

# Unit 2: Class and Object

# 2

## Unit Structure

2.1 Learning Objectives

2.2 Arrays

2.3 class, object & method

2.4 Defining class

2.5 Adding variables

2.6 Adding methods

2.7 Creating objects

2.8 Constructor

2.9 this keyword

2.10 Garbage collection

2.11 finalize() method

2.12 Accessing class members

2.13 Methods overloading

2.14 Static members

2.15 Nesting of methods

2.16 Vectors

2.17 Wrapper classes

2.18 Let us sum up

2.19 Check your Progress

2.20 Check your Progress: Possible Answers

2.21 Further Reading

2.22 Assignments

---

## 2.1 LEARNING OBJECTIVE

---

After studying this unit student should be able to:

- Understand the basics of array and its usage
- Understand and use the class and object
- Function of Garbage collector
- Use method overloading, nested function and static members of class
- Explain the usage of wrapper classes.

---

## 2.2 ARRAYS

---

### 2.2.1 ONE DIMENSIONAL ARRAY

An array is a container in which we can store multiple values of a single type. For example, we can create an array that can store 10 values of int type.

The syntax for array declaration is:

```
datatype[] arrayname;
```

here datatype can be any primitive data type and arrayname can be an identifier.

For example

```
double[] s;
```

it means s is an array which stores double type values.

We cannot use array after declaration. We have to allocate memory to the array.

The syntax for memory allocation of array is:

```
double[] s;  
s=new double[10];
```

it means s can hold 10 values which are of double type.

We can also declare and allocate memory of an array simultaneously.

```
double[] s=new double[10];
```

The first element of array can be s[0]. The other elements are s[1], s[2], ....s[9].

We can also initialize the array while declaration. For example,

```
int[] age = {12, 4, 5, 2, 5};
```

For accessing an array we have to use integer index starting from 0 to n-1 (n is length of array). In java, array length can be accessed using arrayname.length, as the length is property of array in java.

For example

```
int[] age = {12, 4, 5, 2, 5};
for(int i=0; i<age.length; i++)
{
    System.out.println(age[i]);
}
```

## 2.2.2 MULTIDIMENSIONAL ARRAY

Multidimensional array is an array of an array. We can create multidimensional array in java. For example.

```
int[][] x = new int[2][3];
```

This array x can store 2 rows of integer values and each row has three integers in it. This is a two dimensional array and it can store 2\*3=6 integers in it.

We can also create three dimensional in java.

```
int[][][] x= new int [2][3][4];
```

it represents 3 dimension and can store 2\*3\*4 integers in it.

Unlike C/C++, multidimensional arrays in java can have different number of integer in each row.

For example

```
int[][] a = {
    {1, 2, 3},
    {4, 5, 6, 9},
    {7}, };
```

In this array a, first row has 3 values, second row has four values and third row has only one value stored in it.

For accessing elements of multidimensional array multiple loops can be used. For example:

```
class MultidimensionalArray {
    public static void main(String[] args) {
        int[][] a = {
            {1, -2, 3},
            {-4, -5, 6, 9},
            {7},
        };
        for (int[] innerArray: a) {
            for(int data: innerArray) {
                System.out.println(data);
            }
        }
    }
}
```

### Array Example 1

A program to sort 5 integers in ascending order.

```
public class Ex_ary1
{
    public static void main(String args[])
    {
        int[] a={23,45,67,8,3};
        System.out.print("Before Sorting :");
        for(int i=0;i<a.length;i++)
            System.out.print(" "+a[i]);

        for(int i=0;i<a.length;i++)
            for(int j=i+1; j<a.length; j++)
```

```

    {
    if(a[i]>a[j])
    {
    int t=a[i];
    a[i]=a[j];
    a[j]=t;
    }
    }

    System.out.print("\nAfter Sorting :");
    for(int i=0;i<a.length;i++)
    System.out.print(" "+a[i]);
}
}

```

```

C:\ajava\oopj>javac Ex_ary1.java
C:\ajava\oopj>java Ex_ary1
Before Sorting : 23 45 67 8 3
After Sorting : 3 8 23 45 67

```

Figure-10 Output of Program

## Array Example 2

A program to add two 3x3 matrix

```

public class Ex_ary2
{
    public static void main(String args[])
    {
        int[][] a={{1,1,1},{2,2,2},{3,3,3}};
        int[][] b={{4,4,4},{5,5,5},{6,6,6}};
        int[][] c=new int[3][3];

        for(int i=0;i<3;i++)
        for(int j=0;j<3;j++)
            c[i][j]=a[i][j]+b[i][j];
    }
}

```

```

        System.out.print("Result matrix:\n");
        for(int i=0;i<3;i++)
        {
            for(int j=0;j<3;j++)
            System.out.print(c[i][j]+" ");
            System.out.print("\n");
        }
    }
}

```

```

C:\ajava\oopj>javac Ex_ary2.java
C:\ajava\oopj>java Ex_ary2
Result matrix:
5 5 5
7 7 7
9 9 9

```

Figure-11 Output of Program

---

## 2.3 CLASS, OBJECT & METHOD

---

An object is an entity which has several attributes and behavior. A number of objects sharing same attributes and behavior form a Class. For example: parrot, peacock, hen, dove are objects of class birds. They have attributes like colour, eating habit, shape of beak etc and behavior like fly, build nest, lay eggs etc. in java we can create a class using class keyword and declare various variables in it for its attributes and create a function for its behavior. In java for creating a class, the class keyword is used. The attributes of the class can be defined as member variable of the class and behaviour of class can be methods of class in java.

