Third Year Diploma Courses in Computer Science & Engineering, Computer Engineering, Computer Technology & Information Technology Branch.

Java Programming

As per MSBTE 'I' Scheme Syllabus
JPR-22412

Unit- II
Derived Syntactical constructs in java

Total Marks- 18

Prof. Gunwant V. Mankar
B.E(IT), M.Tech(CSE), AMIE, MIAEng, MSCI
Director
(ConnectSoft Infotech, Warora, IN)
e-mail:- info@gunwantmankar.com
website- www.gunwantmankar.com

Unit-II Derived Syntactical constructs in java

2.1. Constructor and methods

2.1.1. Methods

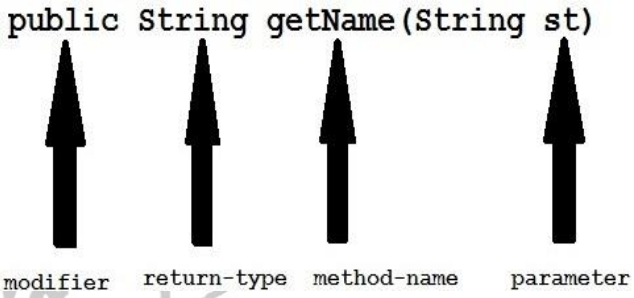
- Method describe behaviour of an object.
- A method is a collection of statements that are group together to perform an operation.

Syntax of method is

```
return-type methodName(parameter-list)
   {
      //body of method
   }
```

Example of a Method

```
public String getName(String st)
   {
     String name="Java Programming";
     name=name+st;
     return name;
   }
```



Modifier – Modifier are access type of method.

Return Type – A method may return value. Data type of value return by a method is declare in method heading

Method name – Actual name of the method

Parameter – Value passed to a method

Method body – collection of statement that defines what method does

2.1.2. Constructor

- Constructor in JAVA is a special type of method that is used to initialize the object.
- JAVA constructor is invoked at the time of object creation.
- It constructs the values i.e. provides data for the object that is why it is known as constructor.
- A constructor has same name as the class in which it resides.
- Constructor in JAVA can not be abstract, static, final or synchronized.
- These modifiers are not allowed for constructor.

```
class Car
    {
        String name ;
        String model;
        Car( )    //Constructor
            {
                name ="";
                model="";
            }
    }
```

Rules for creating JAVA constructor

There are basically two rules defined for the constructor

1. Constructor name must be same as its class name
2. Constructor must have no explicit return type

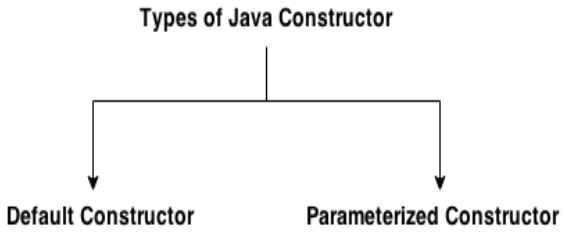
Difference between Constructor and Method

Sr.No	Constructor	Method
1	The name of the constructor must be same as the class name.	The name of the method should not be the class name.
2	It does not return anything hence no return type.	It can return and hence it has a return type. If method does not return anything then the return type is void.
3	The purpose of constructor is to initialize the data members of the class.	The method is defined to execute the core logic of the class.
4	The constructor is invoked implicitly at the time of object creation.	The method must be called explicitly using the object name and dot operator.

2.1.3. Types of JAVA constructors

There are two types of constructors

1. Default constructor (no-arg constructor)
2. Parameterized constructor



1. Default Constructor

- A constructor that have no parameter is known as default constructor.

Syntax of default constructor

```
<class_name> ( )
    {
    }
```

Example of default constructor.

```
class Bike1
  {
    Bike1( )
      {
        System.out.println("Bike is created");
      }
    public static void main(String args[ ])
      {
        Bike1 b=new Bike1();
      }
  }
```

This will produce the following result

```
Bike is created
```

Note – If there is no constructor in a class, compiler automatically creates a default constructor.

