwX6 carX6mod2 = new BmwX6mod( );
        carX6.carRides( );
        ((BmwX6)carX6).lightsShine( );
        carX6mod.carRides( );
        ((BmwX6)carX6mod).lightsShine( );
        carX6mod2.carRides( );
        carX6mod2.lightsShine( );
    }
}
```

Java Classes

- What is **wrong** in the code ?

```
package com.softserve.train;
```

```
public class Parent {  
    int f() {  
        return 1;  
    }  
}
```

- **public int** f() { ... what will in case **???**

Java Classes

```
package com.softserve.train2;
```

```
import com.softserve.train.Parent;
```

```
public class Child extends com.softserve.train.Parent {
```

```
    int f() {  
        return 2;  
    }
```

```
}
```

▪ **protected int** f() { ... what will in case ???

Java Classes

```
package com.softserve.train;
```

```
import com.softserve.train2.Child;
```

```
public class Grandchild extends com.softserve.train2.Child {  
    public int y() {  
        return this.f(); // ???  
    }  
}
```

Java Classes

```
package com.softserve.train;
```

```
public class Appl {  
    public static void main(String[] args) {  
        Grandchild g = new Grandchild();  
        boolean result = (g.f( ) == g.y( )); // ???  
        System.out.println("result=" + result);  
    }  
}
```