---

## 2.4 DEFINING CLASS

---

In java, class can be defined using class keyword follow by class name as shown in example. The definition of class is written within braces. The class name should start with capital letter. If class name has multiple words first letter of each word should be capital. For example: Student, Bird, StringBuffer etc.

```
class Student
{
}
}
```

---

## 2.5 ADDING VARIABLES

---

We can add variables in class by declaring them within class. For each attribute of class we can create variable in it. For example class Student can have attributes like rollNumber, name, course etc. The variable name in class should be in lower case. If variable name has more than one word each word should start with capital letter except first word. For example rollNumber. The student class can be created as follows

```
class Student
{
    int rollNumber;
    String name;
    String course;
}
```

---

## 2.6 ADDING METHODS

---

We can define methods in class. The syntax is return type then name of method followed by arguments in bracket ( ). The function definition is written within braces. For example in Student class we can create two functions getData for assigning values to its variable and printData to print its variables.

```
class Student
{
    int rollNumber;
    String name;
    String course;
    void getData(int r, String n, String c)
    {
        rollNumber=r;
        name=n;
        course=c;
    }
    void printData()
    {
        System.out.println(rollNumber+" "+name+" "+course);
    }
}
```

---

## 2.7 CREATING OBJECTS

---

After defining class, we can use it by creating its object. This is also called instantiation of class. the new keyword is used for creating object of class.

For example,

```
ClassName x = new ClassName ( );
```

In this example ClassName is the name of class created in your program.

Example

```
class Student
{
    int rollNumber;
    String name;
    String course;
    void getData(int r, String n, String c)
    {
        rollNumber = r;
        name = n;
        course = c;
    }
    void printData()
    {
        System.out.println(rollNumber);
        System.out.println(name);
        System.out.println(course);
    }
}
class Exa_Cls
{
    Public static void main(String args[])
    {
        Student s1 = new Student( ); //object s1 is created
        S1.getData(1,"manan","civil");
        s1.printData();
    }
}
```



```

C:\ajava\oopj>javac Exa_Cls.java

C:\ajava\oopj>java Exa_Cls
1
manan
civil

```

Figure-12 Output of Program

---

## 2.8 CONSTRUCTOR

---

In java, we can define Constructors in a class. Constructor is a function which has same name as class name. This function will be called when we create object using new keyword. The constructors are mainly used to initialize the attributes/variables of the class. Constructor can be default constructor or parameterized constructor. In default constructor, nothing is passed as an argument. However in parameterized constructor the parameter values must be passed as arguments of constructor function.

For example:

```

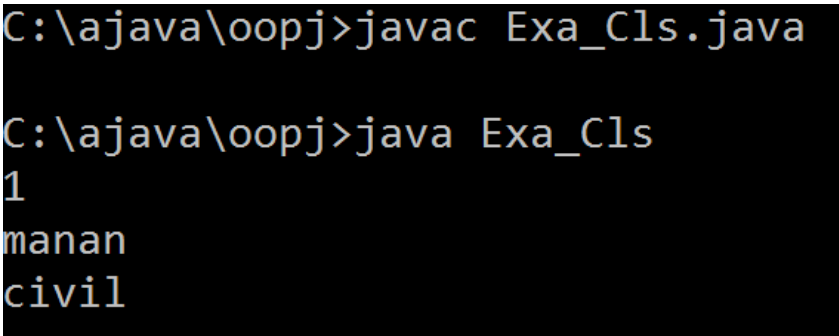
class Student
{
    int rollNumber;
    String name;
    String course;
    Student() //default constructor
    {
        rollNumber=0;
        name="";
        course="";
    }
    Student(int r, String n, String c) //parameterized constructor
    {
        rollNumber=r;
        name=n;
        course=c;
    }
    void printData()
    {
        System.out.println(rollNumber);
        System.out.println(name);
        System.out.println(course);
    }
}

```

```

}
class Exa_Cls
{
    Public static void main(String args[])
    {
        Student s1 = new Student(1,"manan","civil");
        s1.printData();
    }
}

```



```

C:\ajava\oopj>javac Exa_Cls.java

C:\ajava\oopj>java Exa_Cls
1
manan
civil

```

Figure-13 Output of Program

---

## 2.9 THIS KEYWORD

---

**this** is reference variable of java which points to the current object. It can also be used to point instance of the current class as shown in following example.

```

class abc
{
    int a,b,c;
    abc(){ a = 0; b = 0; c = 0;}
    abc( int a,int b, int c)
    {
        this.a = a;
        this.b = b;
        this.c = c;
    }
}
class MyExa
{
    public static void main(String args[])
    {
        abc x = new abc(1,2,3);
    }
}

```

In above example, in class abc, this.a, this.b and this.c are referring the variable of class abc and a,b and c are the parameters of constructor.