Example of default constructor that displays the default values

```
class Student
  {
    int id;
    String name;
    void display( )
      {
        System.out.println(id+" "+name);
      }
    public static void main(String args[])
```

```
    {
        Student  s1=new Student();
        Student  s2=new Student();
        s1.display();
        s2.display();
    }
}
```

This will produce the following result

```
0 null
0 null
```

In the above class, any constructor is not created any constructor. Compiler provides the default constructor. Here 0 and null values are provided by default constructor.

2. Parameterized Constructor

- A constructor that have parameters is known as parameterized constructor.
- Parameterized constructor is used to provide different values to the distinct objects.

Example of parameterized constructor.

```
class Student
  {
    int id;
    String name;
    Student(int i, String n)
      {
        id = i;
        name = n;
      }
    void display( )
      {
        System.out.println(id+" "+name);
      }
    public static void main(String args[ ])
      {
```

```
        Student  s1 = new Student4(111,"Ram");
        Student  s2 = new Student4(222,"Shyam");
        s1.display();
        s2.display();
    }
}
```

This will produce the following result

```
111 Ram
222 Shyam
```

2.1.4. Arguments Passing

- There are two ways to pass an argument to a method.
  1. call-by-value
  2. call-by-reference

NOTE – In JAVA, when a primitive type is passed to a method, it is passed by value whereas when an object of any type is passed to a method, it is passed as reference.

1. call-by-value

- In this approach copy of an argument value is pass to a method. Changes made to the argument value inside the method will have no effect on the arguments.

Example of call-by-value

```
public class Test
{
    public void callByValue(int x)
    {
        x=100;
    }
    public static void main(String[] args)
    {
        int x=50;
        Test t = new Test();
        t.callByValue(x); //function call
        System.out.println(x);
    }
```

```
}
```

This will produce the following result

```
50
```

2. call-by-reference

- In this reference of an argument is pass to a method. Any changes made inside the method will affect the argument value

Example of call-by-reference

```
public class Test
{
    int x=10;
    int y=20;
    public void callByReference(Test t)
        {
            t.x=100;
            t.y=50;
        }
    public static void main(String[] args)
        {
            Test ts = new Test();
            System.out.println("Before "+ts.x+" "+ts.y);
            ts.callByReference(ts);
            System.out.println("After "+ts.x+" "+ts.y);
        }
}
```

This will produce the following result

```
Before 10 20
After 100 50
```

2.1.5. this keyword

- this keyword is used to refer to current object.
- this is always a reference to the object on which method was invoked.
- this can be used to invoke current class constructor.
- this can be passed as an argument to another method.

Example

```
class Box
   {
      Double width, height, depth;
      Box (double width, double height, double depth)
         {
            this.width = width;
            this.height = height;
            this.depth = depth;
         }
   }
```

Here this keyword is used to initialize member of current object.

Program for this Keyword

```
class thisDemo
{
        int a, b;
        public void sum(int a, int b)
        {
                this.a=a;
                this.b=b;
        }
        public void show()
        {
                int c=a+b;
                System.out.println("a="+a);
                System.out.println("b="+b);
                System.out.println("sum="+c);
                System.out.println("-----------------------------------------");
        }
public static void main(String args[])
        {
                thisDemo d1= new thisDemo();
```

```
                d1.sum(10,20);
                d1.show();
                thisDemo d2= new thisDemo();
                d2.sum(20,30);
                d2.show();
        }
}
```

Output:-

a=10

b=20

sum=30

---

a=20

b=30

sum=50

2.1.6. Command line argument in JAVA

- The command line argument is the argument passed to a program at the time when it is run.
- To access the command-line argument inside a JAVA program is quite easy, they are stored as string in String array passed to the args parameter of main() method.

