**Class:**

Clas is a model

Class is blue print

Class is logical entity

Class is plan

Class is templet

**Syntax:**

Class className

{

Members;

}

**Object:**

Object is a physical entity….

Creation of an object:

className objectName=new ClassName();

**Constructor**

To initialize the data fields of a class there are basically two ways

1. Class Student

{ int age=20;}

**2.using** Constructors

* Constructor is a special method which name is same as class name
* Constructor doesn’t have return type even “void”
* Constructor is used to initialize the class members
* Constructor will be invoked automatically at the time of creating an object

**Types of Constructors**

1.Default Constructor: The Constructor which doesn’t accepts arguments

Drawback: The initial value of the speicfied datafield having the same initial value for different object of the class

To overcome this problem we hava to go for PC

2.parametrized Constructors:

The Constructor which accepts arguments

3.Copy Constructor

It is used to copy the one object Data into other object data fields

**ABSTRACT CLASS:**

Abstract Method:

 **A Method which contains only declaration and does not contain any functionality is known as abstract Method…**

 **An abstract Method is declared using abstract keyword…**

 **An abstract Method can be declared only within abstract class…**

 **If an abstract Method declare in non abstract class. Then compilation error occurs…**

**ABSTRACT CLASS:**

 **A class which contains one or more abstract Methods is known as abstract class**

 **Abstract class should be declared using abstract keyword…**

 **An abstract class can not be Instantioated directly…**

 **An abstract class can contain both abstract and non abstract Methods**

 **An abstract class should be using to derive new class…**

abstract class Car

{

public void Wheels()

{

System.out.println("There are 4 wheels");

}

public void Doors()

{

System.out.println("There are 4 Doors");

}

abstract public void Windows() ;

}

class Nano extends Car

{

public void Extra()

{

System.out.println("Air Conditioner Available");

}

public void Windows()

{

System.out.println("Normal windows");

}

}

class Audi extends Car

{

public void Extra()

{

System.out.println("Air Baloons Available");

}

public void Windows()

{

System.out.println("Power windows");

}

}

class AbsDemo

{

public static void main(String args[])

{

Nano n=new Nano();

Audi a=new Audi();

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.out.println("Nano Details");

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

n.Wheels();

n.Doors();

n.Extra();

n.Windows();

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.out.println("Audi Details");

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

a.Wheels();

a.Doors();

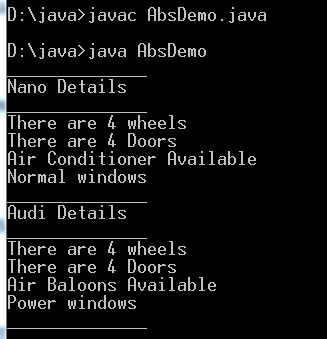
a.Extra();

a.Windows();

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

}



**an interface…**

 **A class which contains all abstract methods is known as an interface…**

 **An interface is declared using interface keyword…**

 **An interface can’t be instantiated directly…**

 **Interface is using to implement multiple inheritances**

 **By default all members of interface are public. All methods are abstract…**

interface Inf1

{

void Show();

void Test();

}

class Cls **implements** Inf1

{

public void Show()

{

System.out.println("This is Show Method from Cls");

}

public void Test()

{

System.out.println("This is Test Method from Cls");

}

}

class InfDemo

{

public static void main(String args[])

{

Cls c=new Cls();

c.Show();

c.Test();

}

}

Implementation of multiple inheritance using Interface

interface Father

{

float HT=6.5f;

void Show();

}

interface Mother

{

float HT=6.0f;

void Show();

}

class Child implements Father,Mother

{

float CHT;

public void Show()

{

CHT=(Father.HT+Mother.HT)/2;

System.out.println("CHild Hight : "+CHT);

}

}

class InfDemoInh

{

public static void main(String args[])

{

Child c=new Child();

c.Show();

}

}

interface inf1

{

void show();

}

abstract class class1 impliments inf1

{

}

class Demo

{

public static void main(String args[])

{

}

}