---

## **2.10 GARBAGE COLLECTION**

---

In C, when we allocate memory at runtime using malloc() function, at the end of program we have to free them using free() function. Similarly in C++, when we create memory for any object/variable using new, we should free them using delete.

In java when we are creating memory for reference variable/object, programmer don't care about destroying them. There is a special component in JVM called garbage collector which will take care of deletion of all memory occupied by java programs. It frees the heap memory occupied by reference variables which are not in use. Java has an automatic garbage collection.

---

## **2.11 FINALIZE() METHOD**

---

The finalize() is a method of java.lang.Object class which is called by garbage collector for the which is identified to be destroyed. It is because there are no reference to that object in program. In a class we can override (redefine) the finalize method to perform the cleanup of system resources.

---

## **2.12 ACCESSING CLASS MEMBERS**

---

To access the member variables and methods of the class, we should create the object of the class using new keyword. And using the object name and variable/method name separated by . we can access the member variable the example is shown in section 2.7 and 2.6.

---

## **2.13 METHODS OVERLOADING**

---

Method overloading is the feature of object oriented programming. It is used to implement polymorphism. In java in a same class we can define more than one method with same but different signature, this concept is called method overloading.

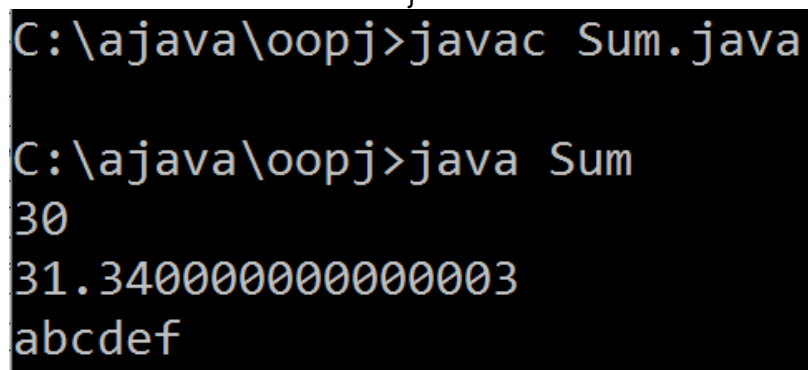
In method overloading same method can be used in different manners. For example in class Add, we can define 3 addition methods shown below,

class Add

```

{
    int addition(int a,int b){ return ( a+b ); }
    float addition(float a,float b) { return ( a+b ); }
    String addition(String a, String b) { return a+b; }
}
public class Sum
{
    public static void main(String args[])
    {
        Sum s1 = new Sum();
        System.out.println(s1.addition(10,20));
        System.out.println(s1.addition(10.56,20.78));
        System.out.println(s1.addition("abc","def"));
    }
}

```



```

C:\ajava\oopj>javac Sum.java

C:\ajava\oopj>java Sum
30
31.3400000000000003
abcdef

```

Figure-14 Output of Program

---

## 2.14 STATIC MEMBERS

---

### 2.14.1 STATIC MEMBER VARIABLES

The variables declared in class can be categorized into two: Class variable and instance variable.

Instance variables are the variable of class which can be access using object/instance of the class. The variable we have used in example of section are the instance variable. For each object the instance variables are separately created in memory.

Class variable are the variables which are shared by all objects of the class. These variables are created in memory once for the class and shared by all objects of the class. The class variables can be accessed using class name then . and the static variable name. No need to create object of class to access the class variable. For creating class variable static keyword is used before its declaration.

Example: In the following example static int n can be used to count number of object created.

```
class Student
{
    static int n = 0;
    int rollNumber = 0;
    String name = "";
    String course = "";
    Student()//default constructor
    {
        rollNumber = 0;
        name = "";
        course = "";
        Student.n++;
    }
    Student(int r, String n, String c)    //parameterized constructor
    {
        rollNumber = r;
        name = n;
        course = c;
        Student.n++;
    }
    void printData()
    {
        System.out.println(rollNumber);
        System.out.println(name);
        System.out.println(course);
    }
}

class Exa_Cls
{
    public static void main(String args[])
    {
        Student s1 = new Student(1,"manan","civil");
        Student s2 = new Student();
        System.out.println("number of objects:"+Student.n);
    }
}
```

```
C:\ajava\oopj>javac Exa_Cls.java

C:\ajava\oopj>java Exa_Cls
number of objects:2
```

Figure-15 Output of Program

## 2.14.2 STATIC MEMBER FUNCTION

We can also declare a static method in a class as a member function. For calling static method we need to use class name instead of object name. Hence we can call the static function without creating object of the class. Also only a static method can be called inside the static function of the class.

