# L33

Monday, January 31, 2022 6:22 PM

## Inheritance

Q1. **Inheritance** is a way by which a class can inherit what is already there in another existing class.

We can classify inheritance into two kinds:

1. **Implementation Inheritance** (or code inheritance)
2. **Interface Inheritance** (or inheritance of behaviour or contract)

Here we will understand *implementation inheritance* and later learn about *interface inheritance* after we learn *interfaces*.

In Java, a class can **inherit** implementation from **only one class**, it is also referred as **single-inheritance**. (Other languages like c++ support multiple inheritance of implementation)

In Java, we use extends keyword when we want a class to inherit (extend) from another class. For example:

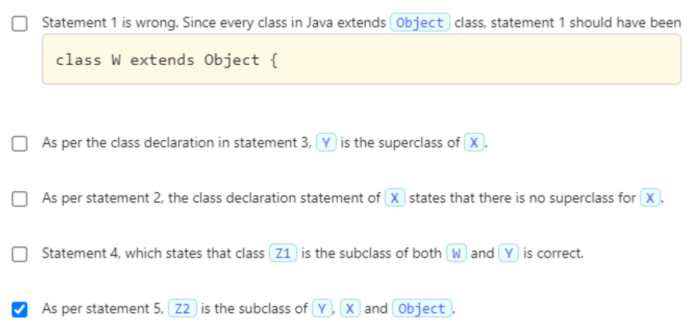
```
class A {
        ...
}
class B extends A {
        ...
}
```

In the above code B extends A. B is called the subclass of A. A is called the super class of B.

**Note:** Every class in Java automatically extends the root class Object if it does not explicitly extends another class. Which means in our above example, class A is a subclass of Object and Object is the superclass of A.

Using the above concepts select all the correct statements from the below code:

```
class W { // statement 1}
class X { // statement 2...
}
class Y extends X { // statement 3...
}
class Z1 extends W, Y { // statement 4...
}
class Z2 extends Y { // statement 5...
}
```



- [ ] Statement 1 is wrong. Since every class in Java extends `Object` class, statement 1 should have been

```
class W extends Object {
```

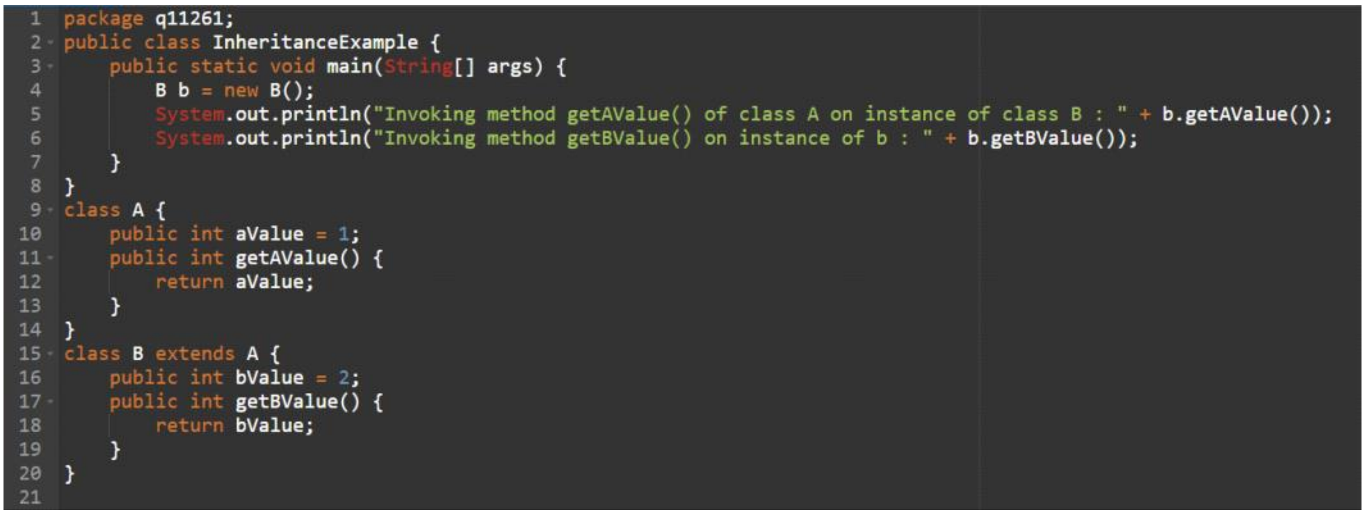
- [ ] As per the class declaration in statement 3, `Y` is the superclass of `X`.
- [ ] As per statement 2, the class declaration statement of `X` states that there is no superclass for `X`.
- [ ] Statement 4, which states that class `Z1` is the subclass of both `W` and `Y` is correct.
- [✓] As per statement 5, `Z2` is the subclass of `Y`, `X` and `Object`.

