```
**************************************************
*  Class Name : VJTech Academy , Maharashtra     *
*  Author Name: Vishal Jadhav Sir                *
*  Mobile No  : 9730087674                       *
**************************************************
=====================================
UNIT-II :Classes, Objects and Methods
=====================================
```

***Defining the Class:
======================
- Class is a collection of similiar types of objects.
- Class is a user defined data type.
- Class contains instance variables and methods.
- Class shows data abstraction and data encapsulation properties.
- Syntax:

```
        class Class_Name
        {
                datatype instance_variable1;
                datatype instance_variable2;
                datatype instance_variableN;
                returntype Method_Name1(parameter_list)
                {
                        //body of method
                }
                returntype Method_Name2(parameter_list)
                {
                        //body of method
                }
                returntype Method_NameN(parameter_list)
                {
                        //body of method
                }
        }
```

- Example:

```
        class Addition
        {
                int a,b,c;
                void getdata()
                {
                        a=100;
                        b=50;
                }
                void display()
                {
                        c=a+b;
                        System.out.println("Addition="+c);
                }
        }
```

Note:
-----
1) No semicolon after the closing curly bracket.
2) ClassName is valid java identifier
3) The instance variable and methods defined inside the class is known as member of

the class.
4) Class declaration only creates template, it does not create an actual object.
5) Memory should not be allocated for the data members of the class.
6) Method definition should present inside the class only.

***Creating an Objects:
=======================
- An object is instance of the class.
- When object is created then memory will be allocated for the instance variables of the class.
- We can create N no of objects from the class.
- Objects are created using new operator.
- The new operator creates an objects of the class and return reference to the object
  created.

1) Declaration of Object:
```
   Syntax : ClassName ObjectName;
   Example: Addition a1;
```

2) Instantiate of Object:
```
   Syntax : ObjectName=new ClassName(Parameter_list);
   Example: a1=new Addition();
```

- Combining the above two steps:
```
  Syntax : ClassName ObjectName=new ClassName(Parameter_list);
  Example: Addition a1=new Addition();
```

***Accessing Class Members:
===========================
- Object contain data members and member function.
- So we can access them by using object name and dot operator.
- Syntax:
```
        ObjectName.Variable_Name=value;
        ObjectName.Method_Name(Parameter_list);
```
- Example:
```
        a1.a=100;
        a1.b=200;
        a1.getdata();
        a1.display();
```

***Array of objects:
====================
- Array of object is the collection of objects of same class.
- Instead of creating mutiple objects, it would be better to create array of

objects.
- Array index should begin with 0 and end with size-1.
- Object Name is same but its indexes are different.
- Syntax:
```
        ClassName ObjectName[]=new ClassName[SIZE];
```
- Example:
```
        VJTech v[]=new VJTech[5];
        for(int i=0;i<5;i++)
        {
       v[i]=new VJTech();//Assigning object to individual reference in the array.
        }
```
==================
Constructors
==================
- Constructor is a special member function of the class.
- It is used to initialize the data members of the objects.
- There is no any return type for the constructor.
- Constructor name and class name both are same.
- Constructor automatically called when object is created.
- Constructor is used for creation of an object.
- Suppose, in our class we have not defined constructor then system will supply the default constructor for creation of an objects.
- There are three different types of the constructor:

1) Default constructor
2) Parameterized constructor
3) Copy constructor

1) Default constructor:

-----------------------
- When constructor does not takes any parameters then it is called as default constructor.
- Syntax:
```
                class ClassName
                {
                        ClassName()
                        {
                                //body of constructor.
                        }
                }
```
2) Parameterized constructor:

-----------------------
- When constructor takes any parameters then it is called as Parameterized constructor.
- Syntax:
```
                class ClassName
                {
                        ClassName(parameter_list)
                        {
                                //body of constructor.
                        }
                }
```

2) Copy constructor:

- To initialize data members of the object, we are passing another object as argument is called as copy constructor.
- When constructor takes reference of its class as paramater then it is called as copy constructor.
- Syntax:

```
                class ClassName
                {
                        ClassName(ClassName ObjectName)
                        {
                                //body of constructor.
                        }
                }
```

