Java Inheritance Example



**class** Employee{

**float** salary=40000;

}

**class** Programmer **extends** Employee{

**int** bonus=10000;

**public** **static** **void** main(String args[]){

   Programmer p=**new** Programmer();

   System.out.println("Programmer salary is:"+p.salary);

   System.out.println("Bonus of Programmer is:"+p.bonus);

}

}

Output:

Programmer salary is:40000.0

Bonus of programmer is:10000

Graphical user interface, text, application, email

Description automatically generated

In java programming, multiple and hybrid inheritance is supported through interface only. We will learn about interfaces later.

// Java program to illustrate the

// concept of inheritance

// base class

class Bicycle {

// the Bicycle class has two fields

public int gear;

public int speed;

// the Bicycle class has one constructor

public Bicycle(int gear, int speed)

{

this.gear = gear;

this.speed = speed;

}

// the Bicycle class has three methods

public void applyBrake(int decrement)

{

speed -= decrement;

}

public void speedUp(int increment)

{

speed += increment;

}

// toString() method to print info of Bicycle

public String toString()

{

return ("No of gears are " + gear + "\n"

+ "speed of bicycle is " + speed);

}

}

// derived class

class MountainBike extends Bicycle {

// the MountainBike subclass adds one more field

public int seatHeight;

// the MountainBike subclass has one constructor

public MountainBike(int gear, int speed,

int startHeight)

{

// invoking base-class(Bicycle) constructor

super(gear, speed);

seatHeight = startHeight;

}

// the MountainBike subclass adds one more method

public void setHeight(int newValue)

{

seatHeight = newValue;

}

// overriding toString() method

// of Bicycle to print more info

@Override public String toString()

{

return (super.toString() + "\nseat height is "

+ seatHeight);

}

}

// driver class

public class Test {

public static void main(String args[])

{

MountainBike mb = new MountainBike(3, 100, 25);

System.out.println(mb.toString());

}

Graphical user interface, application

Description automatically generated

}

#### **Note: Multiple inheritance is not supported in Java through class.**

When one class inherits multiple classes, it is known as multiple inheritance. For Example:



## **Single Inheritance Example**

When a class inherits another class, it is known as a single inheritance. In the example given below, Dog class inherits the Animal class, so there is the single inheritance.

**class** Animal{

**void** eat()

{

System.out.println("eating...");}

}

**class** Dog **extends** Animal

{

**void** bark()

{

System.out.println("barking...");}

}

**class** TestInheritance{

**public** **static** **void** main(String args[]){

Dog d=**new** Dog();

d.bark();

d.eat();

}

}

Output:

barking...

eating...

## **Graphical user interface, text, application Description automatically generated**

## **Hierarchical Inheritance Example**

When two or more classes inherits a single class, it is known as hierarchical inheritance. In the example given below, Dog and Cat classes inherits the Animal class, so there is hierarchical inheritance.

*File: TestInheritance3.java*

**class** Animal{

**void** eat()

{

System.out.println("eating...");}

}

**class** Dog **extends** Animal{

**void** bark()

{

System.out.println("barking...");}

}

**class** Cat **extends** Animal{

**void** meow(){System.out.println("meowing...");}

}

**class** TestInheritance3{

**public** **static** **void** main(String args[]){

Cat c=**new** Cat();

c.meow();

c.eat();

//c.bark();//C.T.Error

}

}

Output:

Graphical user interface, application

Description automatically generated

meowing...

eating...

## **Q) Why multiple inheritance is not supported in java?**

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

Since compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error.

**class** A{

**void** msg(){System.out.println("Hello");}

}

**class** B{

**void** msg(){System.out.println("Welcome");}

}

**class** C **extends** A,B{//suppose if it were

**public** **static** **void** main(String args[]){

   C obj=**new** C();

   obj.msg();//Now which msg() method would be invoked?