Example

```
class cmddemo
   {
      public static void main(String[] args)
         {
            For (int i=0;i< args.length;i++)
               {
                  System.out.println(args[i]);
               }
         }
   }
```

Execute this program a

C:\>javac cmddemo.java

C:\>java cmddemo 10 Gunwant 30

This will produce the following result

10

Gunwant

30

Programs on Command Line arguments

1) Write a program to accept two number as command line arguments and print addition of those number?

```
class demo
{
    public static void main(String args[])
    {
    int n1=Integer.parseInt(args[0]);
    int n1=Integer.parseInt(args[1]);
    int add=n1+n2;
System.out.println(add);
}
}
```

C:\>javac demo.java

C:\>java demo 10 20

30

2) Write a program to accept number from command line and print the number is odd or even?

```
class clsdemo
    {
        public static void main(String[] args)
        {
            int n=Integer.parseInt(args[0]);
            if(n%2==0)
            {
```

```
                    System.out.println(n+" Is a even no");
                }
                else
                System.out.println(n+" Is Odd no");
            }
        }
```

C:\>javac clsdemo.java

C:\>java clsdemo 10

10 Is even no

3) Write a program to accept number from command line and print square root of the number?

```
class clsdemo3
{
    public static void main(String[] args)
    {
        int n= Integer.parseInt(args[0]);
        double ans;
        ans=Math.sqrt(n);
        System.out.println("Square root of "+n+"= " +ans);
    }
}
```

>javac clsdemo3.java

>java clsdemo3 25

Square root of 25= 5.0

2.1.7. Garbage collection

- In JAVA destruction of object from memory is done automatically by the JVM.
- When there is no reference to an object, then that object is assumed to be no longer needed and the memory occupied by the object are released.
- This technique is called Garbage Collection.
- This is accomplished by the JVM.
- Unlike C++ there is no explicit need to destroy object.

2.1.8. finalize( ) method

- Sometime an object will need to perform some specific task before it is destroyed such as closing an open connection or releasing any resources held.
- To handle such situation finalize() method is used.
- finalize()method is called by garbage collection thread before collecting object.
- Its the last chance for any object to perform cleanup utility.

Signature of finalize() method

```
protected void finalize()
  {
    //finalize-code
  }
```

Some Important Points to Remember

1. finalize() method is defined in JAVA.lang.Object class, therefore it is available to all the classes
2. finalize() method is declare as protected inside Object class
3. finalize() method gets called only once by GC(Garbage Collector) threads

2.1.9. Object class

- In java there is a special class named **Object**
- **Object** is a super class of all other classes by default.
- The object can be obtained using getObject() method.

For example

```
Object obj=getObject();
```

2.2. Visibility Control

The access modifiers in Java specifies 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:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **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.

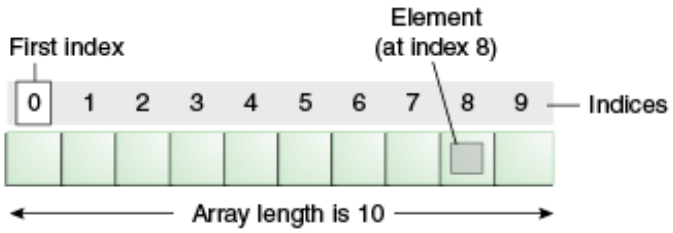
3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Example

```
class accessModifier
    {
        int a;     //default
        public int b;
        private int c;
        void fun(int val)
            {
                c=val;
            }
        void show( )
            {
                System.out.println("c= "+c);
            }
    }
    class Demo
        {
        public static void main(String args[])
        {
        accessModifier obj= new accessModifier();
            obj.a=100; //valid
            obj.b=200; //valid
            obj.c=300; //Error: private access
            obj.fun(300);  //valid
            obj.show(); //valid
        }
        }
```

2.3. Arrays

- An array is a collection of similar type of elements which has contiguous memory location.
- Java array is an object which contains elements of a similar data type.
- We can store only a fixed set of elements in a Java array.
- Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.
- Java provides the feature of anonymous arrays which is not available in C/C++.



