JAVA

Class 23

Agenda

Types of Inheritance
Polymorphism in JAVA
Method Overloading
Method Overriding

Types of Inheritance

Based on number of ways inheriting the feature of **base class into derived class** we have 3 types of inheritance:

Single inheritance
Multilevel inheritance
Hierarchical inheritance

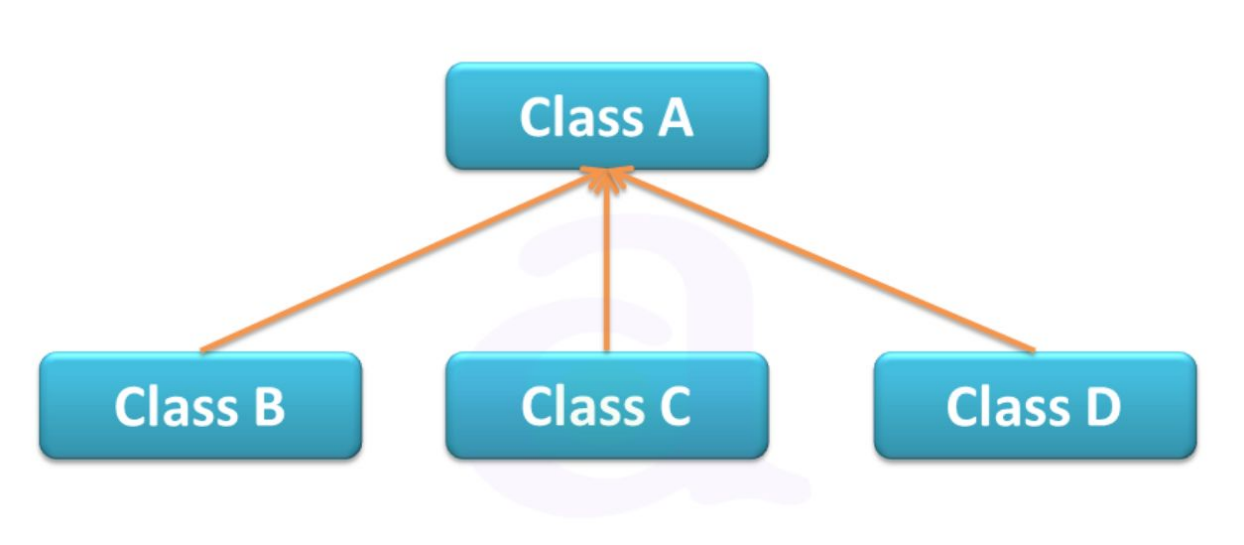
Single inheritance

In single inheritance there exists single base class and single derived class.

```
class Faculty {
   double salary=30000;
}

class Science extends Faculty {
  double bonus=2000;

  public static void main(String args[]) {
     Science obj=new Science();
     SOP("Salary is:"+obj.salary);
     SOP("Bonus is:"+obj.bonus);
  }
}
```

Hierarchical inheritance

When a class has more than one child classes (sub classes) or in other words more than one child classes have the same parent class then this type of inheritance is known as hierarchical inheritance.

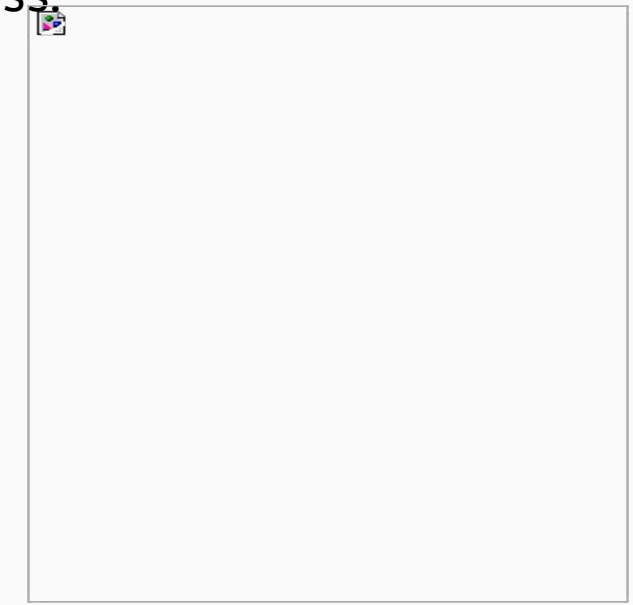


Multilevel inheritance

In Multilevel inheritances there exists single base class, single derived class and multiple intermediate base classes.

Single base class + single derived class + multiple intermediate base classes.

Intermediate base classes : An intermediate base class is one in one context with access derived class and in another context same class access base class.