}

}

Compile Time Error

Graphical user interface, application

Description automatically generated

|  |
| --- |
| / Java program to illustrate the  // concept of inheritance    // base class  **class** Bicycle {      // the Bicycle class has two fields  **public** **int** gear;  **public** **int** speed;        // the Bicycle class has one constructor  **public** Bicycle(**int** gear, **int** speed)      {  **this**.gear = gear;  **this**.speed = speed;      }        // the Bicycle class has three methods  **public** **void** applyBrake(**int** decrement)      {          speed -= decrement;      }    **public** **void** speedUp(**int** increment)      {          speed += increment;      }        // toString() method to print info of Bicycle  **public** String toString()      {  **return** ("No of gears are " + gear + "\n"                  + "speed of bicycle is " + speed);      }  }    // derived class  **class** MountainBike **extends** Bicycle {        // the MountainBike subclass adds one more field  **public** **int** seatHeight;        // the MountainBike subclass has one constructor  **public** MountainBike(**int** gear, **int** speed,  **int** startHeight)      {          // invoking base-class(Bicycle) constructor  **super**(gear, speed);          seatHeight = startHeight;      }        // the MountainBike subclass adds one more method  **public** **void** setHeight(**int** newValue)      {          seatHeight = newValue;      }        // overriding toString() method      // of Bicycle to print more info      @Override **public** String toString()      {  **return** (**super**.toString() + "\nseat height is "                  + seatHeight);      }  }    // driver class  **public** **class** Test {  **public** **static** **void** main(String args[])      {            MountainBike mb = **new** MountainBike(3, 100, 25);          System.out.println(mb.toString());      }  } |

**Output**

No of gears are 3

speed of bicycle is 100

seat height is 25

Graphical user interface, text, application

Description automatically generated

|  |
| --- |
| // Java program to illustrate the  // concept of single inheritance  **import** java.io.\*;  **import** java.lang.\*;  **import** java.util.\*;    **class** one {  **public** **void** print\_geek()      {          System.out.println("Geeks");      }  }    **class** two **extends** one {  **public** **void** print\_for()  {  System.out.println("for");  }  }  // Driver class  **public** **class** Main {  **public** **static** **void** main(String[] args)      {          two g = **new** two();          g.print\_geek();          g.print\_for();          g.print\_geek();      }  } |

**Output**

Geeks

for

Geeks

Graphical user interface, application

Description automatically generated

|  |
| --- |
| // Java program to illustrate the  // concept of Multilevel inheritance  **import** java.io.\*;  **import** java.lang.\*;  **import** java.util.\*;    **class** one {  **public** **void** print\_geek()      {          System.out.println("Geeks");      }  }    **class** two **extends** one {  **public** **void** print\_for() { System.out.println("for"); }  }    **class** three **extends** two {  **public** **void** print\_geek()      {          System.out.println("Geeks");      }  }    // Drived class  **public** **class** Main {  **public** **static** **void** main(String[] args)      {          three g = **new** three();          g.print\_geek();          g.print\_for();          g.print\_geek();      }  } |

**Output**

Geeks

for

Geeks

Graphical user interface, application

Description automatically generated

|  |
| --- |
| / Java program to illustrate the  // concept of Hierarchical  inheritance    **class** A {  **public** **void** print\_A() { System.out.println("Class A"); }  }    **class** B **extends** A {  **public** **void** print\_B() { System.out.println("Class B"); }  }    **class** C **extends** A {  **public** **void** print\_C() { System.out.println("Class C"); }  }    **class** D **extends** A {  **public** **void** print\_D() { System.out.println("Class D"); }  }    // Driver Class  **public** **class** Test {  **public** **static** **void** main(String[] args)      {          B obj\_B = **new** B();          obj\_B.print\_A();          obj\_B.print\_B();            C obj\_C = **new** C();          obj\_C.print\_A();          obj\_C.print\_C();            D obj\_D = **new** D();          obj\_D.print\_A();          obj\_D.print\_D();      }  } |

**Output**

Class A

Class B

Class A

Class C

Class A

Class D

Text

Description automatically generated

|  |
| --- |
| // Java program to illustrate the  // concept of Multiple inheritance  **import** java.io.\*;  **import** java.lang.\*;  **import** java.util.\*;    **interface** one {  **public** **void** print\_geek();  }    **interface** two {  **public** **void** print\_for();  }    **interface** three **extends** one, two {  **public** **void** print\_geek();  }  **class** child **implements** three {      @Override **public** **void** print\_geek()      {          System.out.println("Geeks");      }    **public** **void** print\_for() { System.out.println("for"); }  }    // Drived class  **public** **class** Main {  **public** **static** **void** main(String[] args)      {          child c = **new** child();          c.print\_geek();          c.print\_for();          c.print\_geek();      }  } |

**Output**

Geeks

for

Geeks

Graphical user interface, text, application, chat or text message

Description automatically generated

Programmer salary is:40000.0

Bonus of programmer is:10000