### Example1

```
class A
{
    static int sum(int a, int b)
    {
        int c = a + b;
        return c;
    }
}
public class ExStatic1
{
    public static void main(String args[])
    {
        System.out.println(" sum : " + A.sum(10,30) );
    }
}
```

```
C:\ajava\oopj>java ExStatic1
sum : 40
```

Figure-16 Output of Program

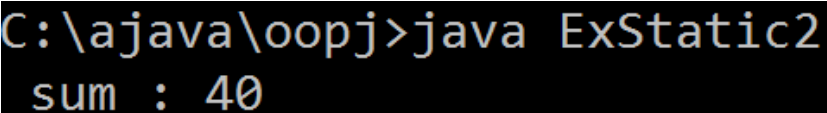
### Example 2

```
class A
{
    static void sum(int a, int b)
    {
        int c = a + b;
        printA(c);    // printA must be static if it is called inside static function sum.
    }
    static void printA(int x)
```

```

        {
            System.out.println(" sum : " + x );
        }
    }
}
public class ExStatic2
{
    public static void main(String args[])
    {
        {
            A.sum(10,30);
        }
    }
}

```



```

C:\ajava\oopj>java ExStatic2
sum : 40

```

Figure-17 Output of Program

---

## 2.15 NESTING OF METHODS

---

When a method of class calling the other method of the same class is called nesting of methods. The following example uses nesting of method.

```

import java.util.Scanner;
class Circle
{
    int radius;
    void getRadius()
    {
        Scanner sc=new Scanner(System.in);
        Radius = sc.nextInt();
    }
    double area()
    {
        getRadius();
        return(3.14*radius*radius);
    }
}
public class Exa
{
    public static void main(String args[])
    {
        {
            Circle c1 = new Circle();
            System.out.println (c1.area());
        }
    }
}

```

```
C:\ajava\oopj>javac Exa.java
C:\ajava\oopj>java Exa
7
153.86
```

Figure-18 Output of Program

---

## 2.16 VECTORS

---

Vector class is available in java.util package. In java array can not be shrink or expand once it is created. Vector is a dynamic array in java which can be shrink or grow as per the requirement. The followings are some of the constructors of Vector class.

- Vector() it creates a vector with capacity 10.
- Vector(int size) it creates a vector with capacity specified by size.

➤ **Methods of Vector class**

1. boolean add(Object obj): it appends obj at the end of the Vector. It returns true if the obj is successfully added.
2. void add(int index, Object obj): it inserts an obj at location specified by index.
3. boolean addAll(Collection c): it is used to add a Vector c in calling Vector. It returns true if the Vector c is successfully added.
4. void addAll(int index, Collection c): it inserts a Vector c at location specified by index.
5. void clear(): it removes all elements in Vector.
6. Object clone(): it creates a clone of this Vector.
7. boolean contains(Object obj): it checks whether the obj exists in Vector or not.
8. void ensureCapacity(int minCapacity): it increases the capacity of vector ensuring that minCapacity elements can be stored in Vector.
9. Object get(int index): it returns object stored at index position in Vector
10. int indexOf(Object obj): it search the first occurrence of the obj and returns its position in Vector.



11. boolean isEmpty(): checks if the Vector has elements in it.
12. Int lastIndexOf(Object obj): it search the last occurrence of the obj and returns its position in Vector.
13. boolean remove(Object obj): it removes the first occurrence of obj in Vector.
14. boolean equals(Object obj): it compares Vector with other Vector
15. Object firstElement(): it returns first element in the Vector.
16. Object lastElement(): it returns last element in the Vector.
17. Void trimToSize(): it trim the capacity of Vector to its size.
18. String toString(): it returns String form of Vector.
19. Object[] toArray(): it converts a Vector into array of Objects.
20. int size(): it returns number of elements stored in Vector.
21. int capacity(): it returns the capacity of vector.
22. void setSize ( int nSize): it set the size of the Vector.
23. void setElementAt(Object obj, int index): it replace an object at index position with obj

**Example:**

```
import java.util.Vector;
public class Exa1
{
    public static void main(String args[])
    {
        Vector v1=new Vector(20);
        v1.add("A");
        v1.add("C");
        v1.add(1,"B");
        System.out.println("size: " + v1.size());
        System.out.println("capacity " + v1.capacity());
        Vector v2=(Vector)v1.clone();
        System.out.println(" Vector v1 " + v1);
        System.out.println(" Vector v2 " + v2);
        v1.addAll(v2);
        System.out.println(" Vector v1.addAll(v2): " + v1);
        System.out.println(" Is B in v1:"+ v1.contains("B"));
        System.out.println(" element at 2 : " + v1.get(2));
        System.out.println(" POsition of A : " + v1.indexOf("A"));
        System.out.println(" Check for empty v1: " + v1.isEmpty());
        System.out.println(" Last index of A: " + v1.lastIndexOf("A"));
        v1.remove("C");
        System.out.println(" After removing C in v1 : " + v1);
        System.out.println("Compare v1 and v2 : " + v1.equals(v2));
    }
}
```

```

System.out.println( " Fisrt element of v1 : " + v1.firstElement());
System.out.println(" Last element of v1 : " + v1.lastElement());
System.out.println(" v1 to String : " + v1.toString());
}
}

```

```

size: 3
capacity 20
Vector v1 [A, B, C]
Vector v2 [A, B, C]
Vector v1.addAll(v2): [A, B, C, A, B, C]
Is B in v1:true
element at 2 : C
Position of A : 0
Check for empty v1: false
Last index of A: 3
After removing C in v1 : [A, B, A, B, C]
Compare v1 and v2 : false
Fisrt element of v1 : A
Last element of v1 : C
v1 to String : [A, B, A, B, C]

```

Figure-19 Output of Program

---

## 2.17 WRAPPER CLASSES

---

Wrapper classes are the classes whose objects wrap the primitive data types. To treat primitive data type as a Class and Object, java provide a wrapper class for each primitive data types. The following is the list of wrapper classes and their corresponding primitive data types.

