International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 12 | Dec 2018 www.irjet.net p-ISSN: 2395-0072

Inheritance in Java

Mrs. Kanchanmala D Talekar[1], Mr. Mithun Jadhav[2]

[1]Lecturer, Dept. of computer Engineering, GVAP Shelu, Maharashtra, India.
[2]HOD, Dept. of computer Engineering, GVAP Shelu, Maharashtra, India

---

1. INTRODUCTION

It is always nice if we could reuse something that already exists rather than creating the same all over again and again. Java supports this concept. Java classes can be reused in several ways. This is basically done by creating new classes, reusing the properties of existing ones. The mechanism of deriving a new class from an old one is called Inheritance.

The old class is known as base class or super class or parent class. The new class is called the derived class or subclass or child class. Inheritance in java is a mechanism in which one object acquires all the properties and behaviors of parent object. The idea behind inheritance is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of parent class and you can add new methods and fields also. Inheritance represents the IS-A relationship, also known as parent-child relationship.

Advantages of Inheritance:

1. For Method Overriding: so runtime polymorphism can be achieved.
2. For Code Reusability.

Syntax of java Inheritance:

```
class Superclass_Name{
        //methods and fields
}
class Subclass_Name extends Superclass_Name
{       //methods and fields
}
```

The extends keyword indicates that you are making new class that derives from an existing class.

Types of Inheritance:

There are following types of Inheritance:

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance
4. Hybrid Inheritance

Super Keyword:

- Super keyword can be used to refer immediate parent class instance variable.
- Super keyword can be used to invoke immediate parent class method.
- Super() can be used to invoke immediate parent class Constructor.

1.1 Single Inheritance:

Single inheritance is damn easy to understand. When a class extends another one class only then we call it a single inheritance. The below flow diagram shows that class B extends only one class which is A. Here A is a **parent class** of B and B would be a **child class** of A.

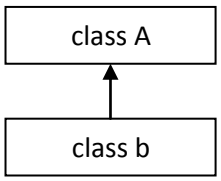

Fig1. Structure of Single Inheritance

Syntax:

```
class A{
        //code
    }
class B  extends  A  {
        //code
        }
```

Example 1:

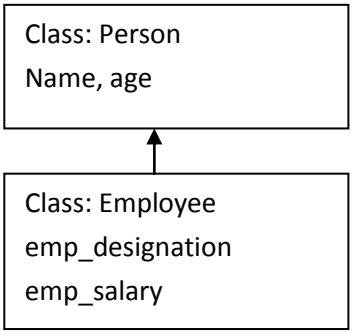

Fig 2. Example of Single Inheritance

```
class Person{
String name;
int age;
Person(String n,int a){
name=n;
age=a;
}
void display(){
System.out.println("\n Name ="+name);
System.out.println("\n Age="+age);
}
}
class Employee extends Person{
String emp_designation;
float emp_salary;
Employee(String n,int a,String d,float s)  {
super(n,a);
emp_designation=d;
emp_salary=s;
}
void display(){
super.display();
System.out.println("\n Employee designation="+emp_designation);
System.out.println("\n Employee salary="+emp_salary);
}
}
public class SingleInheritance{
   public static void main(String args[]){
Employee e=new Employee("Anushree",26,"Developer",35000);
e.display();
}
}
```

Output:

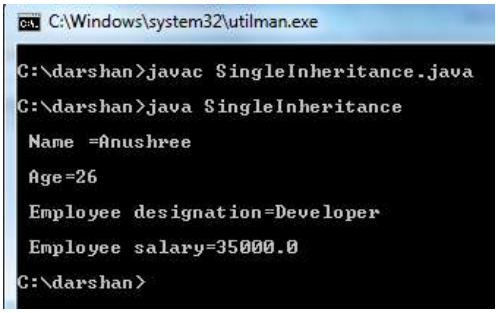

Fig 3: Output of Single Inheritance

1.2 Multilevel Inheritance

Multilevel inheritance refers to a mechanism in OO technology where one can inherit from a derived class, thereby making this derived class the base class for the new class. As you can see in below flow diagram C is subclass or child class of B and B is a child class of A. For more details and example refer – Multilevel inheritance in Java.

