

## Chapter 4

## CLASS and OBJECT

### Class and object

A class is a user defined data type which binds data members and member functions into a single unit and objects are the instance of a class. A class is declared by use of the **class** keyword. The general form/ syntax of a **class** definition is shown here:

```
class class_name
{
    type dataMember1;
    type dataMember2;
    type dataMembern;
    type memberFunction1(parameter_list)
    {
        // body of method
    }
    type memberFunction2(parameter_list)
    {
        // body of method
    }
    type memberFunctionnn(parameter_list)
    {
        // body of method
    }
}
```

The data, or variables, defined within a class are called **data members**. The functions or methods declared within a class are known as **member functions**. Collectively, the data members and member functions defined within a class are called **members of the class**.

**Program ❶** Write a program to calculate volume of a box with its width, height and depth initialized.

**Solution:**

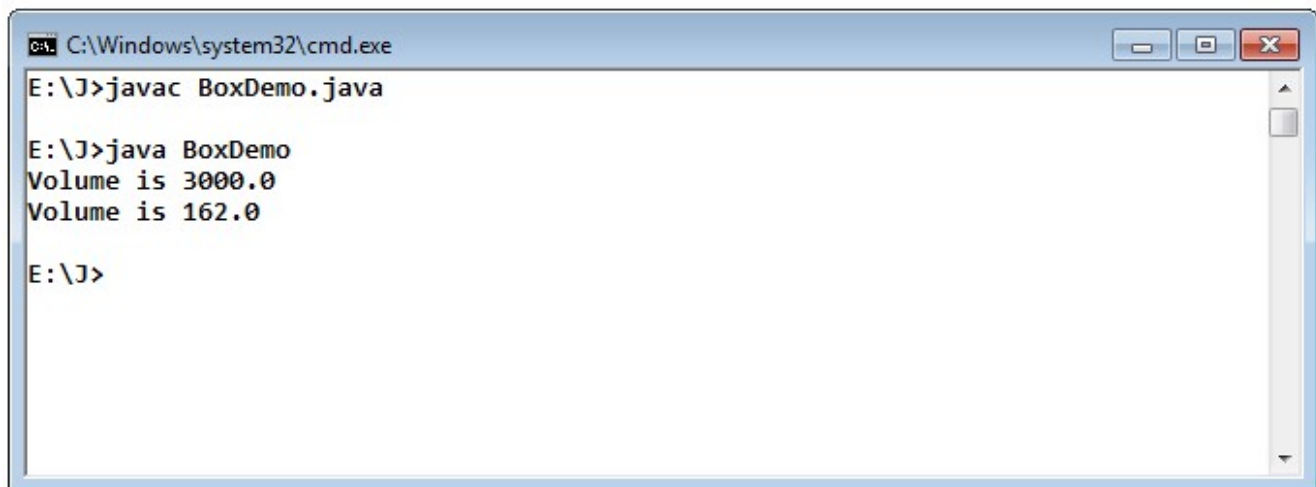
```
class Box
{
    double width;
    double height;
    double depth;
    void volume()
    {
```

```
        System.out.print("Volume is ");
        System.out.println(width * height * depth);
    }
}
class BoxDemo
{
    public static void main(String args[])
    {
        Box mybox1 = new Box(); //creates an object mybox1 and allocates memory
        Box mybox2 = new Box(); //creates an object mybox2 and allocates memory

        mybox1.width = 10;
        mybox1.height = 20;
        mybox1.depth = 15;

        mybox2.width = 3;
        mybox2.height = 6;
        mybox2.depth = 9;

        mybox1.volume();
        mybox2.volume();
    }
}
```

**Output:**

```
C:\Windows\system32\cmd.exe
E:\J>javac BoxDemo.java

E:\J>java BoxDemo
Volume is 3000.0
Volume is 162.0

E:\J>
```

**Explanation**

The **new** operator dynamically allocates memory for an object. It has this general form:

```
class_var = new classname( );
```

Here, `class_var` is a variable of the class type being created. The `classname` is the name of the class that is being instantiated. The class name followed by parentheses specifies the constructor for the class. A constructor defines what occurs when an object of a class is created.

