

# Java - Inheritance & Polymorphism

Md. Mohsin Uddin

East West University

*mmuddin@ewubd.edu*

June 8, 2019

# Inheritance : The extend keyword - Part I

```
/* File name : Calculation.java */
package Mid2;
public class Calculation {
    int z;
    public void addition(int x, int y) {
        z = x + y;
        System.out.println("The sum of the given numbers:" + z);
    }
    public void Subtraction(int x, int y) {
        z = x - y;
        System.out.println
        ("The difference between the given numbers:" + z);
    }
}
```

```
/* File name : My_Calculation.java */
package Mid2;
public class My_Calculation extends Calculation {
    public void multiplication(int x, int y) {
```

# Inheritance : The extend keyword - Part II

```
        z = x * y;  
        System.out.println("The product of the given numbers:" + z);  
    }  
    public static void main(String args[]) {  
        int a = 20, b = 10;  
        My_Calculation demo = new My_Calculation();  
        demo.addition(a, b);  
        demo.Subtraction(a, b);  
        demo.multiplication(a, b);  
    }  
}
```

When the above code is compiled and executed, it produces the following result:

```
The sum of the given numbers:30  
The difference between the given numbers:10  
The product of the given numbers:200
```

# Inheritance : The super Keyword - Part I

```
/* File name : Super_class.java */
package Mid2;
public class Super_class {
    int num = 20;
    public Super_class() {
        System.out.println(" Super_class _constructor" );
    }
    // display method of superclass
    public void display() {
        System.out.println
            (" This_is_the_display_method_of_superclass" );
    }
}
```

```
/* File name : Sub_class.java */
package Mid2;
public class Sub_class extends Super_class {
    int num = 10;
    // display method of sub class
```

# Inheritance : The super Keyword - Part II

```
public Sub_class() {  
    System.out.println("Sub_class_constructor");  
}  
public void display() {  
    System.out.println  
    ("This_is_the_display_method_of_subclass");  
}  
public void my_method() {  
    // Instantiating subclass  
    Sub_class sub = new Sub_class();  
    // Invoking the display() method of sub class  
    sub.display();  
    // Invoking the display() method of superclass  
    super.display();  
    // printing the value of variable num of subclass  
    System.out.println  
    ("value_of_the_variable_named_num_in_sub_class:"  
    + sub.num);  
    // printing the value of variable num of superclass
```

# Inheritance : The super Keyword - Part III

```
        System.out.println
            ("value of the variable named num in super class:"
             + super.num);
    }
    public static void main(String args[]) {
        Sub_class obj = new Sub_class();
        obj.my_method();
    }
}
```

When the above code is compiled and executed, it produces the following result:

```
Super class constructor
Sub class constructor
Super class constructor
Sub class constructor
This is the display method of subclass
This is the display method of superclass
value of the variable named num in sub class:10
value of the variable named num in super class:20
```

# Inheritance : Invoking Superclass Constructor - Part 1

```
/* File name : Superclass.java */
package Mid2;
public class Superclass {
    int age;
    Superclass(int age) {
        this.age = age-5;
    }
    public void getAge() {
        System.out.println
            ("The value of the variable named age in super class is : ")
    }
}

/* File name : Subclass.java */
package Mid2;
public class Subclass extends Superclass {
    Subclass(int age) {
        super(age);
    }
    public static void main(String args[]) {
```

# Inheritance : Invoking Superclass Constructor - Part II

```
Subclass s = new Subclass(24);  
s.getAge();  
}  
}
```

When the above code is compiled and executed, it produces the following result:

| The value of the variable named age in super class is: 19



# Inheritance - IS-A relationship

```
/* File name : Dog.java */
package Mid2.animal;
interface Animal{}
class Mammal implements Animal{}
class Dog extends Mammal {
    public static void main(String args[]) {
        Mammal m = new Mammal();
        Dog d = new Dog();
        System.out.println(m instanceof Animal);
        System.out.println(d instanceof Mammal);
        System.out.println(d instanceof Animal);
    }
}
```

When the above code is compiled and executed, it produces the following result:

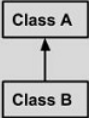
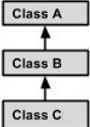
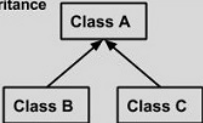
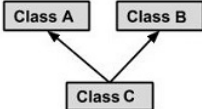
```
true
true
true
```

# Inheritance - IS-A relationship vs HAS-A relationship

```
/* File name : Van.java */  
class Vehicle{}  
class Speed{}  
class Van extends Vehicle {  
    private Speed sp;  
}
```

# Types of Inheritance

There are various types of inheritance as demonstrated below.