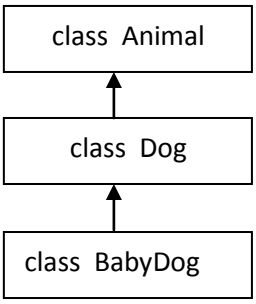

Fig 4: Structure of Multilevel Inheritance

Syntax:

```
class A  {
          //code
}
class B extends  A  {
  //code
}
class C extends A  {
         //code
}
```

Example:

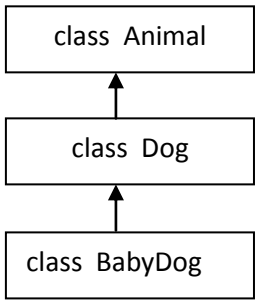

Fig 5: Example of Multilevel Inheritance

Example:

```
class Animal{
void eat(){
System.out.println("\n Eating.....");
}
}
class Dog extends Animal{
void bark(){
System.out.println("\n Barking.....");
}
}
class BabyDog extends Dog{
void weep(){
System.out.println("\n weeping.....");
}
}
class MultipleInheritance{
 public static void main(String args[]){
BabyDog b1=new BabyDog();
b1.weep();
b1.bark();
b1.eat();
}
}
```

Output:

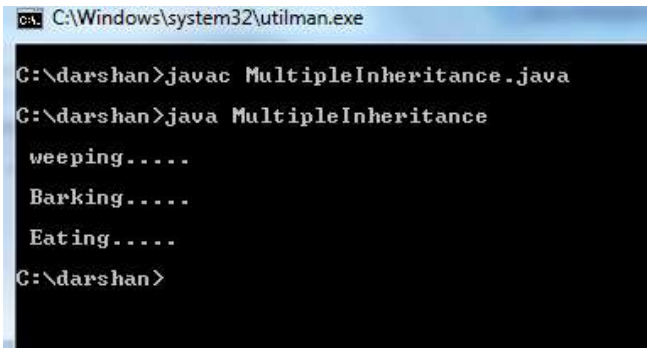

Fig 6: Output of Multilevel Inheritance

1.3 Hierarchical Inheritance

When more than one classes inherits a same class then this is called hierarchical inheritance

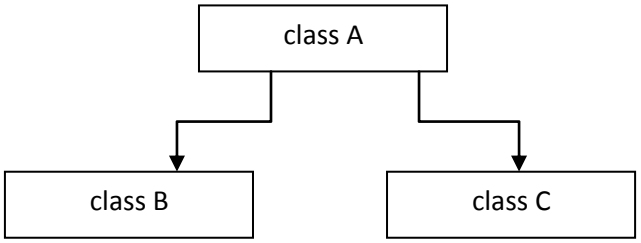

Fig 7: Structure of Hierarchical Inheritance

Syntax:

```
class A{
//code
}
class B extends A{
//code
}
class c extends A{
        //code
}
```

Example:

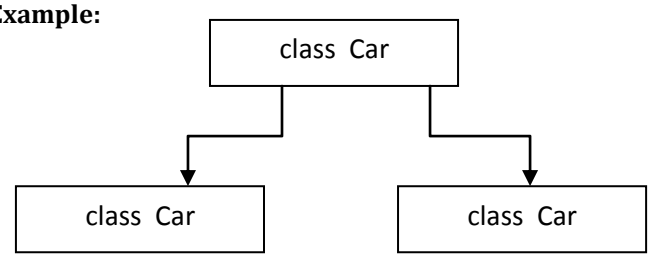

Fig 8: Example of Hierarchical Inheritance

```
class Car{
public Car()  {
System.out.println("Class Car");
 }
public void vehicleType()  {
```

```
System.out.println("Vehicle Type: Car");
}
}
class Maruti extends Car{
public Maruti() {
System.out.println("Class Maruti");
}
public void brand()  {
System.out.println("Brand: Maruti");
}
public void speed()  {
System.out.println("Max: 90Kmph");
}
}
class Maruti800 extends Maruti{
public Maruti800()  {
System.out.println("Maruti Model: 800");
}
public void speed()  {
System.out.println("Max: 80Kmph");
}
}
public class Hirarchical_Inheritance{
public static void main(String args[])
        {
                Maruti800 obj=new Maruti800();
                obj.vehicleType();
                obj.brand();
                obj.speed();
        }
       }
```