Thus the statement:

**Box** mybox1 = **new Box()**; create an object mybox1 and allocate memory to it.

The above statement can be written in two steps:

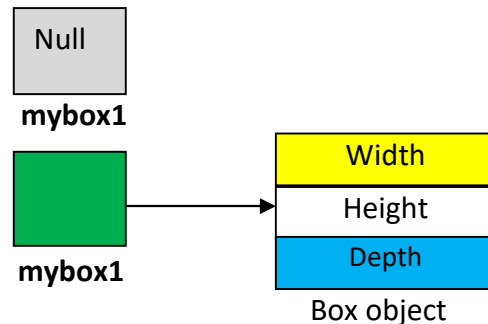
```
Box mybox1;           // declare reference to object
mybox1 = new Box();  // allocate memory to a Box object
```

### Statement

Box mybox1

mybox1=new Box()

### Effect



## Constructor

A constructor is a special member function whose name is same as the class name in which it is declared.

- ◆ A constructor is called automatically when its corresponding object is created.
- ◆ Constructors do not have any return type.
- ◆ Constructors are used to initialize the object of a class.

Constructors are of following types:

**Non-parameterized Constructor:** Constructor which takes no argument(s) is called Default Constructor.

**Parameterized constructor:** Constructor which takes argument(s) is called parameterized Constructor.

**Constructor Overloading-** When a single program contains more than one constructor, then the constructor is said to be overloaded which is also known as **constructor overloading**.

**Program ②** Write a program to illustrate constructor overloading in java.

**Solution:**

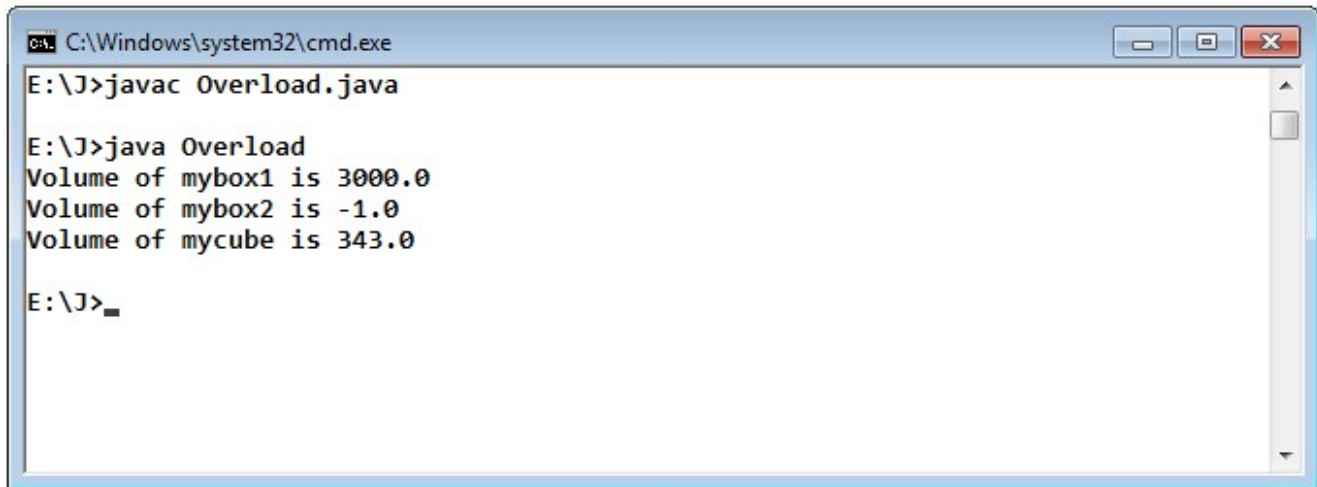
```
class Box
{
    double width;
    double height;
```

```
double depth;
Box(double w, double h, double d)      // Parameterized Constructor
{
    width = w;
    height = h;
    depth = d;
}
Box()                                  // Default Constructor
{
    width = -1;
    height = -1;
    depth = -1;
}

Box(double len)                        // Parameterized Constructor
{
    width = height = depth = len;
}
double volume()
{
    return width * height * depth;
}
}

class Overload
{
    public static void main(String args[])
    {
        Box mybox1 = new Box(10, 20, 15);
        Box mybox2 = new Box();
        Box mycube = new Box(7);
        double vol;
        vol = mybox1.volume();
        System.out.println("Volume of mybox1 is " + vol);
        vol = mybox2.volume();
        System.out.println("Volume of mybox2 is " + vol);
    }
}
```

```
        vol = mycube.volume();  
        System.out.println("Volume of mycube is " + vol);  
    }  
}
```