<b>Single Inheritance</b>  <pre>graph BT; B[Class B] --&gt; A[Class A]</pre>	<pre>public class A {     ..... } public class B extends A {     ..... }</pre>
<b>Multi Level Inheritance</b>  <pre>graph BT; C[Class C] --&gt; B[Class B]; B --&gt; A[Class A]</pre>	<pre>public class A { .....} public class B extends A {.....} public class C extends B {.....}</pre>
<b>Hierarchical Inheritance</b>  <pre>graph BT; B[Class B] --&gt; A[Class A]; C[Class C] --&gt; A</pre>	<pre>public class A { .....} public class B extends A {.....} public class C extends A {.....}</pre>
<b>Multiple Inheritance</b>  <pre>graph BT; A[Class A] --&gt; C[Class C]; B[Class B] --&gt; C</pre>	<pre>public class A { .....} public class B { .....} public class C extends A,B {     ..... } // Java does not support multiple Inheritance</pre>

# Java - Overriding : Example 1 - Part 1

```
/* File name : TestDog.java */
package Mid2.overriding;
class Animal {
    public void move() {
        System.out.println(" Animals can move" );
    }
}
class Dog extends Animal {
    public void move() {
        System.out.println(" Dogs can walk and run" );
    }
}
public class TestDog {
    public static void main(String args[]) {
        Animal a = new Animal();
        Animal b = new Dog();
        //runs the method in Animal class
        a.move();
    }
}
```

# Java - Overriding : Example 1 - Part II

```
        //runs the method in Dog class  
        b.move();  
    }  
}
```

When the above code is compiled and executed, it produces the following result:

```
| Animals can move  
| Dogs can walk and run
```

# Java - Overriding : Example 2 - Part 1

```
/* File name : TestDog.java */
package Mid2.overriding;
class Animal {
    public void move() {
        System.out.println(" Animals can move ");
    }
}
class Dog extends Animal {
    public void move() {
        System.out.println(" Dogs can walk and run ");
    }
    public void bark() {
        System.out.println(" Dogs can bark ");
    }
}
public class TestDog {
    public static void main(String args[]) {
        Animal a = new Animal();
        Animal b = new Dog();
    }
}
```

## Java - Overriding : Example 2 - Part II

```
        a.move();  
        b.move();  
        b.bark(); //error  
    }  
}
```

When the above code is compiled and executed, it produces the following result:

```
TestDog.java:26: error: cannot find symbol b.bark();  
symbol: method bark()  
location: variable b of type Animal  
1 error
```

# Java - Overriding : Example 3 - Part 1

```
/* File name : TestDog.java */
class Animal {
    public void move() {
        System.out.println(" Animals can move" );
    }
}

class Dog extends Animal {
    public void move() {
        super.move();    // invokes the super class method
        System.out.println(" Dogs can walk and run" );
    }
}

public class TestDog {

    public static void main(String args[]) {
        // Animal reference but Dog object
        Animal b = new Dog();
    }
}
```



# Java - Overriding : Example 3 - Part II

```
// runs the method in Dog class  
b.move();  
}  
}
```

When the above code is compiled and executed, it produces the following result:

```
| Animals can move  
| Dogs can walk and run
```

# Java - Polymorphism : Example 1 - Part I

```
/* File name : Employee.java */
package Mid2.polymorphism;
public class Employee {
    private String name;
    private String address;
    private int number;
    public Employee(String name, String address, int number) {
        System.out.println("Constructing an Employee");
        this.name = name;
        this.address = address;
        this.number = number;
    }
    public void mailCheck() {
        System.out.println
            ("Mailing a check to " + this.name + " " + this.address);
    }
    public String toString() {
        return name + " " + address + " " + number;
    }
}
```

## Java - Polymorphism : Example 1 - Part II

```
public String getName() {  
    return name;  
}  
public String getAddress() {  
    return address;  
}  
public void setAddress(String newAddress) {  
    address = newAddress;  
}  
public int getNumber() {  
    return number;  
}  
}  
/* File name : Salary.java */  
package Mid2.polymorphism;  
public class Salary extends Employee {  
    private double salary; // Annual salary  
  
    public Salary
```

# Java - Polymorphism : Example 1 - Part III

```
(String name, String address, int number, double salary) {  
    super(name, address, number);  
    setSalary(salary);  
}  
public void mailCheck() {  
    System.out.println("Within mailCheck of Salary class");  
    System.out.println("Mailing check to " + getName()  
        + " with salary " + salary);  
}  
public double getSalary() {  
    return salary;  
}  
public void setSalary(double newSalary) {  
    if(newSalary >= 0.0) {  
        salary = newSalary;  
    }  
}  
public double computePay() {  
    System.out.println("Computing salary pay for " + getName());
```

# Java - Polymorphism : Example 1 - Part IV

```
        return salary/52;
    }
}
/* File name : TestPolymorphism.java */
package Mid2.polymorphism;
public class TestPolymorphism {
    public static void main(String [] args) {
        Salary s = new Salary("Mizan", "Calgary", 3, 3600.00);
        Employee e = new Salary("John", "Boston", 2, 2400.00);
        System.out.println
            (" Call mailCheck using Salary reference —" );
        s.mailCheck();
        System.out.println
            ("\n Call mailCheck using Employee reference —" );
        e.mailCheck();
    }
}
```

# Java - Polymorphism : Example 1 - Part V

When the above code is compiled and executed, it produces the following result:

```
Constructing an Employee
Constructing an Employee
Call mailCheck using Salary reference –
Within mailCheck of Salary class
Mailing check to Mohd Mohtashim with salary 3600.0
Call mailCheck using Employee reference–
Within mailCheck of Salary class
Mailing check to John Adams with salary 2400.0
```

# References



DEITEL, Java How to Program, 11/e



Java: the complete reference, Herbert Schildt, McGraw-Hill Education Group