Type of Array

There are two types of array.

- One/ Single Dimensional Array
- Two/Multidimensional Array

2.3.1. One/ Single Dimensional Array

The One dimensional array can be represented as

Array a[5]

10	20	70	40	50
0	1	2	3	4

Example

```
class Testarray
{
        public static void main(String args[])
        {
        int a[ ]=new int[5];      //declaration and instantiation
        a[0]=10;//initialization
        a[1]=20;
        a[2]=30;
        a[3]=40;
        a[4]=50;
        //traversing array
```

```
        for(int i=0;i<a.length;i++)  //length is the property of array
        System.out.println(a[i]);
        }
}
```

Output

```
10
20
70
40
50
```

We can declare, instantiate and initialize the java array together by:

```
int a[ ]={33,3,4,5};  //declaration, instantiation and initialization
```

Let's see the simple example to print this array.

```
class Testarray1
{
    public static void main(String args[])
    {
    int a[ ]={10,20,30,40,50};    //declaration, instantiation and initialization
    //printing array
    for(int i=0;i<a.length; i++)  //length is the property of array
    System.out.println(a[i]);
    }
}
```

Output

```
10
20
30
40
50
```

2.3.2. Two dimensional Array

In such case, data is stored in row and column based index (also known as matrix form).

Example to instantiate Multidimensional Array in Java

```
int[ ][ ] arr =new int[3][3]; //3 row and 3 column
```

Example to initialize Multidimensional Array in Java

```
arr[0][0]=1;
arr[0][1]=2;
arr[0][2]=3;
arr[1][0]=4;
arr[1][1]=5;
arr[1][2]=6;
arr[2][0]=7;
arr[2][1]=8;
arr[2][2]=9;
```

Example of Multidimensional Java Array

Let's see the simple example to declare, instantiate, initialize and print the 2Dimensional array.

```
//Java Program to illustrate the use of multidimensional array
  class Testarray2
  {
  public static void main(String args[])
      {
          //declaring and initializing 2D array
          int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
          //printing 2D array
          for(int i=0;i<3;i++)
          {
              for(int j=0;j<3;j++)
              {
              System.out.print(arr[i][j] +" ");
              }
              System.out.println();
          }
      }
  }
```

Output:

1 2 3

2 4 5

4 4 5

2.3.3. Array of objects

Syntax

objname[ ]= new classname( );

Program:-

- **Define a class 'student' with data members stdrn, name and marks. Accept data for five objects using array of objects and print it.**

```
// Reading the data from keyboard
import java.util.*;
class student
{
    int stdrn;
    String name;
    Double marks;
    char gender;
    void getdata()
    {
        Scanner in=new Scanner(System.in);
        System.out.println("Enter Student Roll no:");
        stdrn=in.nextInt();
        System.out.println("Enter Student Name:");
        name=in.next();
        System.out.println("Enter Student Marks:");
        marks=in.nextDouble();
        System.out.println("Enter the Gender:");
        gender=in.next().charAt(0);
    }
void display()
    {
        System.out.println("-----------------------------------");
```

```
                System.out.println("Student Roll no: "+stdrn);
                System.out.println("Student Name: "+name);
                System.out.println("Student Marks: "+marks);
                System.out.println("Student Gender: "+gender);
                System.out.println("-----------------------------------");
        }
}
class arrayofobject
{
        public static void main(String[] args)
        {
                student s[ ]=new student[3];
                System.out.println("Enter the Student data");
                for(int i=0; i<3 ;i++)
                        {
                                s[i]= new student();
                                s[i].getdata();
                        }
                System.out.println("Student Details is as follows");
                for(int i=0; i<3 ;i++)
                {
                        s[i].display();
                }
        }
}
```

Output:

```
Enter the Student data
Enter Student Roll no:
1
Enter Student Name:
rohini
Enter Student Marks:
```

