Arrays, String and Vector

Course: CSL304

Prof. Juhi Ganwani

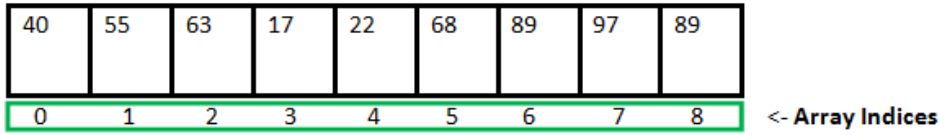
CONTENT

1. Array
2. Strings
3. String Buffer
4. Vectors

Arrays

- **Java array** is an object which contains elements of a similar data type.
- we can create an array of integers, or an array of characters, or an array of String objects, etc.
- Elements of an array are stored in a contiguous memory location.
- Two types:

1. Single dimensional array
2. Multi dimensional array



Array Length = 9
First Index = 0
Last Index = 8

Single dimensional array

- Elements in array is identified by one single index

Two steps:

1. Declaration:

type arrayname[]; OR type[] arrayname;

eg: int a[]; OR int[] a;

2. Creating memory location:

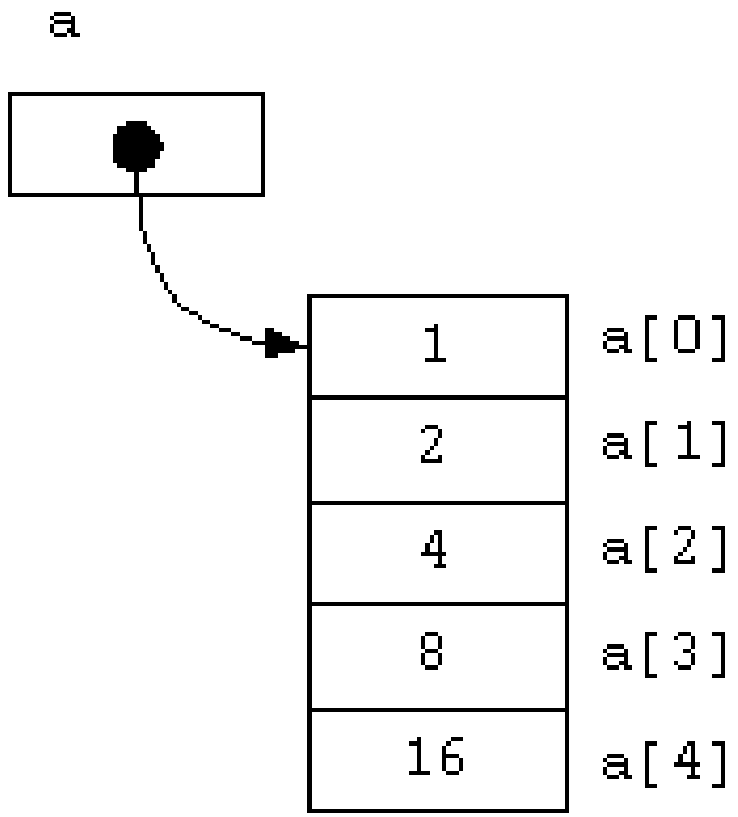
arrayname = new type[size];

eg: a = new int[5];

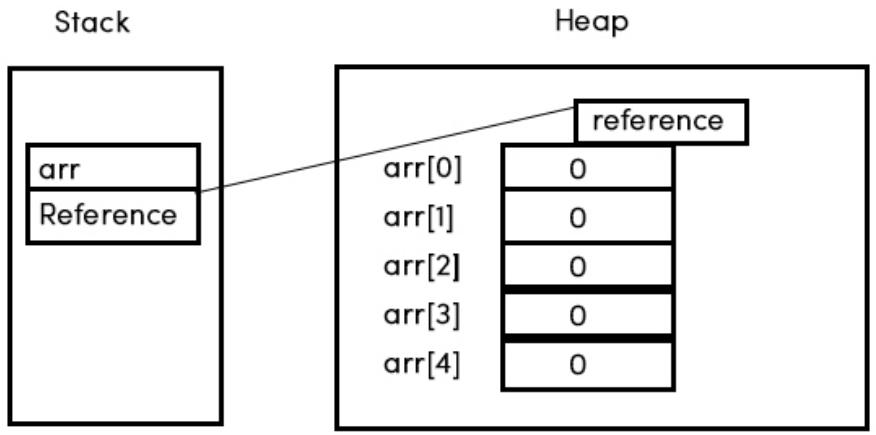
We can combine above two steps

type arrayname[] = new type[size];

eg: int a[]=new int[5];



int[] arr = new int[5];



