```
    putm();
    System.out.println("spwt=" +spwt);
    total=m1+m2+spwt;
    System.out.println("Total=" +total);
    }
  }
 class minherit
   {
   public static void main(String args[])
     {
     sports s = new sports();
     s.getdata();
     s.getm();
     s.putwt();
     }
  }
```

Method overriding

When a method in a sub class has the same name and type signature as a method in its super class then the method in sub class said to override the method in the superclass. When an overridden method is called from within a subclass, it will always refer to the version of that method defined by the subclass.

Example :

```
class A
  {
  int i,j;
  A(int a,int b)
   {
   i=a;
   j=b;
   }
  void show()
   {
   System.out.println("i and j=" + i+" " +j);
   }
  }
class B extends A
  {
  int k;
  B(int a,int b,int c)
   {
   super(a,b);
   k=c;
   }
  void show()
   {
   super.show(); // super prevents override
```

```
        System.out.println("k=" +k);
       }
     }
     class override
       {
       public static void main(String args[])
        {
        B subob=new B(1,2,3);
        subob.show();
        }
      }
```

Abstract method

An abstract method represents undefined methods also known as unimplemented methods.

eg, abstract void getdata();

Abstract class

A class is said to be abstract class iff it contains at least one abstract method.

Or

A class is said to be abstract class iff it contains some concrete methods and some abstract methods. An abstract class is a class that can only be subclassed. It can't be instantiated.

Concrete Methods : A concrete methods represent defined methods.

eg,

```
abstract class Area
     {
      abstract void area();
     }
class Rectangle extends Area
   {
   public void area()
     {
     int length=10;
     int breadth=20;
     System.out.println("Area of rectangle =" + (length*breadth));
     }
 }
class Circle extends Area
   {
   public void area()
     {
     int radius=20;
     System.out.println("Area of circle is :" + 3.14*radius*radius);
     }
 }
class AbstDemo
```

```
  {
  public static void main(String args[])
   {
   Rectangle r=new Rectangle();
   r.area();
   Circle c=new Circle();
 c.area();
}
}
```

Interface

An interface is a collection of abstract methods and static final data members that can be implemented in by the any number of classes residing in a class hierarchy. Interface is known as a prototype for a class. Methods defined in an interface are only abstract methods.

Syntax :

```
interface <interface_name>
    {
     ............
    static final data member;
    return_type public methods(parameter);
    }

class <class_name> extends [super_class_name] implements [interface name]
    {
     .....
     ....
    }
```

Note : with respect to interface we can't instantiate a object directly.

eg,

```
class student
  {
  String stu_name;
  float fees;
  void getdata()
    {
    stu_name="Amit kumar";
    fees=1200.56f;
    }
  void display()
   {
  System.out.println("Name :" +stu_name);
  System.out.println("Fees :" +fees);
  }
}
class marks extends student
  {
```

```
  int m1,m2;
  void getm()
    {
    m1=20;
    m2=30;
    }
   void putm()
    {
    System.out.println("m1=" +m1);
    System.out.println("m2=" +m2);
    }
  }
interface sports
  {
  int spwt=30;
  void putsp();
  }
class result extends marks implements sports
   {
   int total;
   public void putsp()
    {
    total=m1+m2+spwt;
    display();
    putm();
    System.out.println("result=" +total);
   }
}
class face1
  {
public static void main(String args[])
  {
result r=new result();
r.getdata();
r.getm();
r.putsp();
}
}
```