Chapter ✱ INHERITANCE ✱

P55

```
1) Class A
{
    static int j=90;
    int i=12;
    double d=8.3;
}
Class B
{
    static int j=2348;
    int k=67;
    float f = 23.43f;
}
Class Run1
{
    psvm ( )
    {
     Sop("Program starts...");
     A a1 = new A();
     Sop ("i="+a1.i);
     Sop ("d=" + a1.d);
     Sop ("j=" + a1.j);
     Sop ("----------");
     B b1 = new B();
     Sop ("k=" + b1.k);
     Sop ("f=" + b1.f);
     Sop ("j=" + b1.j);
     Sop ("Program ends...")
    }
}
```

O/P:

```
Program starts...
i=12
d=8.3
j=90
--------
```

1✱) Inheriting a member of one class to another class is known as Inheritance

2✱) The class from where members are inherited are known as Super class / parent class / base class.

3✱) The class to which members are inherited are known as Sub class / child class / derived class.

4✱) A sub class can inherit a super class by using 'extends' keyword

Syntax :-

```
Class SubClassName extends SuperClass
{
    ____
    ____
    ____
}
```

5✱) The inheritance always happens from super class to SubClass.

6✱) Only non-static members of SuperClass are inherited to Sub Class.

7✱) The static member of SuperClass will not be inherited to SubClass because :-

NOTE:

- ✱ The static members will be loaded into static pool at the time of Classloading.
- ✱ Static members will not be loaded into the Object.

O/p continued...

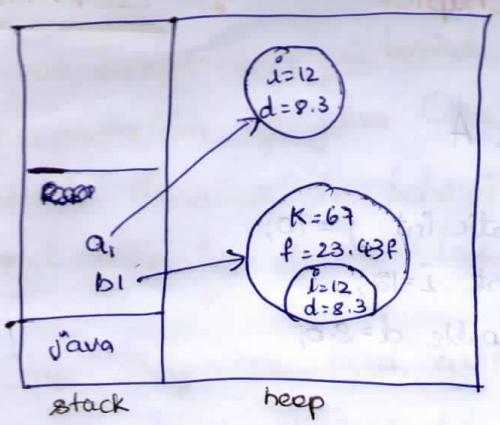
```
k=67
f=23.4
j=2348
Program ends...
```

2) Class A

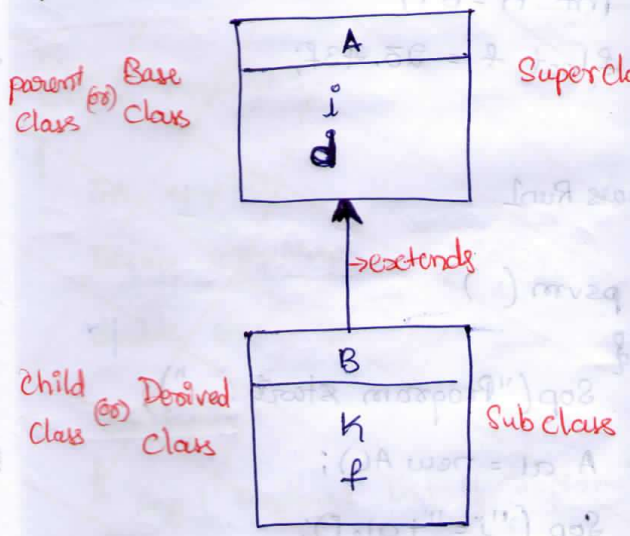
```
{
    int i=12;
    double d=8.3;
}

Class B extends A
{
    int K=67;
    float f=23.43f;
}

Class Run1
{
    psvm ( )
    {
    Sop("Program starts...");
    A a1 = new A();
    Sop("i=" + a1.i);
    Sop("d=" + a1.d);
    Sop("----------");
    B b1 = new B();
    Sop("K=" + b1.K);
    Sop("f=" + b1.f);
    Sop("i=" + b1.i);
    Sop("d=" + b1.d);
    Sop("Program ends...");
    }
}
```

O/p: Program starts...
i = 12
d = 8.3

K = 67
f = 23.43
i = 12
d = 8.3
Program ends...