**Output:**

```
C:\Windows\system32\cmd.exe  
E:\J>javac Overload.java  
  
E:\J>java Overload  
Volume of mybox1 is 3000.0  
Volume of mybox2 is -1.0  
Volume of mycube is 343.0  
  
E:\J>
```

**Garbage Collection**

Java garbage collection is the process by which Java programs perform automatic memory management. Java programs compile to bytecode that can be run on a Java Virtual Machine, or JVM for short. When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program. Eventually, some objects will no longer be needed. The garbage collector finds these unused objects and deletes them to free up memory.

Java garbage collection is an automatic process. The programmer does not need to explicitly mark objects to be deleted. The garbage collection implementation lives in the JVM. Each JVM can implement garbage collection however it pleases; the only requirement is that it meets the JVM specification.

In some languages, such as C++, dynamically allocated objects must be manually released by use of a delete operator. Java takes a different approach; it handles deallocation automatically. The technique that accomplishes this is called garbage collection.

**The finalize () method**

By using this method, we can define specific actions that will occur when an object is just about to be reclaimed by the garbage collector. The finalize ( ) method has this general form or syntax:

```
protected void finalize( )
{
    // finalization code here
}
```

### The static Keyword

The **static keyword** in Java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static entities belong to the class rather than the object of the class.

The static keyword can be applied to variable, method or block as discussed below:

- Static variable (also known as a class variable)
- Static method (also known as a class method)
- Static block

### Static variable

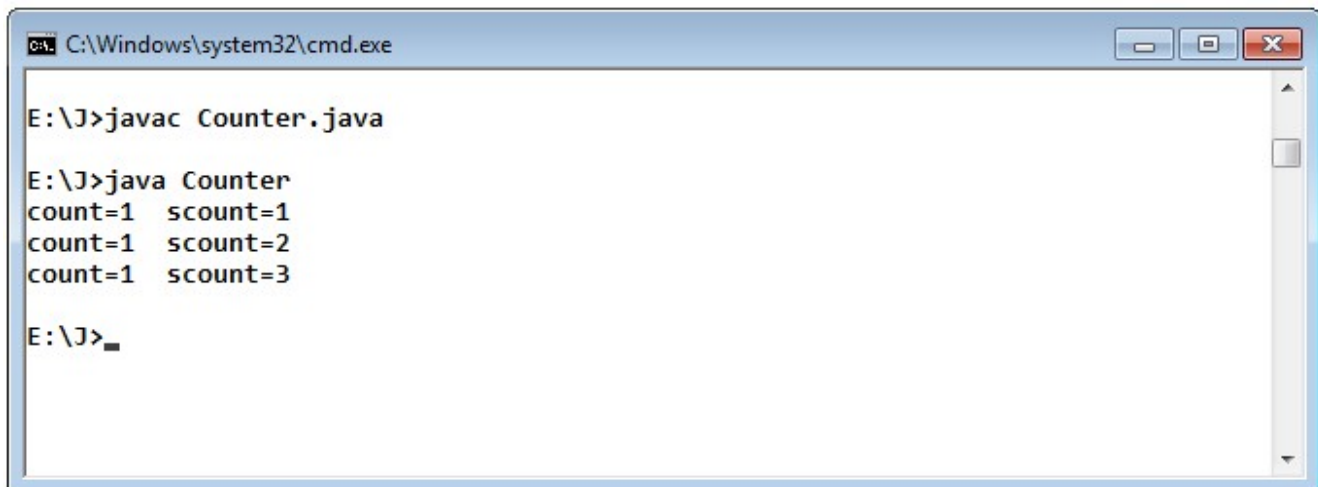
These are variables which are initialized only once during the course of program execution and their declaration are preceded by the keyword static. These variables are used to represent the common property of all objects of a class. Therefore static variables are also referred as class variable. Alternately, only one copy of static variable exists for all the objects of a class. The static variable gets memory only once in the class area at the time of class loading