Primitive Data type	Wrapper Class
boolean	Boolean
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character

Table-7 list of wrapper classes and their corresponding primitive data types

Advantages of wrapper class:

- 1). They convert a primitive data type into object when we need to pass them as reference argument to the function. By default the primitive data types are passed as value into the function.
- 2). The Vector can store objects only. If we want to store primitive data values in Vector, we need to convert them into objects.

Autoboxing is an important concept related to wrapper classes. Autoboxing is an automatic conversion of primitive data types into object of its wrapper class. The reverse process of autoboxing is called unboxing. Unboxing is automatic conversion of object of wrapper class into its corresponding primitive data type.

For example

- 1) 

```
int a = 5;
Integer aa = a;    //autoboxing
```
- 2) 

```
Vector v1 = new Vector();
v1.add(24); //autoboxing 24 into Integer object
v1.add(89);
int n=v1.firstElement(); //unboxing
```

### Example

```
class Exa3
{
    public static void main(String args[])
    {
        //Autoboxing
        byte a = 10;
        Byte aobj = new Byte(a);

        int b = 289;
        Integer bobj = new Integer(b);

        float c = 508.5f;
        Float cobj = new Float(c);

        double d = 90.3;
        Double dobj = new Double(d);

        char e='x';
        Character eobj=e;

        System.out.println("Autoboxing");
        System.out.println(aobj);
        System.out.println(bobj);
    }
}
```

```

System.out.println(cobj);
System.out.println(dobj);
System.out.println(eobj);

//Unboxing
byte v = aobj;
int w = bobj;
float x = cobj;
double y = dobj;
char z = eobj;

System.out.println("Unboxing");
System.out.println(v);
System.out.println(w);
System.out.println(x);
System.out.println(y);
System.out.println(z);
}
}

```

```

C:\ajava\oopj>javac Exa3.java
C:\ajava\oopj>java Exa3
Autoboxing
10
289
508.5
90.3
x
Unboxing
10
289
508.5
90.3
x

```

Figure-20 Output of Program

➤ **Methods of wrapper classes**

The following are some of the methods of wrapper class.

- **valueOf(String s)**

All wrapper class except Character class have this function. It is a static function hence called using class name. This function converts a String representation of any primitive value into its corresponding wrapper class object.

Example:

```
Integer a=Integer.valueOf("100");
Byte b=Byte.valueOf("8");
Double c=Double.valueOf("10.80");
```

- **valueOf(String s, int radix):**

This is a static function of Byte, Short, Integer and Long wrapper class. This function converts a string into corresponding wrapper class object. However the String stores the value represented in radix form. Radix 2 is for binary, 8 is for octal, 16 is for hexadecimal and so on.

For example

```
Integer a=Integer.valueOf("101",2);//store 7 in a because 101 is binary of 7.
```

- **valueOf(primitive\_data\_type x):**

All wrapper classes have this static function which converts a primitive data value into its corresponding wrapper class object.

For example

```
Integer a = Integer.valueOf(100);
Double b = Double.valueOf(34.6);
```

### Example

```
public class ExWrap1
{
    public static void main(String args[])
    {
        // example of valueOf
        System.out.println(" valueOf converts String into Wrapper class object");
        Integer a=Integer.valueOf("100");
        Byte b=Byte.valueOf("8");
        Double c=Double.valueOf("10.80");
        System.out.println("Integer: " + a);
        System.out.println("Byte: " + b);
        System.out.println("Double: " + c);

        System.out.println(" valueOf converts String with differnt base into Wrapper
                            class object");
        Integer a1=Integer.valueOf("1110",2);
        System.out.println("Integer: " + a1);

        System.out.println(" valueOf converts primitive data type into Wrapper class
                            object");
```

```

Integer a2 = Integer.valueOf(100);
Double b2 = Double.valueOf(34.6);
System.out.println("Integer: " + a2);
System.out.println("Integer: " + b2);

    }
}

```

```

C:\ajava\oopj>java ExWrap1
valueOf converts String into Wrapper class object
Integer: 100
Byte: 8
Double: 10.8
valueOf converts String with differnt base into Wrapper class object
Integer: 14
valueOf converts primitive data type into Wrapper class object
Integer: 100
Integer: 34.6

```

Figure-21 Output of Program

### ➤ Primitive data type conversion functions

public byte byteValue(), public short shortValue(), public int intValue(), public long longValue(), public float floatValue(), public float doubleValue() are the non static functions. They need object of Wrapper class to call. The numeric wrapper classes like Byte, Short, Integer, Long, Float, and Double has these all methods defined in them. These methods are used to return corresponding primitive data type value.