9*) Pictorial Representation of Inheritance



3) P57

```
class C
{
  int i=78;
}
Class D extends C
{
  double d=12.34;
}
Class E extends D
{
  float f = 324.0f;
}

Class Run2
{
   psvm ( )
   {
     Sop ("Program starts...");
     D d1 = new D( );
     Sop ("i = " + d1.i);
     Sop ("d = " + d1.d);
     Sop ("_ _ _ _ _ _ _ _ _");
     E e1 = new E ( );
     Sop ("i = " + e1.i);
     Sop ("d = " + e1.d);
     Sop ("f = " + e1.f);
     Sop ("program ends...");
   }
}
```

O/P:

```
Program starts...
i = 78
d = 12.34
_ _ _ _ _ _ _ _ _ _
i = 78
d = 12.34
f = 324.0f
Program ends...
```

Assignment 30-01-2013

P58

```
class Employee
{
    int empID;
    String empName;
    double empSalary;
    Employee()
    {
        Sop("running default constructor");
        this.empID = 000;
        this.empName = "noname";
        this.empSalary = 0.0;
    }
    Employee(int empID, String empName, double empSalary)
    {
        Sop("running 3 arg constructor");
        this.empID = empID;
        this.empName = empName;
        this.empSalary = empSalary;
    }
    int getEmpID()
    {
        return this.empID;
    }
    String getEmpName()
    {
        return this.empName;
    }
    double getEmpSalary()
    {
        return this.empSalary;
    }
```

O/P:

```
Program starts...
running default constructor
Emp ID : 12012
Emp Name : Ramesh
Emp Salary : 23455.87
Program ends...
```

```
void setempID (int empID)
{
    this.empID = empID;
}
void setempSalary(double empSalary)
{
    this.empSalary = empSalary;
}
void setEmpName(String empName)
{
    this.empName = empName;
}
Class CreateEmployee
{
    psvm()
    {
        Sop("Program starts...");
        Employee empl = new Employee(12012, "Ramesh", 23455.87);
        Sop("Emp ID:" + empl.getEmpID());
        Sop("Emp Name:" + empl.getEmpName());
        Sop("Emp Salary:" + empl.getEmpSalary());
        Sop("Program ends...");
    }
}
```

31-01-2013 Thursday

Constructor Chain

4) P59

```
class F
{
    F()
    {
      Sop("running F() constructor...");
    }
}
class Gr extends F
{
    Gr()
    { super();
      Sop("running Gr() constructor...");
    }
}
Class H extends Gr
{
    H()
    { super();
      Sop("running H() constructor...");
    }
}
Class Run3
{
  psvm()
  { Sop("Program starts...");
    H h1 = new H();
    Sop("Program ends...");
  }
}
```

O/P: Program starts...
Running F() constructor...
Running Gr() constructor...
Running H() constructor...
Program ends...

1*) In inheritance, when an object of sub class is created, the constructor of sub class calls the constructor of super class.

2*) The super class constructor calls its super class constructor this is known as Constructor Chain

3*) Inheritance happens only if the constructor chain happens

4*) The chain of constructor is implicitly done by compiler by using a statement called super() statement.
The super() statement calls the constructor of super class

5*) Whenever compiler makes an implicit call to super class, it always calls to default constructor of super class.

6*) If super class doesn't have default constructor then sub class constructor should explicitly calls parametrized constructor of Super Class.

7*) The Super() statement has to be written in the first line of constructor body.
Inside constructor body

5) P60

```
class F
{
   F()
    {
      super(); //calls
      Sop("running F() constructor...");
    }
}
class Gr extends F
{
    Gr(int a)
    {
      Sop("running Gr() constructor...");
    }
}
class  H extends Gr
{
   H()
    {
      super(12);
      Sop("running H() constructor...");
    }
}

Class Run3
{
    psvm ( )
     {
       Sop ("Program starts...");
       H h1 = new H ( );
       Sop ("Program ends...");
     }
}
```

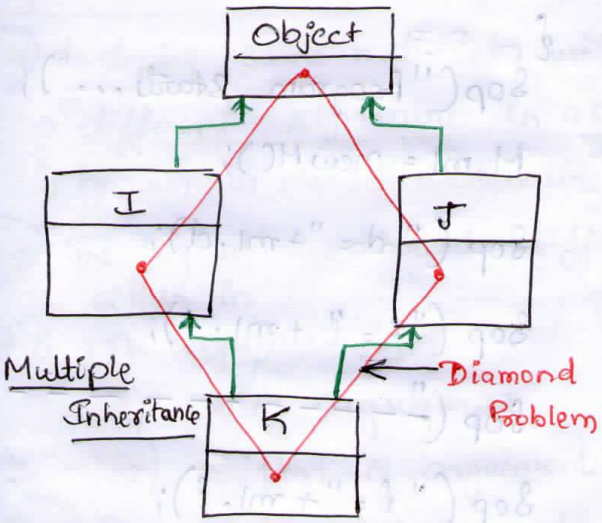
O/P: Program starts...
running F() constructor...
running Gr() constructor...
running H() constructor
Program ends.

we cannot write both this() & super() statement. either one of statement should be written

8*) Every java class should inherit from Object class.

Object class is supermost class in java.