**Program 3** Write a program which demonstrate the use of instance variable & static variable

**Solution:**

```
class Counter
{
    int count=0;//will get memory each time when an object is created
    static int scount=0;//will get memory once
    Counter()
    {
```

```
count++; //incrementing instance or object variable
scount++; //incrementing static or class variable
System.out.println("count="+count+"\t scount="+scount);
}
public static void main(String args[])
{
    Counter c1=new Counter();
    Counter c2=new Counter();
    Counter c3=new Counter();
}
}
```

**Output:**

```
C:\Windows\system32\cmd.exe

E:\J>javac Counter.java

E:\J>java Counter
count=1  scount=1
count=1  scount=2
count=1  scount=3

E:\J>
```

**Static method**

Any method declaration preceded by the keyword static is known as static method. The properties of static methods are:

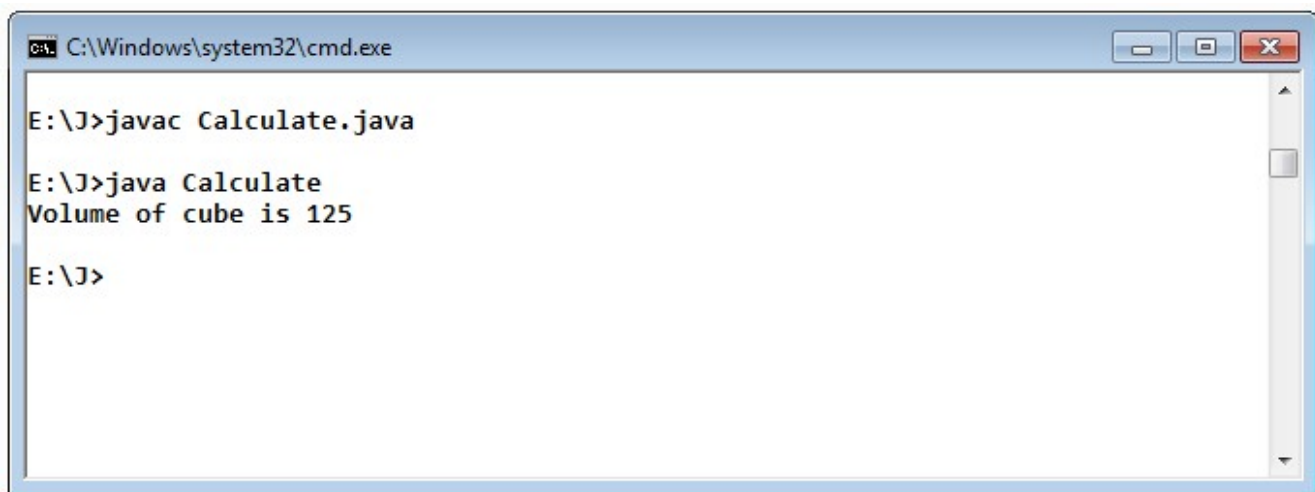
- A static method belongs to the class rather than the object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- A static method can access static data member and can change the value of it.