- Example:

```
//copy constructor
class Item
{
        int x;
        Item()
        {
                x=100;
        }
        Item(Item m)
        {
                x=m.x;
        }
        void display()
        {
                System.out.println("Value of X : "+x);
        }
        public static void main(String args[])
        {
                Item i1=new Item();
                Item i2=new Item(i1);
                i1.display();
                i2.display();
        }
}
```

***Constructor overloading:
==========================
- Constructor names are same but its arguments are different.
- Example:

```
//constructor overloading
class Room
{
        float length;
        float width;
        Room(float x)
        {
                length=x;
                width=20.50f;
        }
        Room(float m,float n)
        {
                length=m;
                width=n;
        }
        void display()
        {
                float room_area;
                room_area=(length*width);
                System.out.println("Area of Room : "+room_area);
        }
        public static void main(String args[])
        {
                Room r1=new Room(10.10f);
                Room r2=new Room(50.50f,23.50f);

                r1.display();
                r2.display();
        }
}
```

=================
Static Members
=================

- Static members can be data and methods.
- When we create objects of the class then separate memory allocated for each data members of the class.
- But sometime, there could be situation where we want to keep one variable common between all objects.
- In this case, we can make that variable as static.
- When we make variable as static then only one copy of that variable created in computer memory and all objects share it commonaly.
- Static members comes under the scope of class.
- We can access static members using classname and dot operator.
- Inside the body of static member function, we can access only other static data members.
- There is no need to define static data member outside the class.
- By default static variable contain zero value.
- Example:

```
//static data member and static member function
class StaticDemo
{
        int no;
        static int count;               //static variable
        void getdata(int x)
```

```
	{
		no=x;
		count++;
	}
	void display_no()
	{
		System.out.println("Value of no="+no);
	}
	static void display_count() //static method
	{
		System.out.println("Value of count="+count);
	}
	public static void main(String args[])
	{
		StaticDemo s1=new StaticDemo();
		StaticDemo s2=new StaticDemo();
		StaticDemo s3=new StaticDemo();

		s1.getdata(100);
		s2.getdata(200);
		s3.getdata(300);

		System.out.println("Object s1 :");
		s1.display_no();
		StaticDemo.display_count();

		System.out.println("Object s2 :");
		s2.display_no();
		StaticDemo.display_count();

		System.out.println("Object s3 :");
		s3.display_no();
		StaticDemo.display_count();
	}
}
```

```
OUTPUT:
---------
Object s1 :
Value of no=100
Value of count=3
Object s2 :
Value of no=200
Value of count=3
Object s3 :
Value of no=300
Value of count=3
```

==

Visibility Control/Access Specifiers Parameter:

===

- Java provides four types of visibility control.
1) public
2) private
3) protected
4) Friendly Access

public:

- Any variables and methods declared as public, it can be accessible outside the class.

private:

- Those members are declared as private, it can accessible within the class in which they are declared.
- It cannot be inherited in its subclass.

protected:

- Those members are declared as protected, it can accessible in same class and its immediate sub-class.

Friendly Access:

- In the situation where no access modifier is specified then by default all members considered as friendly access level.
- There is basic difference between public and friendly access is that public members accessible anywhere but friendly access member available in same package not outside the package.

===================
Arrays
===================
- Normally, one variable can store one value at a time.
- But sometimes, we need to store multiple values then creation of multiple variables is not a better solution.
- In this case, we can create array variable.
- Array variable name is same but it will store multiple values.
- Array is a collection of similiar types of elements.
- Array index should begin with Zero and end with SIZE-1.
- Array elements are stored in continues memory location.
- There are three different types of array:
1) One dimensional array
2) Two dimensional array
3) Multi-dimensional array

- Creating an Array involes following steps:
I) Declaration of Array:
- Syntax:
```
        form-1  datatype ArrayName[];
        form-2  datatype [] ArrayName;
```
- Example:
```
                        int marks[];
                        float average[];
                        int [] sum;
```