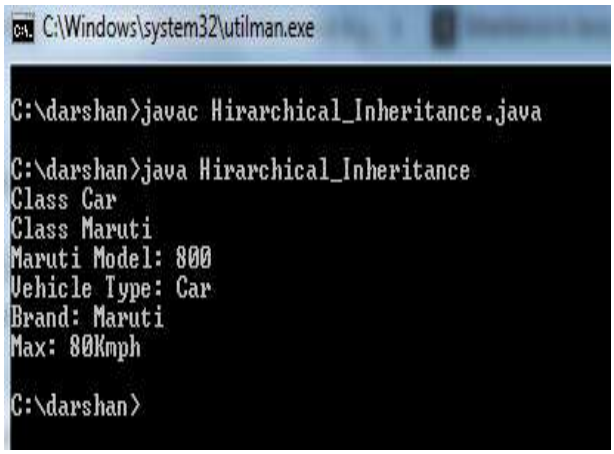

Fig 9: output of Hierarchical Inheritance

1.4 Multiple Inheritance

"**Multiple Inheritance**" refers to the concept of one class extending (Or inherits) more than one base class. The inheritance we learnt earlier had the concept of one base class or parent. The problem with "multiple inheritance" is that the derived class will have to manage the dependency on two base classes.

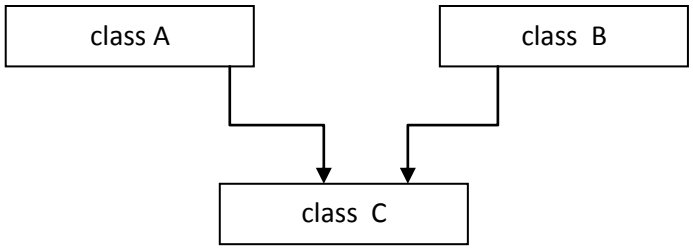

Fig 10: Structure of Multiple Inheritance

Note 1: Multiple Inheritances is very rarely used in software projects. Using multiple inheritance often leads to problems in the hierarchy. This results in unwanted complexity when further extending the class.

Note 2: Most of the new OO languages like Small Talk, Java, and C # do not support Multiple inheritance**.** Multiple Inheritances is supported in C++.

1.5 Hybrid Inheritance

In simple terms you can say that Hybrid inheritance is a combination of **Single** and **Multiple inheritance.** A typical flow diagram would look like below. A hybrid inheritance can be achieved in the java in a same way as multiple inheritance can be!! Using interfaces. yes you heard it right. By using **interfaces** you can have multiple as well as **hybrid inheritance** in Java.

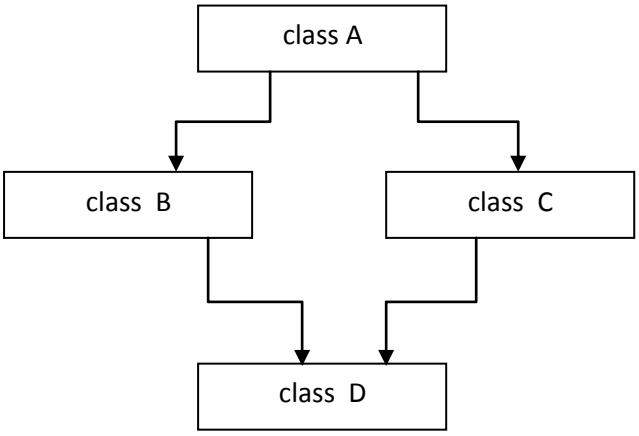

Fig 10: Structure of Multiple Inheritance

REFERENCES

1. https://beginnersbook.com
2. https://www.javatpoint.com