For example,

```

Integer x = new Integer(189);
int y = x.intValue();
byte z = x.byteValue();
float a = x.floatValue();

```

### Example:

```

public class ExWrap2
{
    public static void main(String args[])
    {
        System.out.println(" xxxValue functions converts one numeric datatype into
                           other ");
        Integer x = new Integer(122);
    }
}

```

```

    int y = x.intValue();
    byte z = x.byteValue();
    float a = x.floatValue();
    System.out.println(" int :" + y);
    System.out.println(" byte :" + z);
    System.out.println(" float :" + a);
}
}

```

```

C:\ajava\oopj>java ExWrap2
xxxValue functions converts one numeric datatype into other
int :122
byte :122
float :122.0

```

Figure-22 Output of Program

➤ **String to primitive data type conversion functions**

public static int parseInt(String s), public static byte parseByte(String s), public static short parseShort(String s), public static long parseLong(String s), public static float parseFloat(String s), public static double parseDouble(String s), public static boolean parseBoolean(String s)

All the wrapper class except Character class has parse function. This function is used to convert a String argument into corresponding primitive data type value.

For example:

```

int x = Integer.parseInt("123");
double y = Double.parseDouble("123.56");
boolean z = Boolean.parseBoolean("false");

```

The parse function has one more version which is,

public static int parseInt(String s, int radix) for Integer class.

Similarly the wrapper classes Byte, Short and Long have this function. It converts a String s, which represents a number with base radix into primitive data types.

For example,

```

int x=Integer.parseInt("1111",2); //this converts a binary 1111 into integer.

```

This function can be used to convert string representation of binary (radix 2), octal(radix 8) or hexadecimal (radix 16) number into decimal value.

**Example:**

```
public class ExWrap3
{
    public static void main(String args[])
    {
        System.out.println(" parseXXX functions converts String to primitive data type");

        int x = Integer.parseInt("123");
        double y = Double.parseDouble("123.56");
        boolean z = Boolean.parseBoolean("false");

        System.out.println(" int :" + x);
        System.out.println(" double :" + y);
        System.out.println(" boolean :" + z);

        System.out.println(" parseXXX functions converts a String representation of a
                            number with base radix into primitive data types.");
        int x1=Integer.parseInt("1111",2);
        System.out.println(" decimal of 1111 is int :" + x1);

    }
}
```

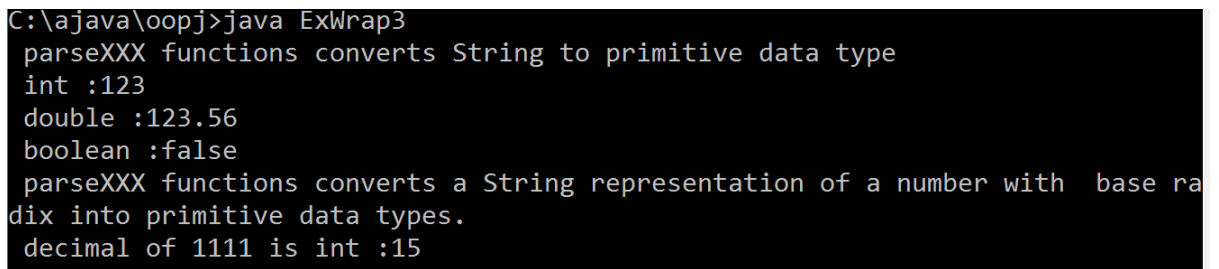


Figure-23 Output of Program

➤ **public String toString()**

every wrapper class has this function. It is used to convert a wrapper class object into String.

For example,

```
Double d=new Double(123.88);
String s=d.toString(); //stores "123.88" into s
```

➤ **public static String toString(primitive p)**



every wrapper class has this function. It is used to convert a primitive data type value into String.

For example,

```
String s=Double.toString(123.89);
```

**Example:**

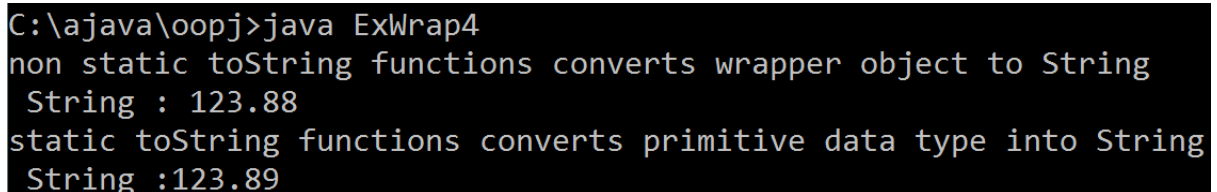
```
public class ExWrap4
{
public static void main(String args[])
{
System.out.println( "non static toString functions converts wrapper object to String ");

Double d = new Double(123.88);
String s = d.toString();

System.out.println(" String : " + s);

System.out.println(" static toString functions converts primitive data type into String
");
String x1 = Double.toString(123.89);
System.out.println(" String : " + x1);

}
}
```



```
C:\ajava\oopj>java ExWrap4
non static toString functions converts wrapper object to String
String : 123.88
static toString functions converts primitive data type into String
String :123.89
```

**Figure-24 Output of Program**

---

## 2.18 LET US SUM UP

---

**Array:** one dimensional and multi dimensional arrays

**class:** a non primitive data type which encapsulates variables and function in it.

**object:** an instance of class or variable of type class. The new keyword is used to create object.

**member variable:** list of variables defined in class

**member function:** methods/functions defined within class

**constructor:** It is a function of a class having same name as class name. It is called to initialize object when it is created.