45.67
Enter the Gender:
F
Enter Student Roll no:
2
Enter Student Name:
vaidehi
Enter Student Marks:
78.65
Enter the Gender:
F
Enter Student Roll no:
3
Enter Student Name:
Ashish
Enter Student Marks:
46.65
Enter the Gender:
M
Student is as follows

Student Roll no: 1
Student Name: rohini
Student Marks: 45.67
Student Gender: F

Student Roll no: 2
Student Name: vaidehi
Student Marks: 78.65
Student Gender: F

Student Roll no: 3

Student Name: Ashish

Student Marks: 46.65

Student Gender: M

2.4. Strings

Definition- String is a collection of characters.

In java, the string can be defined using two commonly used methods.

1) Using String class
2) Using StringBufferClass

2.4.1. String classes

- The syntax of String class is

  String string_variable;

Example:-

```
class stringDemo
{
public static void main(String args[])
        {
        String s="Welcome to java programming";
        Syste.out.println(" "+s);
        }
}
```

Output

Welcome to java programming

Operation on Strings using String class.

Following are some commonly defined methods by a string class.

Method	Description
s1.charAt(position)	Return the character present at the index position.
s1.compareTo(s2)	If s1<s2 then it returns positive. If s1>s2 then it return negative and if s1=s2 then it returns zero.

s1.concat(s2)	It returns the concatenated string of s1 and s2.
s1.equals(s2)	If s1 and s2 are both equal then it returns true.
s1.equalsIgnoreCase(s2)	By ignoring case, if s1 and s2 are equal then it returns true.
s1.indexof('c')	It returns the first occurrence of character 'c' in the string s1.
s1.indexof('c', n)	It returns the position of 'c' that occur at after nth position in string s1.
s1.lenght()	It gives the length of string s1.
String.valueof(var)	Converts the value of the variable passed to it into string type.

Program for string demonstrating String class methods.

```
class StringMethodsDemo
{
        public static void main(String[] args)
        {
                String s1="Java";
                char ch;
                //length();
                System.out.println("The length of string "+s1+ " = " +s1.length()); //4
                //charAt();
                ch=s1.charAt(2);
                System.out.println("The Char at 2 of string "+s1+ " = " +ch);  //v
                String s2="Java";
                String s3="Programming";
                //compareTo( )
                System.out.println("This is for CompareTo Method");
                System.out.println(s1.compareTo(s2)); //0
                System.out.println(s1.compareTo(s3)); // -6
                System.out.println(s3.compareTo(s1)); //6
                //equals();
                if(s1.equals(s2)==true)
                        {
                        System.out.println(s1+" and " +s2+ " are equals");
                        }
```

```
        //concatnation (join)
        System.out.println("Concatnation of " + s1+" and " +s3+ " = "+s1.concat(s3));
    }
}
```

Output:-

```
The length of string Java = 4
The Char at 2 of string Java = v
This is for CompareTo Method
0
-6
6
Java and Java are equals
Concatnation of Java and Programming = JavaProgramming
```

2.4.2. String Buffer

- The StringBuffer is a class which is alternative to the String class. But StringBuffer class is more flexible to use than the String class.
- That means, using StringBuffer we can insert some components to the existing string or modify the existing string but in case of String class once the string is defined then it remains fixed.
- Following are some simple methods used for StringBuffer-

Name of method	Description
append(String str)	Appends the string to the buffer
capacity()	It returns the capacity of the string buffer
charAt(int index)	It returns a specific character from the sequence which is specified by the index.
delete(int start, int end)	It deletes the characters from the string specified by the starting and ending index.
insert(int offset, char ch)	It inserts the character at the positions specified by the offset.
length()	It return the length of the string buffer.
setCharAt(int index, char ch)	The character specified by the index from the stringbuffer is set to ch.

setLength(int new_len)	It sets the length of the string buffer.
toString()	It converts the string representing data in this string buffer.
replace(int start, int end, String str)	It replaces the characters specified by the new string
reverse( )	The character sequence is reversed.