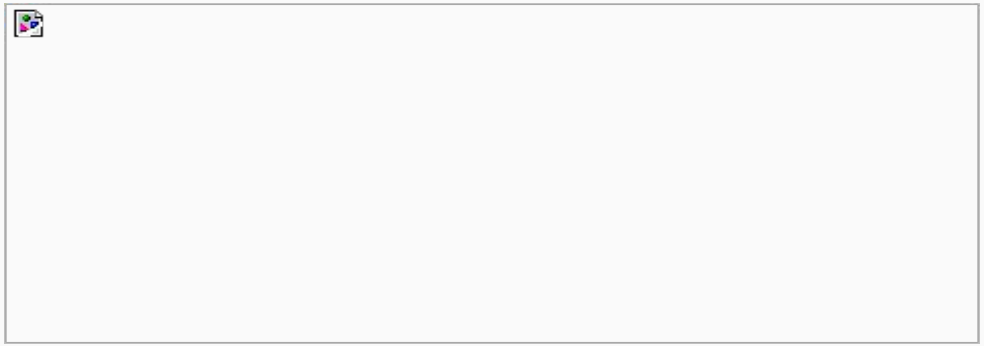


```
class Faculty {
        float total_sal=0, salary=30000;
}

class HRA extends Faculty {
        float hra=3000;
}

class DA extends HRA {
        float da=2000;
}

class Science extends DA {
        float bonus=2000;
        public static void main(String args[]) {
                Science obj=new Science();
                obj.total_sal=obj.salary+obj.hra+obj.da+obj.bonus;
                SOP("Total Salary is:"+obj.total_sal);
        }
}
```

All the above 3 inheritance types are supported by both classes and interfaces

Multiple inheritance

In multiple inheritance there exist multiple classes and single derived class. The concept of multiple inheritance is not supported in java through concept of classes but it can be supported through the concept of interface. 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 same method and you call it from child class object, there will be ambiguity to call method of A or B class.



Multiple inheritance

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 now.

```
class A{
   void msg(){SOP("Hello");}
   }
   class B{
   void msg(){SOP("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?
   }
   }
```

Important points for Inheritance:

Whenever we develop any inheritance application first create an object of bottom most derived class but not for top most base class.

When we create an object of bottom most derived class, first we get the memory space for the data members of top most base class, and then we get the memory space for data member of other bottom most derived class.

Bottom most derived class contains logical appearance for the data members of all top most base classes.

What is not possible in Inheritance?

1. Private members of the superclass are not inherited by the subclass and can only be indirectly accessed.

2. Members that have default accessibility in the superclass are also not inherited by subclasses in other packages, as these members are only accessible by their simple names in subclasses within the same package as the superclass.

3. Since constructors and initializer blocks are not members of a class, they are not inherited by a subclass.

4. A subclass can extend only one superclass

Polymorphism

Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms.

Polymorphism is the ability of an object to take on many forms.

The process of representing one form in multiple forms is known as Polymorphism.

Polymorphism

An important usage of polymorphism occurs in oops is how a parent class refers to a child class object.

Real life example of polymorphism
Suppose if you are in class room that time you behave like a student, when you are in market at that time you behave like a customer, when you at your home at that time you behave like a son or daughter, Here one person present in different-different behaviors.

Polymorphism

Polymorphism can be achieved in two of the following ways:

1. Compile time Polymorphism/Method Overloading/Static Binding

1. Run time Polymorphism/Method Overriding/ Dynamic Binding

Compile / Static

Compile time polymorphism can be achieved by using Method overloading

Whenever same method name is exiting multiple times in the same class with different number of parameter or different order of parameters or different types of parameters is known as method overloading

Java does **not allow overloading by changing the return type**, though overloaded methods can change the return type.

Method Overloading

Syntax:

```
class  class_Name {

    Returntype  method() {
    .....}

    Returntype  method (datatype1 variable1) {
    .....}

}
```

In java, method overloading is not possible by changing the return type of the method

Method Overloading

There are two ways to overload the method in java:

1. By changing number of arguments
2. By changing the data type

Different ways to overload the method

By changing number of arguments
In this example, we have created two overloaded methods, first sum method performs addition of two numbers and second sum method performs addition of three numbers.

```
class Addition
{
  void sum(int a, int b) {
   SOP(a+b);
  }
  void sum(int a, int b, int c) {
  SOP(a+b+c);
 }
public static void main(String args[]) {
Addition obj=new Addition();
obj.sum(10, 20);
obj.sum(10, 20, 30);
}
}
```

By changing the data type
In this example, we have created two overloaded methods that differs in data type. The first sum method receives two integer arguments and second sum method receives two float arguments.

```
class Addition
{
 void sum(int a, int b) {
 SOP(a+b);
}
 void sum(float a, float b) {
 SOP(a+b);
}
public static void main(String args[]) {
Addition obj=new Addition();
obj.sum(10, 20);
obj.sum(10.05, 15.20);
}
}
```

Task

Create 1 class in which create a methods that will calculate the area of

- Rectangle
- Square
- Box

Use separate class to test your code

Advantage of Method Overloading

- The main advantage of this is cleanliness of code.
- Method overloading increases the readability of the program.
- Flexibility
- Overloaded methods give programmers the flexibility to call a similar method for different types of data. If you are working on a mathematics program, for example, you could use overloading to create several multiply classes, each of which multiplies a different number of type of argument: the simplest multiply(int a, int b) multiplies two integers; the more complicated method multiply(double a, int b, int c) multiplies one double by two integers -- you could then call "multiply" on any combination of variables that you created an overloaded method for and receive the proper result.