Q2. In implementation inheritance the subclass inherits the implementation facilitating code reuse. For example:

```
class A {
        public int aValue = 1;
        public int getAValue() {
                return aValue;
        }
}
class B extends A {
        public int bValue = 2;
        public int getBValue() {
                return bValue;
        }
}
```

In the above code, class B also contains the inherited feild aValue and the inherited method getAValue().

See and retype the below code. When you click submit, in the output you will notice that we are able to invoke the inherited method getAValue() (which is implemented in class A) on an instance of class B.



```java
package q11261;
public class InheritanceExample {
    public static void main(String[] args) {
        B b = new B();
        System.out.println("Invoking method getAValue() of class A on instance of class B : " + b.getAValue());
        System.out.println("Invoking method getBValue() on instance of b : " + b.getBValue());
    }
}
class A {
    public int aValue = 1;
    public int getAValue() {
        return aValue;
    }
}
class B extends A {
    public int bValue = 2;
    public int getBValue() {
        return bValue;
    }
}
```

**Q3.** In implementation inheritance, when the subclass inherits from a superclass (also callled base class), the subclass instance can be referred by the base class reference.
For example:

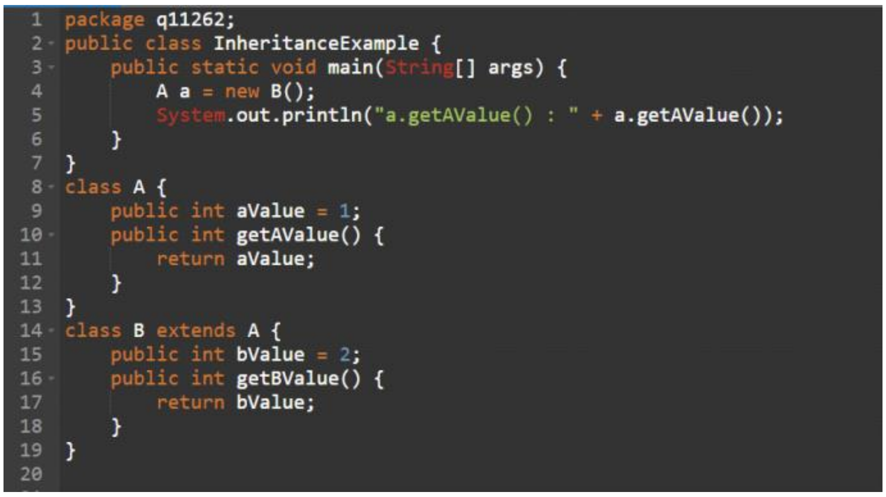
```
class A {
        public int aValue = 1;
        public int getAValue() {
                return aValue;
        }
}
class B extends A {
        public int bValue = 2;
        public int getBValue() {
                return bValue;
        }
}
```

Then the below statement is valid:
A a = new B();

**Note:** In the above code, we can access all members of class A via the reference a even though the actual object created in memory using the new keyword is an instance of class B.

According to the above code, the statement`a.getBValue()` will result in compilation error. This is because reference a which is of type class A does not know about the members declared in its subclass B.