**Program 4** Write a program which demonstrates the use of static method in java.

**Solution:**

```
class Calculate
{
    static int cube(int x)
```

```
{  
    return x*x*x;  
}  
public static void main(String args[])  
{  
    int r=Calculate.cube(5);  
    System.out.println("Volume of cube is "+r);  
}  
}
```

**Output:**

The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The prompt is at "E:\J>". The user enters "javac Calculate.java", followed by "java Calculate". The output of the program is "Volume of cube is 125". The prompt returns to "E:\J>".

**Restriction of static methods:**

Two main restrictions for the static method are:

- The static method cannot use non static data member or call non-static method directly.
- this and super cannot be used in static context.

Below program demonstrate the illegal use of static methods:

class A

```
{  
    int a=40;                //a is non static  
    public static void main(String args[])  
    {  
        System.out.println(a);    // Compilation error !! a is non static  
    }  
}
```



```
}
```

### Java static block

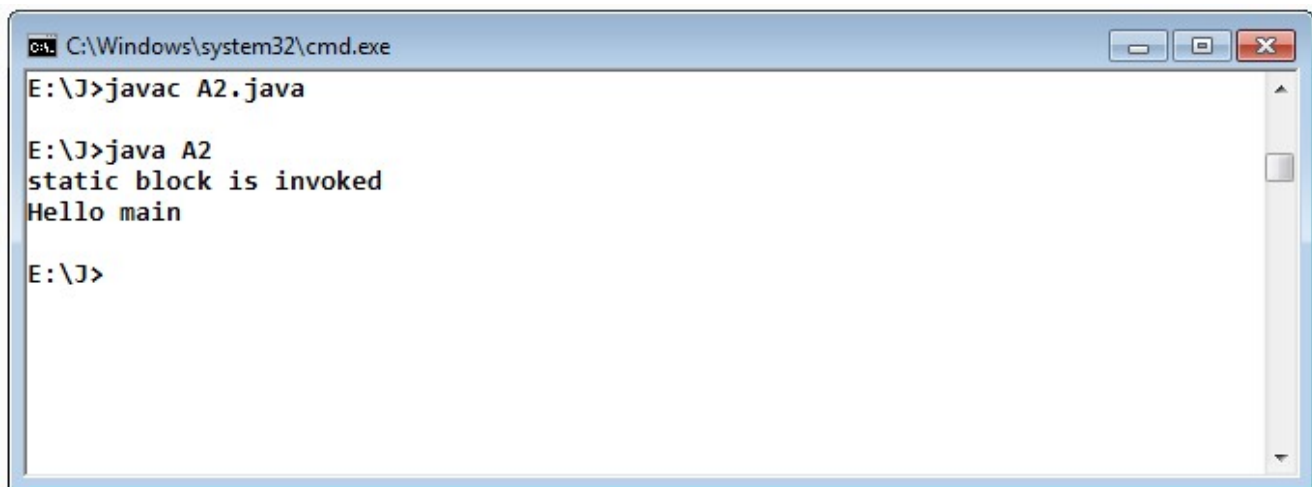
Static block is used to initialize the static data member. It is executed before the main method at the time of class loading.

**Program 5** Write a program which demonstrates the use of static block in java.

**Solution:**

```
class A2
{
    static
    {
        System.out.println("static block is invoked");
    }
    public static void main(String args[])
    {
        System.out.println("Hello main");
    }
}
```

**Output:**



```
C:\Windows\system32\cmd.exe
E:\J>javac A2.java
E:\J>java A2
static block is invoked
Hello main
E:\J>
```

### The this Keyword

In java, this is a reference variable that refers to the current object. Several usages of java this keyword are:

- this can be used to refer current class instance variable.