II) Creation of Array:
- After declaration of array, we need to create it in the memory. Java allows us to create array using new operator only.
- Syntax:
```
                        ArrayName=new datatype[SIZE];
```
- Example:
```
                        marks=new int[5];
                        average=new float[10];
```

- Combining step-I and step-II:

Syntax: datatype ArrayName[]=new datatype[SIZE];

Example: int marks[]=new int[5];

Array Initialization:

- In this step, we will put values into the array.
- This process is known as initialization.
- We can initialize array elements using index number and subscripts.
- Syntax:
```
                        ArrayName[index]=value
```
- Example:
```
                        marks[0]=99;
                        marks[1]=78;
                        marks[2]=65;
```
- We can also initialize array elements automatically at the time of declaration.
- Syntax:
```
                        datatype ArrayName[]={list of values};
```
- Example: int number[]={10,20,30,40,50};

Array Length:

- Java provides predefined method length to calculate length of array.
- Example:
```
                int a[]={10,20,30};
                int len=a.length;
                len=3;
```
- Example-1:
```
class OneDArray
{
        public static void main(String args[])
        {
                int a[]=new int[5];
                a[0]=10;
                a[1]=20;
                a[2]=30;
                a[3]=40;
                a[4]=50;
                for(int i=0;i<a.length;i++)
                {
                   System.out.println("Element present at index "+i+" is "+a[i]);
                }
        }
}
/*
Element present at index 0 is 10
Element present at index 1 is 20
Element present at index 2 is 30
Element present at index 3 is 40
Element present at index 4 is 50
*/
```

- Example-2:

```
import java.util.*;
class OneDArray1
{
        public static void main(String args[])
        {
                int a[]=new int[5];
                int i;
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Five Array Elements:");
                for(i=0;i<5;i++)
                {
                   a[i]=sc.nextInt();
                }
                System.out.print("Array Elements are : ");
                for(i=0;i<5;i++)
                {
                   System.out.print(a[i]+" ");
                }
        }
}
/*
Enter Five Array Elements:
100
200
300
400
500
Array Elements are : 100 200 300 400 500
*/
```

- Example-3:

```
//one dimensional Array : reverse the elements of array
```

```
import java.util.*;
class OneDArray2
{
	public static void main(String args[])
	{
		int a[]=new int[5];
		int i;
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter Five Array Elements:");
		for(i=0;i<5;i++)
		{
		   a[i]=sc.nextInt();
		}
		System.out.print("Display Array Elements in Reverse Order: ");
		for(i=a.length-1;i>=0;i--)
		{
		   System.out.print(a[i]+" ");
		}
	}
}
/*
Enter Five Array Elements:
100
200
300
400
500
Display Array Elements in Reverse Order: 500 400 300 200 100
*/
```

```
======================
Two Dimensional Array
======================
```

- Two Dimensional array is used to maintain elements in rows and column format.
- We can represent data in tabular format.
- Rows index should begin with 0 and end with size-1.
- Columns index should begin with 0 and end with size-1.
- Creating an Array involes following steps:

I) Declaration of Array:

- Syntax:

```
	form-1  datatype ArrayName[][];
	form-2  datatype [][] ArrayName;
```

- Example:

```
			int marks[][];
			float average[][];
			int [][] sum;
```

II) Creation of Array:

- After declaration of array, we need to create it in the memory. Java allows us to create array using new operator only.

- Syntax:
```
                        ArrayName=new datatype[ROWS][COLUMNS];
```
- Example:
```
                        marks=new int[3][3];
                        average=new float[10][10];
```

- Combining step-I and step-II:

Syntax: `datatype ArrayName[][]=new datatype[ROWS][COLUMNS];`
Example: `int marks[][]=new int[3][3];`

Array Initialization:

- In this step, we will put values into the array.
- This process is known as initialization.
- We can initialize array elements using index number and subscripts.
- Syntax:
```
                        ArrayName[Row-Index][Column-Index]=value;
```
- Example:
```
                        int a[][]=new int[3][3];
                        a[0][0]=10;
                        a[0][1]=20;
                        a[0][2]=30;
                        a[1][0]=40;
                        a[1][1]=50;
                        a[1][2]=60;
                        a[2][0]=70;
                        a[2][1]=80;
                        a[2][2]=90;
```

