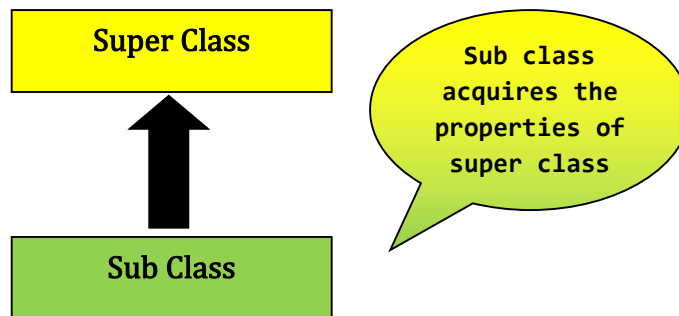


Chapter 5

INHERITANCE

Inheritance

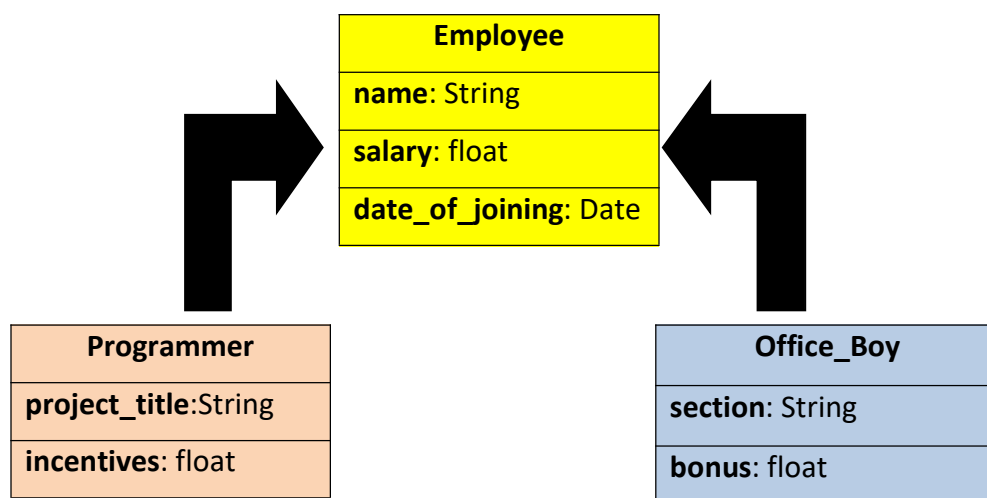
It is the process by which object of one class can acquire the properties of object of another class. The class, from which properties are inherited to another class, is known as **super class**. The class, to which properties are inherited from an existing class, is known as **sub class**. A sub class inherits all non private variables and methods defined by the super class and add its own, unique elements. To derive a sub class from super class, we use **extends** keyword.



Advantages of using inheritance

- Inheritance allows the creation of hierarchical classifications of classes.
- It allows code reusability feature.
- The sub class extends the properties of super classes to create more dominant objects.

Let us take very simple example of inheritance shown below. Here Programmer and Office_Boy are two classes which inherit properties from the class Employee. Both these class will have attributes like name, salary, date_of_joining along with their own attributes.



The syntax of a class declaration that inherits a super class is :

```
class subclass-name extends superclass-name
{
    // body of class
}
```

Point to Remember: A sub class can access all the **public** and **protected** members of a super class, not the private members. (By default, the variable or functions of a class are public in nature)

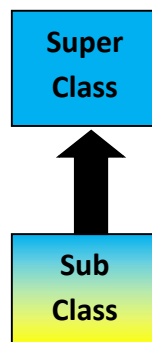
Types of Inheritance

Java supports following types of inheritance:

- Single Inheritance
- Hierarchical Inheritance
- Multiple Inheritance (not supported using class, but supported through interface)
- Multilevel Inheritance
- Hybrid Inheritance (may not be supported using class, but supported through interface)

Single Inheritance

In single inheritance, we have only one super class and one sub class. The subclass inherits properties from the only super class. A single inheritance can be shown as:

**Syntax of single inheritance:**

```
Class superClass
{
    // Body of super class
}
class subclass extends superClass
{
    // body of sub class
}
```