**Garbage collection:** it automatically frees the unnecessary memory area of the program.

**finalize():** this method will be called by garbage collector before destroying the object.

**method overloading:**In a class we can write more than one method with same name and different signature.

**static variables:**They are also the class variable. All objects of a class share the static variables defined in the class. They can be accessed using class name.

**static methods:**They are the method of class which calls static method inside it. They can also be called using class name.

**vector:**It is a dynamic array which can be grow and shrink run time as per requirement. It is in java.util package.

**wrapper classes:**For each primitive data type there is a class in java which is called wrapper class. The wrapper class wraps the primitive data value as an object and can have various data conversion functions.

---

## 2.19 CHECK YOUR PROGRESS

---

➤ True-False with reason:

1. Class and object are same.
2. Static member function can be called without object.
3. We can enhance capacity of Vector at run time.
4. Constructor function can have any name.
5. We can write only one constructor function for a class.
6. We can not call static function inside non static function.
7. Instance variables are shared by all objects of the class
8. A[1] refer to the first element of the array
9. Array can be initialized.
10. We can implement matrix using single dimensional array.

➤ Answer the followings:

1. List all wrapper class.
2. How can we create an object of wrapper class?

3. How can we create an array of 10 integers?
4. How can we create an object of a class?
5. Give example of method overloading.
6. How can we convert a string "102" into a number?
7. How can we find size of a vector object?
8. Compare class variable and instance variable
9. Compare Vector and array.
10. Compare class and object.

➤ Identify the class and its attributes and methods from following problem statement.

1. In school software, they are storing information of each students and staff.
2. In library software, they are allowing issue and return of the book by library members.
3. We want to design software for restaurant bill generation.

➤ Multiple choice questions:

1) What is output of the following code,

```
class Test {
    int i;
}
class Main {
    public static void main(String args[]) {
        Test t = new Test();
        System.out.println(t.i);
    }
}
```

- |                       |                    |
|-----------------------|--------------------|
| (a) garbage value     | (b) 0              |
| (c) compilation error | (d) run time error |

2) What is output of the following code,

```
class Test {
    int i;
}
```

```
class Main {
    public static void main(String args[]) {
        Test t;
        System.out.println(t.i);
    }
}
```

- (a) garbage value
- (b) 0
- (c) compilation error
- (d) run time error

3) The default value of a static integer variable of a class in Java is?

- (a) 0
- (b) 1
- (c) Garbage value
- (d) Null (e) -1

4) What will be printed as the output of the following program?

```
public class testincr
{
    public static void main(String args[])
    {
        int i = 0;
        i = i++ + i;
        System.out.println("I = " +i);
    }
}
```

- (a) I = 0
- (b) I = 1
- (c) I = 2
- (d) I = 3
- (e) Compile-time Error.

5) What is the stored in the object obj in following lines of code?

**box obj;**

- a) Memory address of allocated memory of object
- b) NULL
- c) Any arbitrary pointer
- d) Garbage

- 6) Which of these keywords is used to make a class?
- a) class
  - b) struct
  - c) int
  - d) none of the mentioned

- 7) Which of these operators is used to allocate memory for an object?
- a) malloc
  - b) alloc
  - c) new
  - d) give

- 8) What is the output of this program?

```
class box
{
    int width;
    int height;
    int length;
}
class mainclass
{
    public static void main(String args[])
    {
        box obj = new box();
        System.out.println(obj);
    }
}
```

- a) 0
  - b) 1
  - c) Runtime error
  - d) classname@hashCode in hexadecimal form
- 9) Which keyword is used by the method to refer to the object that invoked it?
- a) import
  - b) catch
  - c) abstract
  - d) this
- 10) Which of the following is a method having same name as that of its class?
- a) finalize
  - b) delete
  - c) class
  - d) constructor

11) Which operator is used by Java run time implementations to free the memory of an object when it is no longer needed?

- a) delete
- b) free
- c) new
- d) none of the mentioned

12) Which function is used to perform some action when the object is to be destroyed?