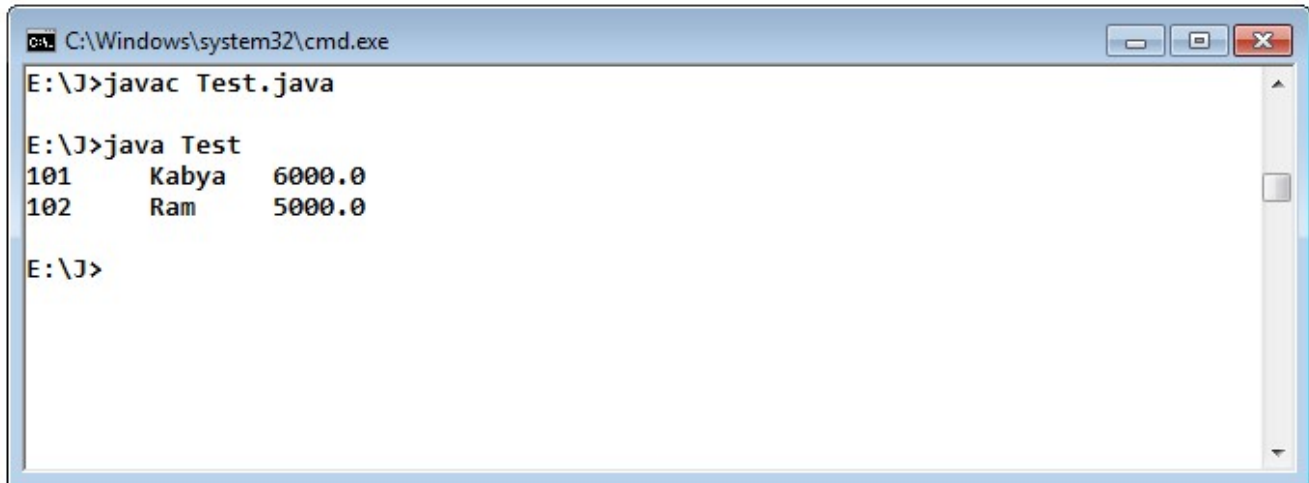
- this can be used to invoke current class method (implicitly)
- this() can be used to invoke current class constructor.
- this can be passed as an argument in the method call.
- this can be passed as argument in the constructor call.
- this can be used to return the current class instance from the method.

**Program 6** Write a program which demonstrates the use of this keyword in java.

**Solution:**

```
class Student
{
    int rollNo;
    String name;
    float fee;
    Student(int rollNo,String name,float fee)
    {
        this.rollNo=rollNo;
        this.name=name;
        this.fee=fee;
        this.display();
    }
    void display()
    {
        System.out.println(rollNo+"\t"+name+"\t"+fee);
    }
}

class Test
{
    public static void main(String args[])
    {
        Student s1=new Student(101,"Kabya",6000f);
        Student s2=new Student(102,"Ram",5000f);
    }
}
```

**Output:**

```
C:\Windows\system32\cmd.exe
E:\J>javac Test.java

E:\J>java Test
101      Kabya      6000.0
102      Ram        5000.0

E:\J>
```

**Access modifiers**

There are two types of modifiers in Java: access modifiers and non-access modifiers. There are many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient, etc.

The **access modifiers** in Java specify the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it. There are four types of Java access modifiers:

- **Private:** The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
- **Default:** The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
- **Protected:** The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
- **Public:** The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Let's understand the access modifiers in Java by a simple table.