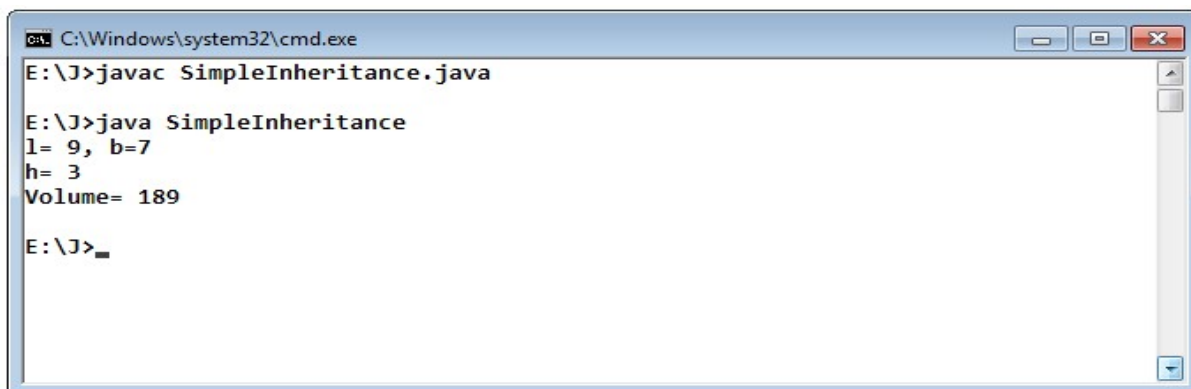
Program 2 Write a program to demonstrate single inheritance in java.

Solution:

```
class Rectangle
{
    int l, b;
    void disp_lb()
    {
        System.out.println("l= " +l+", b="+b);
    }
}
class Cuboid extends Rectangle
{
    int h;
    void disp_h()
    {
        System.out.println("h= " +h);
    }
    void find_volume()
    {
        System.out.println ("Volume= "+l*b*h);
    }
}

class SimpleInheritance
{
    public static void main(String args[])
    {
        Cuboid c=new Cuboid();
        c.l=9;
        c.b=7;
        c.h=3;
        c.disp_lb();
        c.disp_h();
        c.find_volume();
    }
}
```

Output



```
C:\Windows\system32\cmd.exe
E:\J>javac SimpleInheritance.java

E:\J>java SimpleInheritance
l= 9, b=7
h= 3
Volume= 189

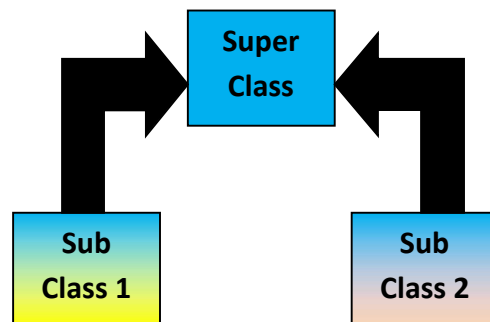
E:\J>
```

Explanation: Here sub class cuboid is able to access data l, b and method disp_lb() of super class Rectangle through single inheritance.

Hierarchical Inheritance

In hierarchical inheritance, two or more subclass inherits properties from a single super class. For example in a family, a son and a daughter of a single father inherit properties from the father. A hierarchical inheritance can be show as:

:



Syntax of hierarchical inheritance:

```
class superClass
{
    // Body of super class
}
class subclass1 extends superClass
{
    // body of sub class 1
}
class subclass2 extends superClass
{
    // body of sub class 2
}
```

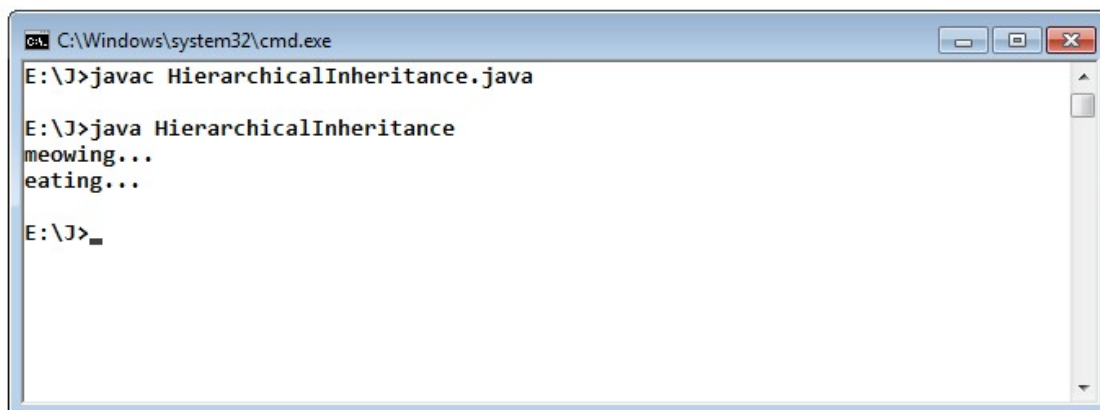
Program ② Write a program to demonstrate hierarchical inheritance in java.