- We can also initialize array elements automatically at the time of declaration.
- Syntax:
```
                        datatype ArrayName[][]={list of values};
```
- Example:
```
            int number[][]={{10,20,30},
                                                            {40,50,60},
                                                            {70,80,90}
                                                       };
```

- Program:
```
import java.util.*;
class TwoDArray
{
        public static void main(String args[])
        {
                int a[][]=new int[3][3];
                int i,j;
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter 3*3 Array Elements:");
                for(i=0;i<3;i++)
                {
                        for(j=0;j<3;j++)
                        {
                                a[i][j]=sc.nextInt();

                        }
                }
                System.out.println("Your 3*3 Array Elements: ");
                for(i=0;i<3;i++)
                {
                        for(j=0;j<3;j++)
                        {
                                System.out.print(a[i][j]+"  ");
                        }
                        System.out.println();
                }
        }
}
/*
Enter 3*3 Array Elements:
10
20
30
40
50
60
70
80
90
Your 3*3 Array Elements:
10  20  30
40  50  60
70  80  90
*/
```

=================
Vectors
=================
- Vector is an extensible array.
- Vector is a collection of objects and it can be retrived by using index number.
- Array is a collection of similiar types of elements but its size is fixed.
- But vector is a collection of objects but its size is not fixed.
- Vector class provides array of variable size.
- The main difference between vector and array : Vector automatically grow when they run out of space.
- Vectors class provides extra method for adding and removing elements.
- The class is used to create dynamic array known as Vector that can holds objects of any type and any numbers.
- Vector is a predefined class which is present under java.util package.

- Vectors are created like array as follow:
1) Declaration of Vector without size.

```
   Vector VectorName=new Vector():
```

```
2) Declaration of Vector with size.
   Vector VectorName=new Vector(10):

Advantages of Vector over arrays:
----------------------------------
1) It is convenient to the vector to store objects.
2) Vector is used to store multiple objects and its size not fixed.
3) Objects can be added and deleted from the vector whenever required.

DisAdvantages of Vector
-----------------------
1) It can not directly store simple data type only objects can be stored.
2) To store simple data type value in the vector, they must be converted into
objects.
3) Wrapper classes are required for above conversion.

Important Vector Methods:
-------------------------
Vector v1=new Vector();

1) v1.addElement(item) - Adds the item to the vector at the end.
2) v1.elementAt(10)    - Gives the name of 10th object.
3) v1.size()           - Gives the number of objects present.
4) v1.removeElement(item) - Removes the specified item from the vector.
5) v1.removeElementAt(n)  - Removes the item stored in nth position.
6) v1.removeAllElements() - Removes all the elements in the list.
7) v1.copyInto(array)     - Copies all items from vector to array.
8) v1.insertElementAt(item,n) - Insert the item at nth position.
----------
Program-1:
----------
/*Write a program to create vector with six elements (10,30,60,70,80,100). Removes
element 3rd and 4th position. Insert new element at 3rd position. Display the
original and current size of vector
*/
import java.util.*;
class VectorDemo1
{
        public static void main(String args[])
        {
                Vector v1=new Vector();
                v1.addElement(new Integer(10));
                v1.addElement(new Integer(30));
                v1.addElement(new Integer(60));
                v1.addElement(new Integer(70));
                v1.addElement(new Integer(80));
                v1.addElement(new Integer(100));
                System.out.println("Initial Vector Size = "+v1.size());
                v1.removeElementAt(3);
                v1.removeElementAt(4);
                v1.insertElementAt(new Integer(150),3);
                System.out.println("Final Vector Size = "+v1.size());

        }
}
/*
Initial Vector Size = 6
Final Vector Size = 5
*/
------------
Program-2:
-----------
import java.util.*;
class VectorDemo2
{
        public static void main(String args[])
        {
                Vector v1=new Vector();
                for(int i=0;i<args.length;i++)
                {
                        v1.addElement(args[i]);
                }
                System.out.println("Vector of Size = "+v1.size());
                System.out.println("Your Vector Elements:");
                for(int i=0;i<v1.size();i++)
                {
                        System.out.println(v1.elementAt(i));
                }
        }
}
/*
F:\Academic 2022\JavaBatch2022\UNIT-II>java VectorDemo2 10 20 30 40 50
Vector of Size = 5
Your Vector Elements:
10
20
30
40
50
*/
```