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

### Example of private access specifier

In the below example, class A contains private data member and private method. We are trying to access these private members from outside the class, so there is a compile-time error.

class A

```
{
    private int data=40;
    private void msg()
    {
        System.out.println("Hello java");
    }
}
```

public class Simple

```
{
    public static void main(String args[])
    {
        A obj=new A();
        System.out.println(obj.data);    //Compile Time Error
        obj.msg();                       //Compile Time Error
    }
}
```

### Example of default access specifier

If we don't use any modifier, it is treated as default by default. The default modifier is accessible only within package. It cannot be accessed from outside the package. It provides more accessibility than

private. But, it is more restrictive than protected, and public. In the above example, the scope of class A and its method msg() is default so it cannot be accessed from outside the package.

```
package pack;
class A
{
    void msg()
    {
        System.out.println("Hello");
    }
}
package mypack;
import pack.*;
class B
{
    public static void main(String args[])
    {
        A obj = new A();           //Compile Time Error
        obj.msg();                 //Compile Time Error
    }
}
```

### Example of protected access specifier

In the below example, we have created two packages pack and mypack. The class A of package pack is public, so it can be accessed from outside the package. But msg method of this package is declared as protected, so it can be accessed from outside the class only through inheritance.

```
//A.java
package pack;
public class A
{
    protected void msg()
    {
        System.out.println("Hello");
    }
}
```

```
//B.java
package mypack;
import pack.*;
class B extends A
{
    public static void main(String args[])
    {
        B obj = new B();
        obj.msg();
    }
}
```

**Output:**

Hello

**Example of public access specifier**

The public access modifier is accessible everywhere. It has the widest scope among all other modifiers.

```
//A.java
package pack;
public class A
{
    public void msg()
    {
        System.out.println("Hello");
    }
}
```

```
//B.java
package mypack;
import pack.*;
class B
{
    public static void main(String args[])
    {
```

```
A obj = new A();  
obj.msg();  
}  
}
```

**Output:**

Hello

\*\*\*\*\*

### Review Questions of Chapter 4 (Class & Object)

#### MCQ Type

1. Choose the most appropriate answer:

a) Which one of the following is an instance of a class?

- a) field
- b) method
- c) object
- d) None of these

b) Collection of similar objects is termed as

- a) class
- b) array
- c) structure
- d) Both a) and c)

c) Technique used in java for memory release is known as

- a) Destructor
- b) Free

- c) Constructor                      d) Garbage collection
- d) Which one of the following is used for memory allocation in java?**  
a) Constructor                      b) Garbage collection  
c) both a) and b)                      d) Destructor
- e) Fields which are initialized only once and only one copy of it exists for all the object are**  
a) static fields                      b) private fields  
c) public fields                      d) All of these
- f) The property of OOPs by which an entity can acquire multiple forms is known as**  
a) Inheritance                      b) Polymorphism  
c) Encapsulation                      d) Data Aggregation
- g) Which variables are known as class variables?**  
a) public                      b) static  
c) private                      d) final
- h) The act of hiding data and methods from external world with the help of access modifier is known as**  
a) Data Hiding                      b) Polymorphism  
c) Data Encapsulation                      d) Data Abstraction
- i) The object pointer which point to the recent object through which a method is invoked is known as**  
a) this pointer                      b) static pointer  
c) final pointer                      d) None of these
- j) What is the syntax to create an object in java?**  
a) class\_name instance=new class\_name;  
b) class\_name instance=new() class\_name;  
c) class\_name instance=new class\_name[];  
d) class\_name instance=new class\_name();
- k) Which of the following is not true about constructor?**  
a) don't have return type                      b) name is same as class name  
c) can't be overloaded                      d) called automatically

## Short Type

### 2. Answer briefly



- a) Define class.
- b) Define constructor.
- c) What is Garbage collection?
- d) Name the method which is called prior to garbage collection.
- e) Can a static method access non static variable?
- f) What is this keyword in java?
- g) Name the technique by which memory is de-allocated in java?
- h) How do we create objects in java?

### Long Type

#### 3. Answer in details

- a) Define constructor. What are its characteristics? Explain different types of constructor with proper programming examples.
- b) Explain various access modifiers available in java with proper examples.
- c) Explain static methods with the help of proper example.
- d) Explain the use of static block in java with proper example.

\*\*\*\*\*