Solution:

```
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 HierarchicalInheritance  
{  
    public static void main(String args[])  
    {  
        Cat c=new Cat();  
        c.meow();  
        c.eat();  
    }  
}
```

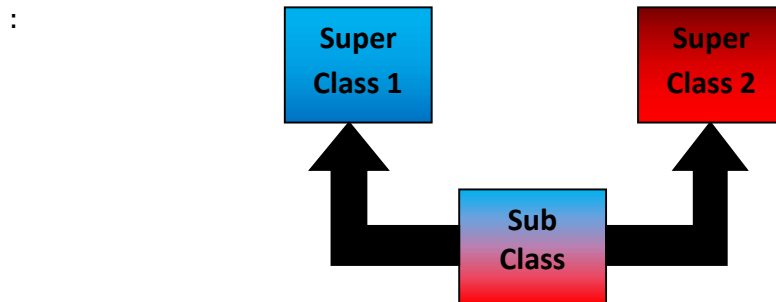
Output



```
C:\Windows\system32\cmd.exe  
E:\J>javac HierarchicalInheritance.java  
E:\J>java HierarchicalInheritance  
meowing...  
eating...  
E:\J>
```

Multiple Inheritance

In multiple inheritance, one subclass inherits properties from two or more super class. For example in a family, a father and mother having a, where the son inherits properties from both father and mother. A multiple inheritance can be show as:



Remember:

In Java, multiple inheritance is not supported using classes due to ambiguity problem. Ambiguity is a situation where compiler cannot predicate to inherit which field out of identical fields present in multiple super classes.

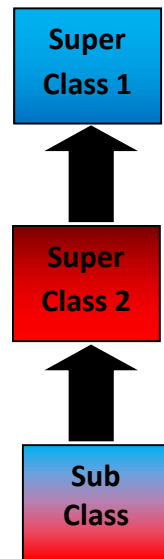
But same can be achieved using the concept of interface.

Therefore, following syntax is illegal in java:

```
class superClass1
{
    // Body of super class 1
}
class superClass2
{
    // Body of super class 2
}
class subClass extends superClass1, superClass2 //Illegal
{
    // body of sub class 2
}
```

Multi level Inheritance

In multi level inheritance, a number of classes are represented at different level as shown in the below diagram. For example, in a family, a father inherits properties from grandfather and son inherits properties from father.



Note: Here super class 2 inherits properties from super class 1 and subclass inherits properties from super class 2.

Syntax of multilevel inheritance:

```
class superClass1
{
    // Body of super class 1
}
class superClass2 extends superClass1
{
    // body of super class 2
}
class subClass extends superClass2
{
    // body of sub class
}
```

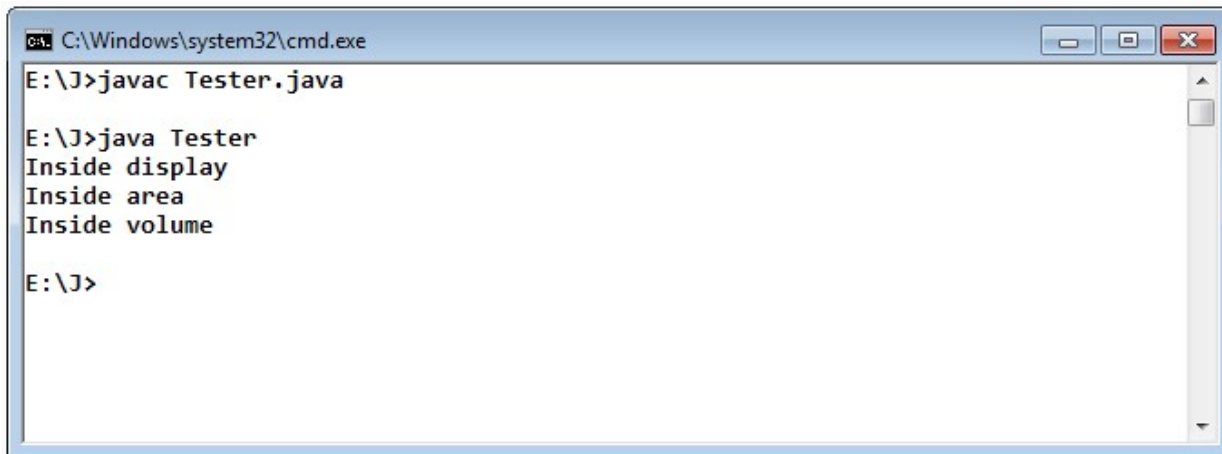
Program ③ Write a program to demonstrate multilevel inheritance in java.