- a) finalize()
- b) delete()
- c) main()
- d) none of the mentioned

13) What is the output of this program?

```
class box
{
    int width;
    int height;
    int length;
    int volume;
    box()
    {
        width = 5;
        height = 5;
        length = 6;
    }
    void volume()
    {
        volume = width*height*length;
    }
}
class constructor_output
{
    public static void main(String args[])
    {
        box obj = new box();
        obj.volume();
        System.out.println(obj.volume);
    }
}
```

```

        }
    }
a) 100
c) 200
b) 150
d) 250

```

- 14) Which of the following statements are incorrect?
- a) default constructor is called at the time of object declaration
  - b) Constructor can be parameterized
  - c) finalize() method is called when a object goes out of scope and is no longer needed
  - d) finalize() method must be declared protected

---

## 2.20 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

---

➤ True-False with reason:

1. False. Objects are the instance of class.
2. True.
3. True.
4. False. Constructor function must have name as class name.
5. False. We can write multiple constructor for a class with different argument in each.
6. False. We can call static function inside non static function.
7. False. Class variables/static variables are shared by all objects of the class
8. False. A[0] refer to the first element of the array
9. True.
10. False. We can implement matrix using two dimensional array

➤ Answer the followings:

1. Wrapper Classes:  
Boolean, Byte, Short, Integer, Long, Float, Double, Character
2. To create an object of wrapper class:  
Boolean a=true;  
Boolean x=a;

3. To create an array of 10 integers :

```
int[] a=new int[10];
```

4. To create an object of a class:

```
Class_Name obj= new Class_Name();
```

5. Example of method overloading:

```
class Ex_Add
{
    static int add(int a,int b){return a+b;}
    static int add(int a,int b,int c){return a+b+c;}
}
class ExOverloading
{
    public static void main(String[] args)
    {
        System.out.println(Adder.add(11,11));
        System.out.println(Adder.add(11,11,11));
    }
}
```

6. To convert a string "102" into a number:

```
int a= Integer.parseInt("102");
```

7. The size() method of Vector class in Java is used to get the size of the Vector.

8. Class variable v/s Instance variable

<b>Class variable</b>	<b>Instance variable</b>
They are static member variables of class	They are non static member variables of class
They are shared among all object of class	They are separately created for each object
To access class variable class name is used.	To access instance variable object name is used.

9. Vector v/s array.



<b>Vector</b>	<b>Array</b>
Vector is resizable array	The length of an Array is fixed.
Vector is synchronized	Array is not synchronized.
Vector can store any type of objects	Array can store same type of objects
Vector is slow to access.	Array supports efficient random access to the members

10. Class v/s object.

<b>Class</b>	<b>Object</b>
It is a blueprint/structure of object.	It is an instance of class
Class is a group of similar entities	Object is a real world entity
Class is declared once	Object is created many times as per requirement.
Class doesn't allocated memory when it is created.	Object allocates memory when it is created.

➤ Identify the class and its attributes and methods from following problem statement.

1. In school software, they are storing information of each students and staff.

Class name : Student

Attributes : enrollment number, name, course, address, phone number, semester

Methods: enroll\_course(int enr\_no, String crs), print\_data(), get\_data()

Class name : Staff

Attributes : Employ ID, name, designation, address, phone number, qualification

Methods: enroll\_course(int enr\_no, String crs), print\_data(), get\_data()

2. In library software, they are allowing issue and return of the book by library members.

a. Class name: Member

b. Attributes : Library ID, name, address, phone number

c. Methods: add\_member(), searchMember(), printAllMembers(), deleteMember()

d. Class name: Book

e. Attributes : bookID, title, author, publisher, price, qty

f. Methods: addBook(), searchBook(), printAllBooks(), deleteBook()

g. Class name: Book\_transaction

h. Attributes: bookID, Library ID, date\_issue, date\_return, fine.

i. Methods : bookIssue(), bookReturn()

3. We want to design software for restaurant bill generation.

a. Customer : custId, custName, custAddr, custPhone

b. Methods : addCust(), searchCust(), deleteCust()

c. Item: itemID, itemName, itemCategory, itemPrice

d. Methods : addItem(), searchItem(), deleteItem()

e. Bill : billID, custID, itemID, qty, billDate, billAmount

f. Methods : billGeneration(), billPayment(), printBill()

➤ Multiple choice questions.

1) b

5) b

9) d

2) c

6) a

10) d

3) a

7) c

11) d

4) b

8) d

12) b

---

## 2.21 FURTHER READING

---

1. “Java 2: The Complete Reference” by Herbert Schildt, McGraw Hill Publications.
2. “Effective Java” by Joshua Bloch, Pearson Education.

---

## 2.22 ASSIGNMENTS

---

- Write java program for following:
- 1) Create a class name meter which represents a distance in meter and centimeter. Also create class name kilometer which represents distance in km and meter. In both class write a function which converts one class to other.
  - 2) Create a class name Doctor with properties and methods. The properties can be name, phone number, qualification, specialization etc. The methods include getting information of doctor and printing them.
  - 3) Sort numbers in descending order.
  - 4) Create a menu driven program for matrix operations like add, subtract, and multiply.
  - 5) Find maximum and minimum from the n numbers.
  - 6) Create a class student with necessary properties, methods and constructor. Overload a function name search in this class which allows us to search student based on roll number, name and city.
  - 7) To print transpose of matrix.
  - 8) To implement pop and push operation of stack using array.

# Unit 3: Inheritance and Interface

# 3

## Unit Structure