================

Wrapper Classes

================

- As you already aware about, in java some concepts only works on objects.
- Let take one example - Vector : Vector cannot handle primitive data types like int,float,long,char and double.
- Using Wrapper classes, we can convert primitive data type into objects.
- In Java different wrapper clases are given which is present under java.lang package.

- Following table shows wrapper classes for converting simple types:

Simple Type	Wrapper Class
1) boolean	Boolean
2) char	Character
3) float	Float
4) double	Double
5) int	Integer
6) long	Long

- The wrapper classes have number of unique method for handling primitive data type and objects.

I)Converting primitive numbers to object numbers using constructor methods

Constructor Calling	Conversion Action
1)Integer x=new Integer(i)	Primitive integer to Integer object
2)Float y=new Float(f)	Primitive float to Float object.
3)Double z=new Double(d)	Primitive double to Double object.
4)Long w=new Long(l)	Primitive long to Long object.
5)Character v=new Character(c)	Primitive char to Character object.

Note: i, f, d, l, c are primitive data type values which denoting int,float,double,long and char data types.

Example:

```
                int i=100;                                      //Primitive integer i
                Integer x=new Integer(i):   // Integer Object x

                float f=100.10;                         //Primitive float f
                Float y=new Float(f):      // Float Object y
```

II)Converting Object numbers to primitive numbers using typeValue() method

Method Calling	Conversion Action
1)int i=x.intValue();	Object to primitive int
2)float f=y.floatValue();	Object to primitive float
3)double d=z.doubleValue();	Object to primitive double
4)long l=w.longValue();	Object to primitive long
5)char c=v.charValue();	Object to primitive char

III)Converting numbers to string using toString() method

Method Calling	Conversion Action
1) str=Integer.toString(i)	Primitive Integer to String

2) str=Float.toString(f)	Primitive float to String
3) str=Double.toString(d)	Primitive double to String
4) str=Long.toString(l)	Primitive Long to String

IV)Converting string objects to numeric objects using ValueOf() method

Method Calling	Conversion Action
1) DoubleVal=Double.ValueOf(str);	Converting string to Double Object
2) floatVal=Float.ValueOf(str);	Converting string to Float Object
3) intVal=Integer.ValueOf(str);	Converting string to Integer Object.
4) longVal=Long.ValueOf(str);	Converting string to Long object

V)Converting numeric string to primitive numbers using parsing methods

Method Calling	Conversion Action
1) int i=Integer.parseInt(str)	Converts string to primitive integer
2) long l=Long.parseLong(str)	Converts string to primitive long
3) float f=Float.parseFloat(str)	Converts string to primitive float
4) double d=Double.parseDouble(str)	Converts string to primitive double.

Program:

```
class WrapperClassDemo
{
        public static void main(String args[])
        {
                int i=100;
                Integer ii=new Integer(i);
                System.out.println("Primitive Integer Value = "+i);
                System.out.println("Object Integer Value = "+ii);
        }
}
/*
Primitive Integer Value = 100
Object Integer Value = 100
*/
```

=========================
Strings:
=========================
- String is a sequence of characters.
- Collection of characters is known as String.
- String is a predefined class which is present under java.lang package.
- String should be represented by using double quotation.
- The easiest way to represent string in java is by using character array.
- Example:

```
                        char name[]=new char[4];
                        name[0]='J';
                        name[1]='a';
                        name[2]='v';
                        name[3]='a';
```
- By using above approach, we require lot of overhead to manage the string.
- For example, copy one character array into another array is difficult due to size issues.
- In java, we have one good approach to manage string i.e creation of string class object.
- This class provides lots of good methods to manipulate string.
- Syntax:
```
                String strName="Value";
                OR
                String strName=new String("String");
```
- Example:
```
                String firstName="Vishal";
                String firstName=new String("Vishal");
```
- In java, we use + operator as concatenation operator.