❖ **Difference between String & StringBuffer**

Sr. No	String	StringBuffer
1	The length of string object is fixed.	The length of StringBuffer can be increased.
2	The String object is immutable, that means we can not modify the string once created.	The StringBuffer class is immutable.
3	It is slower in performance.	It is faster in performance.
4	It consumes more memory.	It consumes less memory.

Programs

1) **Performs the following string/ string buffer operations, write java program**
   a) **Accept a password from user**
   b) **Check if password is correct then display "Good", else display "Wrong".**
   c) **Display the password in reverse order**
   d) **Append password with "Welcome".**

```
import java.util.*;
class  StringBufferDemo
{
    public static void main(String[] args)
    {
            String s="MSBTE-S-21";
            Scanner sc=new Scanner(System.in);
            System.out.print("Enter the password: ");
            String pwd=sc.nextLine();
```

```
                if(s.compareTo(pwd)==0)
                        System.out.println("Good");
                else
                        System.out.println("Wrong");

                StringBuffer s1=new StringBuffer(pwd);
                s1=s1.reverse();
                System.out.println("The reversed string is :"+s1);

                StringBuffer s2=new StringBuffer(pwd);
                s2=s2.append(" Welcome: ");
        System.out.println(" "+s2);
        }
    }
```

Output:-

```
java StringBufferDemo
Enter the password: MSBTE-S-21
Good
The reversed string is :12-S-ETBSM
MSBTE-S-21 Welcome:
```

2) Write a simple java program to find the reverse string and check the weathered the entered string is palindrome or not?

```
import java.util.*;
class ReversePalindrome
{
        public static void main(String[] args)
        {
                String s1;
                Scanner sc=new Scanner(System.in);
                System.out.print("Enter the String: ");
```

```
                s1=sc.nextLine();  //nextLine()-- Accepting blank spaces

                StringBuffer s2=new StringBuffer(s1);
                s2.reverse();
                System.out.println("-----------------------------------");
                System.out.println("Orignal String: "+s1);
                System.out.println("Reversed String: "+s2);
                System.out.println("Checking for Palindrome String: ");
                System.out.println("-----------------------------------");
                if(s1.equals(s2.toString()))
                        {
                        System.out.println(s1+" is Palindrome String:");
                        }
                else
                        System.out.println(s1+" is Not Palindrome String");
        }
}
Enter the String: mom
---------------------------------
Orignal String: mom
Reversed String: mom
Checking for Palindrome String:
---------------------------------
mom is Palindrome String

Enter the String: Gunwant
-----------------------------------
Orignal String: Gunwant
Reversed String: tnawnuG
Checking for Palindrome String:
-----------------------------------
Gunwant is Not Palindrome String
```

2.5. Vector

- The vector is a class in java that implements the dynamic array. This class stores any no. of objects of any data type.
- This class is defined by java.util package.
- In vector one can store the elements of any data type. That means in a single vector you store integer, string, double or any other data type elements altogether.
- Vector can be created like this-

```
Vector vectobj= new Vector( );   //declaring vector without size
Vector vectobj= new Vector(5);  //declaring vector with size
```

Advantages of vectors over array

1) Vectors contains elements of varying data types.
2) The size of vector can be changed whenever required.
3) It can store simple objects.

Following are the some most commonly used methods of vector class.

Methods	Description
void addElement(object)	For adding the element in the vector
Object elementAt(int index)	Return the element present at specified location(index) in vector.
void insertElementAt(object obj, int pos)	For inserting the element in the vector specified by its position.
boolean removeElement(object ele)	Removes the specified element
void removeAllElements()	For removing all the elements from the vector.
void removeAllElements(int pos)	The elements specified by its position gets deleted from the vector.
int capacity()	Returns the capacity of the vector.
int size()	It returns the total no. of elements present in the vector.
boolean isEmpty()	Return true if the vector is empty.
Object firstElement()	Return the first element of the vector.
int indexOf(object ele)	Returns the index of corresponding element in the vector.
void setSize(int size)	This method is for setting the size of the vector.