See and retype the below code to understand the above concept.



```java
package q11262;
public class InheritanceExample {
    public static void main(String[] args) {
        A a = new B();
        System.out.println("a.getAValue() : " + a.getAValue());
    }
}
class A {
    public int aValue = 1;
    public int getAValue() {
        return aValue;
    }
}
class B extends A {
    public int bValue = 2;
    public int getBValue() {
        return bValue;
    }
}
```

**Q4.** Write a Java program to illustrate the **single inheritance** concept.

Create a class Marks
- contains the data members **id** of **int** data type, **javaMarks**, **cMarks** and **cppMarks** of **float** data type
- write a method **setMarks()** to initialize the data members
- write a method **displayMarks()** which will display the given data

Create another class Result which is derived from the class Marks
- contains the data members **total** and **avg** of **float** data type
- write a method **compute()** to find total and average of the given marks
- write a method **showResult()** which will display the total and avg marks

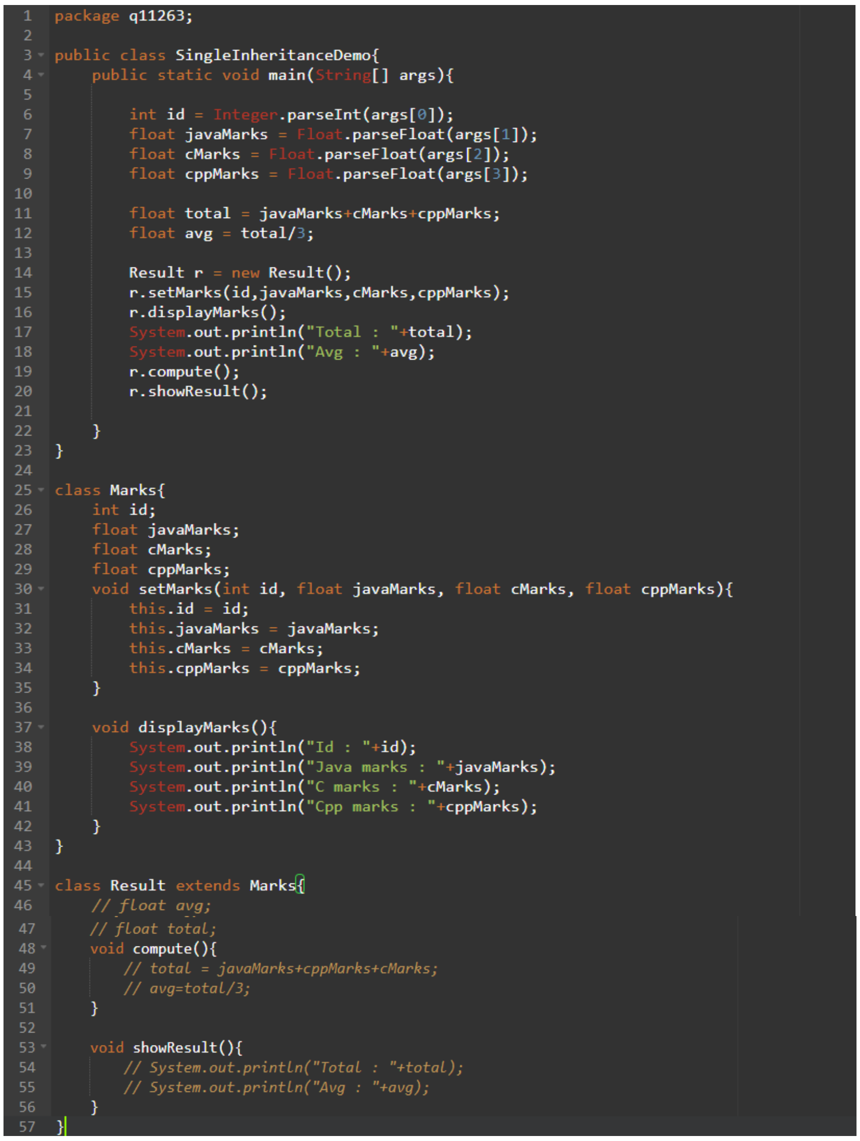
Write a class SingleInheritanceDemo with **main()** method it receives four arguments as **id**, **javaMarks**, **cMarks** and **cppMarks**.

Create object only to the class Result to access the methods.