String Arrays:

- We can also create and use arrays that contain strings.
- Example:
```
                String x[]=new String[5];
```
- In above example, we can see string name is x and its size is 5. It means we can hold five string constants.

String Methods:

- String class provides different methods.

Method Names	Working
1) s1.toLowerCase();	convert the string s1 to lowercase
2) s1.toUpperCase();	convert the string s1 to Uppercase
3) s1.replace('x','y');	Replace all x with y in given string s1.
4) s1.trim();	Remove the spaces present at begin & end
5) s1.equals(s2)	Return true if s1 is equal to s2
6) s1.equalsIgnoreCase(s2)	Return true if s1==s2. It ignore the case.
7) s1.length()	Gives the length of s1.
8) s1.charAt(n)	Gives the nth character of s1.
9) s1.compareTo(s2)	s1==s2 : 0,s1>s2 : +ve value,s1<s2 : -ve value
10) s1.concat(s2)	Concatenates s1 and s2.
11) s1.substring(n)	Gives the substring starting from nth character.
12) s1.substring(n,m)	Gives substring starting from n & upto m characters.
13) p.toString()	Create string representation of object p.
14) s1.indexof('x')	Return index of x character in the string s1.
15) s1.indexof('x',n)	Return index of x which is occurred after nth character
16) String.valueof(p)	It will create string object of the parameter p.

- Example:

```
//String class
class StringDemo
{
        public static void main(String args[])
        {
                String str=new String("VJTech");

                System.out.println("Value of str ="+str);
                System.out.println("Length of str ="+str.length());
                System.out.println("To Lower Case ="+str.toLowerCase());
                System.out.println("To Upper Case ="+str.toUpperCase());
                System.out.println("Character present at 2 index ="+str.charAt(2));

                System.out.println("Concatenation="+str.concat("Academy"));
                System.out.println("Index of e character ="+str.indexOf('e'));
                System.out.println("Equals method ="+str.equals("VJTech"));
                System.out.println("CompareTo method ="+str.compareTo("VJTech"));

        }
}
/*
Value of str =VJTech
Length of str =6
To Lower Case =vjtech
To Upper Case =VJTECH
Character present at 2 index =T
Concatenation=VJTechAcademy
Index of e character =3
Equals method =true
CompareTo method =0
*/
```

StringBuffer class:
====================
- StringBuffer is a peer class of String.
- String class creates fixed length of string.
- StringBuffer class creates flexible length of string.
- We can insert characters and substring in the middle of the string.
- We can append another string to the end.
- Some methods of String class also supported in StringBuffer class.
- Commonly used StringBuffer methods:

1) s1.setCharAt(n,'x')	- modifies the nth character to x.
2) s1.append(s2)	- Appends the string s2 to s1 at the end.
3) s1.insert(n,s2)	- Inserts the string s2 at the position n of the string s1.
4) s1.setLength(n)	- Sets the length of string s1.
5) s1.length()	- Gives the length of s1.
6) s1.charAt(n)	- Gives the nth character of s1.

```
7) s1.equals(s2)         - Return true if s1 is equal to s2
- Example:
//StringBuffer class

class StringBufferDemo
{
        public static void main(String args[])
        {
                StringBuffer str=new StringBuffer("VJTech");

                System.out.println("Original String :"+str);
                System.out.println("Length of String :"+str.length());
                for(int i=0;i<str.length();i++)
                {
                        System.out.println("Character at position "+i+" is 
"+str.charAt(i));
                }
                str.setCharAt(3,'T');
                System.out.println("Modified String :"+str);
                str.append("Academy");
                System.out.println("Appended String :"+str)     ;
        }
}
/*
Original String :VJTech
Length of String :6
Character at position 0 is V
Character at position 1 is J
Character at position 2 is T
Character at position 3 is e
Character at position 4 is c
Character at position 5 is h
Modified String :VJTTch
Appended String :VJTTchAcademy
*/
```