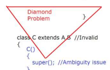
Program on Multilevel Inheritance :

```
 public class MultiLevelDemo
 {
      public static void main(String[] args)
      {
           SportsCar car = new SportsCar("Mustang",330, 2, false, 5000);
           IO.println(car);
      }
 }

 class Vehicle
 {
      protected String brand;
      protected int speed;

    public Vehicle(String brand, int speed)
      {
           this.brand = brand;
           this.speed = speed;
      }

      public String toString()
      {
           return "[Vehicle brand is "+this.brand+", Vehicle Speed is "+this.speed+"]";
      }
 }

 class Car extends Vehicle
 {
      protected int noOfDoors;
      protected boolean isEV;

      public Car(String brand, int speed, int noOfDoors, boolean isEV)
      {
           super(brand, speed);
           this.noOfDoors = noOfDoors;
           this.isEV = isEV;
      }

   public String toString()
      {
           return super.toString()+"[Car has "+this.noOfDoors+" doors, Is Car EV "+this.isEV+"]";
      }
 }
 class SportsCar extends Car
 {
      protected int horsePower;
    public SportsCar(String brand, int speed, int noOfDoors, boolean isEV, int horsePower)
      {
           super(brand, speed, noOfDoors, isEV);
           this.horsePower = horsePower;
      }

      public String toString()
      {
           return super.toString()+"[SpotsCar has "+this.horsePower+"]";
      }
 }
```

****Why Java does not support multiple Inheritance using classes ?**

```
               class A                              class B
               {                                    {

               }                                    }

                          class C extends A,B  //Invalid
                          {
                             C()
                             {
                               super();  //Ambiguity issue  [There is a ambiguity that this super()
                             }                                 will call which class default no argument
                                                               constructor (either A OR B)]
                          }
```



Multiple Inheritance is a situation where a sub class wants to inherit the properties two or more than two super classes.

In every constructor we have super() or this() to the first line. When compiler will add super() to the first line of the constructor then we have an ambiguity issue that super() will call which super class constructor as shown in the diagram [either A class OR B class]

It is also known as Diamond Problem in java so the final conclusion is we can't achieve multiple inheritance using classes but same we can achieve by using interface [interface does not contain any constructor]

Assignment :

```
//Program on Hybrid Inheritance

class Vehicle
{
}
class Car extends Vehicle
{}
Maruti  and Ford
```

JVM Architecture with class loader subsystem :
----

* The entire JVM Architecture is divided into 3 sections:

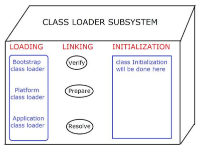
  1) Class Loader subsystem
  2) Runtime Data Areas (Memory Areas)
  3) Execution Engine [Interpreter + JIT Compiler]

1) Class Loader subsystem :
----

* The main purpose of Class Loader subsystem to **load the java .class files into JVM memory** from various location.
[All the static fields are initialized with default value and memory will be allocated at the time of loading the .class file into JVM memory]

* In order to load the .class file into JVM memory, Class loader subsystem, internally using an algorithm called "**Delegation Hierarchy Algorithm**".

* A class loader subsystem internally performs the following tasks :
  a) LOADING   b) LINKING   c) INITIALIZATION



Bootstrap class Loader :