Solution:

```
class Shape
{
    public void display()
    {
        System.out.println("Inside display");
    }
}
class Rectangle extends Shape
{
    public void area()
    {
```

```
        System.out.println("Inside area");
    }
}
class Cube extends Rectangle
{
    public void volume()
    {
        System.out.println("Inside volume");
    }
}
public class Tester
{
    public static void main(String[] arguments)
    {
        Cube cube = new Cube();
        cube.display();
        cube.area();
        cube.volume();
    }
}
```

Output



```
C:\Windows\system32\cmd.exe
E:\J>javac Tester.java

E:\J>java Tester
Inside display
Inside area
Inside volume

E:\J>
```

Hybrid Inheritance

It is the combination of all the above inheritance.

How constructor of a super class gets called ?

A subclass constructor can call or invoke the constructor of its super class in two ways:

- A subclass constructor invokes the default or non parameterized constructor of its super class implicitly or automatically.
- A subclass constructor can invoke the parameterized constructor of its super class explicitly by using the “super” statement.

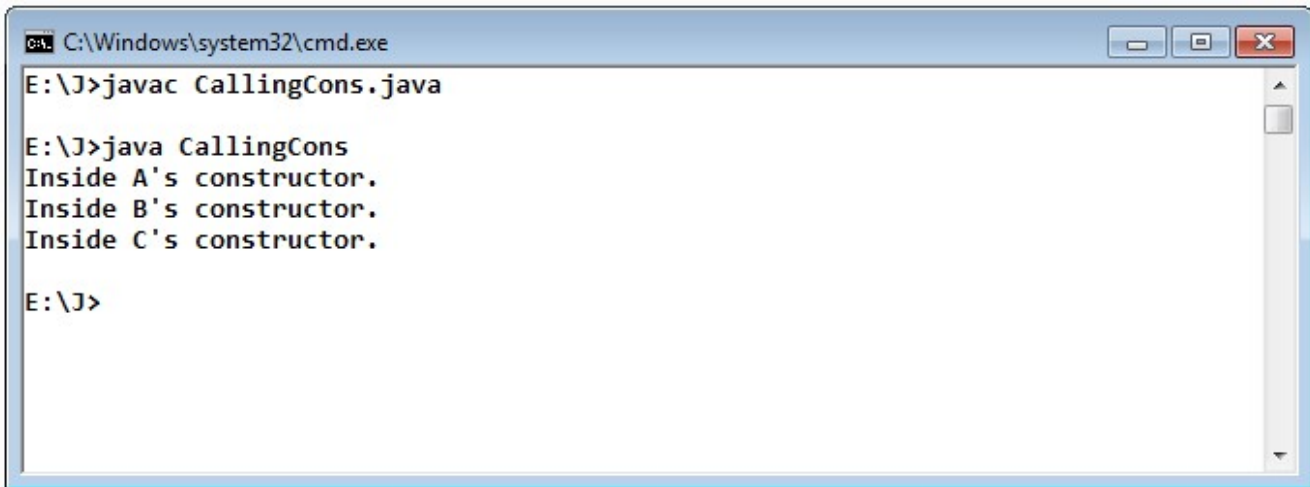
Note: In a class hierarchy, constructors are called in order of derivation, from super class to subclass. super() must be the first statement to be executed in a subclass constructor to call the parameterized constructor of super class.. If super() is not used, then the default constructor of super class will be called.

Program 4 Write a program to demonstrate subclass constructor invoking the constructor of the super class implicitly.