If the input is given as command line arguments to the **main()** as **"101"**, **"45.50"**, **"67.75"**, **"72.25"** then the program should print the output as:

```
Id : 101
Java marks : 45.5
```

```
C marks : 67.75
Cpp marks : 72.25
Total : 185.5
Avg : 61.833332
```



```java
package q11263;

public class SingleInheritanceDemo{
    public static void main(String[] args){

        int id = Integer.parseInt(args[0]);
        float javaMarks = Float.parseFloat(args[1]);
        float cMarks = Float.parseFloat(args[2]);
        float cppMarks = Float.parseFloat(args[3]);

        float total = javaMarks+cMarks+cppMarks;
        float avg = total/3;

        Result r = new Result();
        r.setMarks(id,javaMarks,cMarks,cppMarks);
        r.displayMarks();
        System.out.println("Total : "+total);
        System.out.println("Avg : "+avg);
        r.compute();
        r.showResult();

    }
}

class Marks{
    int id;
    float javaMarks;
    float cMarks;
    float cppMarks;
    void setMarks(int id, float javaMarks, float cMarks, float cppMarks){
        this.id = id;
        this.javaMarks = javaMarks;
        this.cMarks = cMarks;
        this.cppMarks = cppMarks;
    }

    void displayMarks(){
        System.out.println("Id : "+id);
        System.out.println("Java marks : "+javaMarks);
        System.out.println("C marks : "+cMarks);
        System.out.println("Cpp marks : "+cppMarks);
    }
}

class Result extends Marks{
    // float avg;
    // float total;
    void compute(){
        // total = javaMarks+cppMarks+cMarks;
        // avg=total/3;
    }

    void showResult(){
        // System.out.println("Total : "+total);
        // System.out.println("Avg : "+avg);
    }
}
```

## Q5. Write a Java program to illustrate the **multilevel inheritance** concept.

Create a class Student

- contains the data members **id** of **int** data type and **name** of **string** type
- write a method **setData()** to initialize the data members
- write a method **displayData()** which will display the given **id** and **name**

Create a class Marks which is derived from the class Student

- contains the data members **javaMarks**, **cMarks** and **cppMarks** of **float** data type
- write a method **setMarks()** to initialize the data members
- write a method **displayMarks()** which will display the given data

Create another class Result which is derived from the class Marks

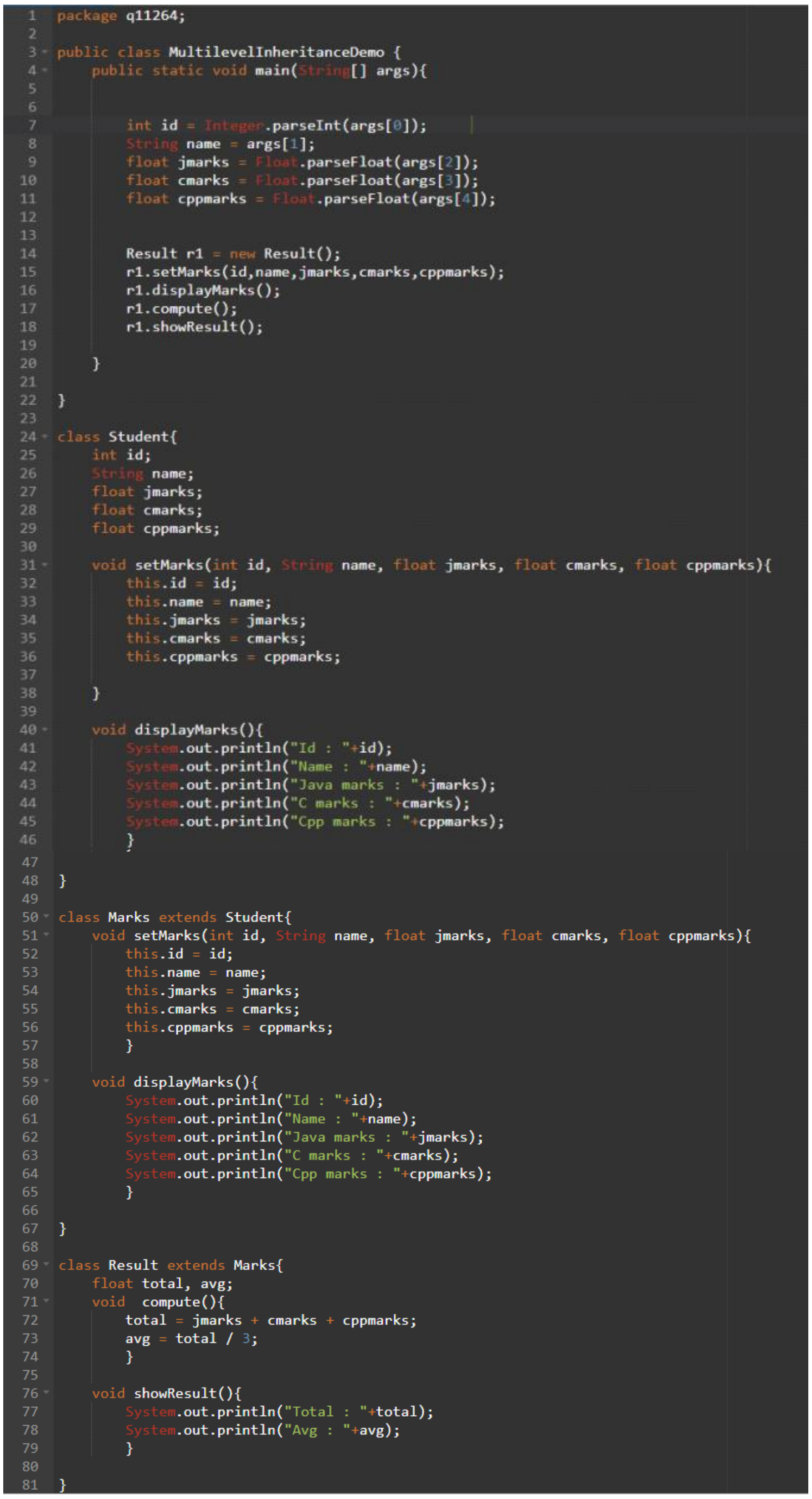
- contains the data members **total** and **avg** of **float** data type
- write a method **compute()** to find total and average of the given marks
- write a method **showResult()** which will display the total and avg marks