Program 1: Write a program, to create a vector with seven elements as {10, 30, 50, 20, 40, 10 , 20}. Remove element at 3rd and 4th position. Insert new element at 3rd position. Display the original and current size of the vector.

```
import java.util.*;
class VectorDemo
{
	public static void main(String[] args)
	{
		Vector v1= new Vector(7);
		v1.addElement(10);
		v1.add(30);
		v1.add(50);
		v1.add(20);
		v1.add(40);
		v1.add(10);
		v1.add(20);
		System.out.println("The elements in the vector are:"+v1);
		System.out.println("The orignal size of vector : "+v1.size());
		System.out.println("Removing the Third Element ");
		v1.remove(2);  //3rd element
		System.out.println("Removing the Fouth Element ");
		v1.remove(3);  //4rd element
		System.out.println("Now The elements in the vector are:"+v1);

		System.out.println("Inserting element 100 at 3rd position:");
		v1.insertElementAt(100, 2); //inserting 100  at 3rd position
		System.out.println("Now The elements in the vector are:"+v1);
		System.out.println("The current size of vector : "+v1.size());
	}
}
```

Output:-

The elements in the vector are:[10, 30, 50, 20, 40, 10, 20]

The orignal size of vector : 7

Removing the Third Element

Removing the Fouth Element

Now The elements in the vector are:[10, 30, 20, 10, 20]

Inserting element 100 at 3rd position:

Now The elements in the vector are:[10, 30, 100, 20, 10, 20]

The current size of vector : 6

Program 2:- Write a program to add 2 integer, 2 string and 2 float objects to a vector. Remove element specified by user and display the list.

2.6. Wrapper classes

- Wrapper classes are those classes that allow primitive data types to be accessed as objects.
- The Wrapper class is one kind of Wrapper around a primitive data type.
- The wrapper classes represent the primitive data types in its instances of the class.
- Following tables shows various primitive data types and the corresponding wrapper classes.

Primitive data type	Wrapper class
boolean	Boolean
byte	Byte
char	Char
double	Double
float	Float
int	Integer
long	Long
short	Short
void	Void

2.6.1. Uses of Wrapper class

- Wrapper classes are used to convert numeric strings into numeric values.
- Wrapper classes are used to convert numeric values to string.
- Using the typeValue() method we can retrieve value of the object as its primitive data type.

Program:-

1. Define wrapper class. Give the following wrapper class methods with syntax and use.

1) To convert integer number to string

2) To convert numeric string to integer number.

```
class  WrapperDemo
{
        public static void main(String[] args)
        {
                System.out.println("Integer number to string Converstion:");
                int i=100;
                String s=Integer.toString(i);
                System.out.println("int value:"+i);
                System.out.println("Equivalent to string:"+s);
                System.out.println("----------------------------------------");
                System.out.println("String to Integer Converstion:");
                String s1="200";
                int n=Integer.parseInt(s1);
                System.out.println("String value:"+s1);
                System.out.println("Equivalent to int:"+n);
        }
}
```

Output:-

```
Integer number to string Converstion:
int value:100
Equivalent to string:100
----------------------------------------
```

String to Integer Converstion:

String value:200

Equivalent to int:200

2.7. Enumerated Types

- The enumerated data types can be denoted by the keyword enum.
- The enum helps to define the user defined data type.
- The value can also be assigned to the elements in the enum data type.

Program for enum Demonstration

```
class enumDemo
{
        enum Color
        {
                Red, Green, Blue;
        }
        public static void main(String[] args)
        {
                Color c1=Color.Red;
                System.out.println(c1);
        }
}
```

Output

Red

End of Unit-II