Solution:

```
class A
{
    A()
    {
        System.out.println("Inside A's constructor.");
    }
}
class B extends A
{
    B()
    {
        System.out.println("Inside B's constructor.");
    }
}
class C extends B
{
    C()
    {
        System.out.println("Inside C's constructor.");
    }
}
class CallingCons
{
    public static void main(String args[])
    {
        C c = new C();
    }
}
```

```
}
```

Output:

```
C:\Windows\system32\cmd.exe
E:\J>javac CallingCons.java
E:\J>java CallingCons
Inside A's constructor.
Inside B's constructor.
Inside C's constructor.
E:\J>
```

Program 5 Write a program to demonstrate subclass constructor invoking the parameterized constructor of the super class explicitly by using super keyword.

Solution:

```
class Rect
```

```
{
```

```
    int length, breadth;
```

```
    Rect (int l, int b)
```

```
    {
```

```
        length=l;
```

```
        breadth=b;
```

```
    }
```

```
}
```

```
class Cuboid extends Rect
```

```
{
```

```
    double height;
```

```
    Cuboid (int l, int b, int h)
```

```
    {
```

```
        super(l, b);
```

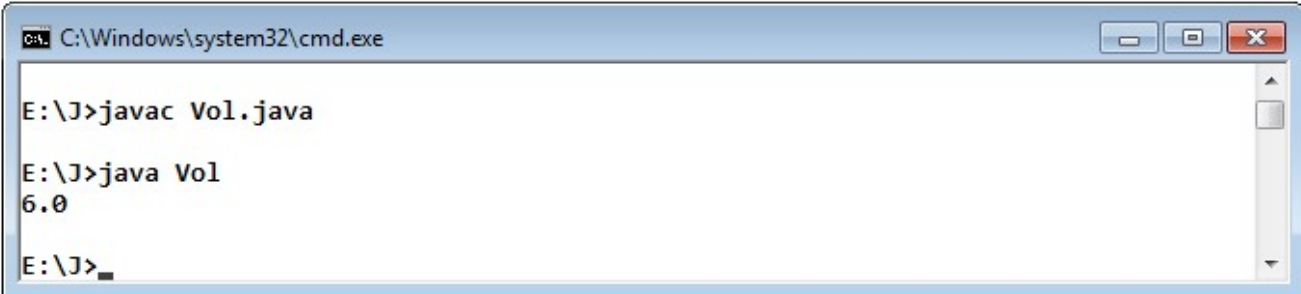
```
        height=h;
```

```
    }
```

```
    void volume()
```

```
    {
```

```
        System.out.println (length*breadth*height);
    }
}
class Vol
{
    public static void main(String args[]) {
        Cuboid v=new Cuboid(1, 2, 3);
        v.volume();
    }
}
```

Output:

The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The prompt is at "E:\J>". The user enters "javac Vol.java" and presses Enter. The prompt changes to "E:\J>". The user enters "java Vol" and presses Enter. The output "6.0" is displayed on the next line. The prompt returns to "E:\J>".

```
C:\Windows\system32\cmd.exe
E:\J>javac Vol.java
E:\J>java Vol
6.0
E:\J>
```

Review Questions of Chapter 5 (Inheritance)**1. Choose the most appropriate answer:****a) Advantage(s) of inheritance is/are**

- | | |
|---------------------|---|
| a) Code reusability | b) Hierarchical representation of classes |
| c) Both a) and b) | d) None |

b) Keyword used to achieve inheritance in java is

- | | |
|------------|-------------|
| a) extend | b) extends |
| c) inherit | d) inherits |

c) Which inheritance is not supported in java through classes?

- | | |
|-----------------|---------------|
| a) Single | b) Multiple |
| c) Hierarchical | d) Multilevel |

d) A system of three classes where two classes inherits properties from a class is known as _____ inheritance.

- | | |
|-----------------|-------------|
| a) Single | b) Multiple |
| c) Hierarchical | d) Hybrid |

e) Which keyword is used to call the parameterized constructor of a super class from sub class constructor?

- | | |
|----------|---------|
| a) final | b) sup |
| c) super | d) call |

f) The technique by which one class acquires the properties of another class is known as

- | | |
|------------------|---------------------|
| a) Inheritance | b) Polymorphism |
| c) Encapsulation | d) Data Aggregation |

Short Type**2. Answer briefly**

- Define inheritance?
- List different types of inheritance in java.
- Which inheritance is not supported in java & why?
- What is hierarchical inheritance in java?
- What is multilevel inheritance?

Long Type**3. Answer in details**

- a) Define inheritance. Explain different types of inheritance with their syntax.
- b) Explain how a sub class constructor can call the constructors of its super class