Write a class MultilevelInheritanceDemo with the **main()** method which will receive five arguments as **id**, **name**, **javaMarks**, **cMarks** and **cppMarks**.

Create object only to the class Result to access the methods.

If the input is given as command line arguments to the **main()** as **"99", "Lakshmi", "55.5", "78.5", "72"** then the program should print the output as:

```
Id : 99
Name : Lakshmi
Java marks : 55.5
C marks : 78.5
Cpp marks : 72.0
Total : 206.0
Avg : 68.666664
```



```java
package q11264;

public class MultilevelInheritanceDemo {
    public static void main(String[] args){

        int id = Integer.parseInt(args[0]);
        String name = args[1];
        float jmarks = Float.parseFloat(args[2]);
        float cmarks = Float.parseFloat(args[3]);
        float cppmarks = Float.parseFloat(args[4]);


        Result r1 = new Result();
        r1.setMarks(id,name,jmarks,cmarks,cppmarks);
        r1.displayMarks();
        r1.compute();
        r1.showResult();

    }

}

class Student{
    int id;
    String name;
    float jmarks;
    float cmarks;
    float cppmarks;

    void setMarks(int id, String name, float jmarks, float cmarks, float cppmarks){
        this.id = id;
        this.name = name;
        this.jmarks = jmarks;
        this.cmarks = cmarks;
        this.cppmarks = cppmarks;

    }

    void displayMarks(){
        System.out.println("Id : "+id);
        System.out.println("Name : "+name);
        System.out.println("Java marks : "+jmarks);
        System.out.println("C marks : "+cmarks);
        System.out.println("Cpp marks : "+cppmarks);
        }

}

class Marks extends Student{
    void setMarks(int id, String name, float jmarks, float cmarks, float cppmarks){
        this.id = id;
        this.name = name;
        this.jmarks = jmarks;
        this.cmarks = cmarks;
        this.cppmarks = cppmarks;
        }

    void displayMarks(){
        System.out.println("Id : "+id);
        System.out.println("Name : "+name);
        System.out.println("Java marks : "+jmarks);
        System.out.println("C marks : "+cmarks);
        System.out.println("Cpp marks : "+cppmarks);
        }

}

class Result extends Marks{
    float total, avg;
    void  compute(){
        total = jmarks + cmarks + cppmarks;
        avg = total / 3;
        }

    void showResult(){
        System.out.println("Total : "+total);
        System.out.println("Avg : "+avg);
        }

}
```