9*) Diamond Problem



Example

```
Class I
{
}
class J
{
}
class K extends I, J
{   //error, multiple inheritance
    not supposted in java using Class
}
```

6) P61

```
class L
{
    static int i=12;
           double d=23.45;
}
class M extends L
{
    float f=12.34f;
}
class Run5
{
    psvm( )
    {
     Sop("Program starts...");
     M m1 = new M();
     Sop ("d = "+m1.d);
     Sop ("f = " +m1.f);
     Sop ("----------");
     Sop ("i = "+m1.i);
     Sop ("Program ends...");
    }
}
```

O/P:

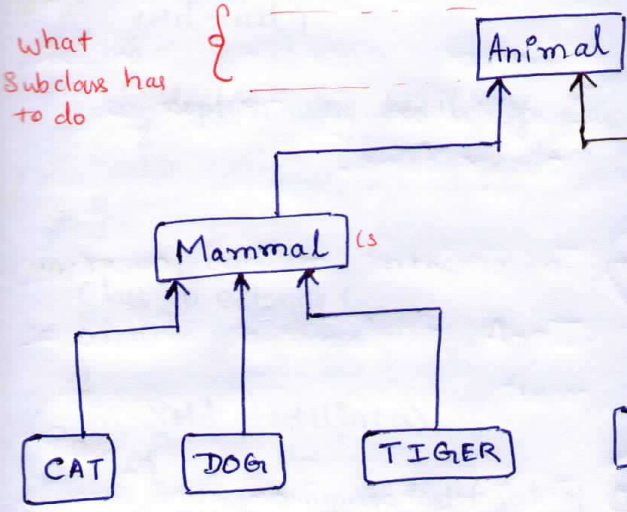
10) Inheriting more than one class members at a time is known as multiple inheritance.

Java doesn't support multiple inheritance using class because subclass constructor cannot make call to more than one super class. Also Object class member cannot be inherited to single subclass in more than one path.

01-02-2013 | Friday

* method - behaviour
* variables - fields

what Subclass has to do { _____ _____



* Subclass is a type of Superclass
* Cat Is-A type of Animal
* Cat Is-A type of Mammal
* Snake Is-A type of Reptile
* Snake Is A type of Animal

P68

METHODS | OverLoading

7)

```
class A
{
3*  void test1()
    {
      Sop("running test1() method of class A");
    }

    void test1(int a)
    {
      Sop("running test1(int) method of Class A");
    }
    Void test1(double a)
    {
      Sop("running test1(double) method of Class A");
    }
14* }
}

Class Run1
{
  psvm()
  {
    Sop("Program starts...");
```

1 *) developing same methods() with a different signature in a class is known as Method overloading.

2 *) The signature should differ either in
- 2.1) argument type
- 2.2) No of arguments
- 2.3) position of arguments.

3 *) Both static methods and non-static methods of a class can be overloaded in the class.

4 *) A super class method can be overloaded in sub class

5 *) A method overloading should be done whenever same operation has to implemented based on arguments.

6 *) The jvm executes the method based on arguments value.

```
A a1 = new A();
a1.test1();
a1.test1(12);
a1.test1(12.34);
Sop("Program ends...");
```

O/P:

```
Program starts...
running test1() method of class A
running test1(int) method of ClassA
running test1(double) method of ClassA
Program ends...
```

---

8) Class B P63

```
{
  static void test1()
  {
    Sop("running test1() method of classA");
  }
  static void test1(int a)
  {
    Sop("running test1(int) method of ClassA");
  }
  static void test1(double a)
  {
    Sop("running test1(double) method of ClassA");
  }
```

15*

26 *

9) P64

```
Class C
{
   void test1()
   {
     Sop("running test1() of class C");
   }
}
Class D extends C
{
    void test1(int a)
    {
      Sop("running test1(int of Class D");
    }
}

Class Run3
{
    psvm()
   {
    Sop ("Program starts...");
    D d1 = new D();
    d1. test1(); // inherited member
    d1. test1(123); // derived member
    Sop("Program ends...");
    }
}
```

O/P:

```
Program starts...
running test1() of class C...
running test1(int) of class D...
Program ends...
```

10)

```
class C
{ //overridden method
    void test1()
    {
      Sop("running test1() of class C");
    }
}

class D extends C
{
  //overridded method
   void test1()
   {
     Sop("running test1() of class D");
   }

class Run3
{
    psvm()
    {
      Sop("Program start ...");

      D d1 = new D();
      d1.test1();

      Sop ("Program ends ...");
    }
}
```

O/P:

```
Program starts ...
running test1() of class D
Program ends ...
```

1.* Inheriting the behavior of superclass and changing the behavior according subclass specification is known as overriding.

2.* To perform method overriding Is-A relationship should be must.

3.* The method signature in the subclass should be same as superclass type.

4.* Sub class should provide a diff implementation.

5.* whenever an object of subclass is created for overridded method sub class implementation will be available.

6.* We cannot override static methods of a super class because it will not inherited to sub class object.

7.* The super class method which is overrided in subclass is called as overridden method the same method in a sub class is called as overridded method.