Initialization

Syntax:

type arrayname[] = { list of values };

eg:

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

OR

arrayname[subscript] = value;

eg:
a[0]=12

- It is possible to assign an array object to another
  eg:

  int b[];

  b=a;
- We can find the length of the array using member length

eg:

int asize=a.length;

Program to calculate sum of array elements

```
public class Sum
{
	public static void main(String[] args)
	{
		int my_array[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
		int sum = 0;

		//for (int i : my_array)
			//sum+=i;
		for(int i=0;i<my_array.length;i++)
			sum += my_array[i];
		System.out.println("The sum is " + sum);
	}
}
```

To find largest number in an array

```
class Largest
{
public static void main(String args[])throws IOException
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
int n,max;
System.out.println("Enter the size of array");
n=Integer.parseInt(br.readLine());
int a[]=new int[n];
System.out.println("Enter elements of an array");
for(int i=0;i<n;i++)
        a[i]=Integer.parseInt(br.readLine());
max=a[0];
for(int i=1;i<n;i++)
{
        if(max<a[i])
        max=a[i];
}
System.out.println("Largest element is"+max);
}
}
```

Inserting an element in a specified position

```
import java.util.Arrays;
class Position
{
public static int[] insert(int[] a, int key, int index)
{
int[] result = new int[a.length + 1];
for (int i = 0; i < index; i++)
{
result[i] = a[i];
}
result[index] = key;

for (int i = index + 1; i <= a.length; i++)
{
result[i] = a[i - 1];
}
return result;
}
public static void main(String[] args)
{
        int[] a = {11,22,44,55};
        int key =  33;
        int index = 2;

        a = insert(a, key, index);

        System.out.println(Arrays.toString(a));
}
}
```

Array of objects

- Array of objects is created just like an array of primitive type data items in the following way.

```
Subject[] arr = new Subject[6];
```

- Subject array contains six memory spaces each of size of subject class in which the address of six subject objects can be stored.
- Objects have to be instantiated using the constructor of the class.

Example

```
class Subject
{
   int id;
   String name;
   Subject(int id, String name)
   {
      this.id = id;
      this.name = name;
   }
}
```

```
public class SubjectMain
{
   public static void main (String[] args)
   {
      Subject[] arr=new Subject[5];
      // initializing elements
      arr[0] = new Subject(1,"CG");
      arr[1] = new Subject(2,"DLCA");
      arr[2] = new Subject(3,"DS");
      arr[3] = new Subject(4,"DSGT");
      arr[4] = new Subject(5,"OOP");
      // accessing the elements of the specified array
      for (int i = 0; i < arr.length; i++)
         System.out.println("Element at " + i + " : " + arr[i].id +" "+ arr[i].name);
   }
}
```

Multi dimensional array

- A multidimensional array is an array of arrays.
- Each element of a multidimensional array is an array itself.

eg:

int a[][]=new int[3][4];

Or

int a[][];

a=new int[3][4];

- It is a 2-dimensional array, that can hold a maximum of 12 elements

	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Initializing multi dimensional array

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

Or

```
int a[1][0]=4;
```

Or

Using for loops

	Column 1	Column 2	Column 3	Column 4
Row 1	1 a[0][0]	2 a[0][1]	3 a[0][2]	
Row 2	4 a[1][0]	5 a[1][1]	6 a[1][2]	9 a[1][3]
Row 3	7 a[2][0]			

Initializing array a

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
for(i=0;i<4;i++)
        for(j=0;j<4;j++)
        {
                if(i==j)
                        a[i][j]=1;
                else
                        a[i][j]=0;
        }
```

Input

```
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
a[i][j]=Integer.parseInt(br.readLine());
}
}
```

Output

```
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
System.out.println(a[i][j]+” “);
}
System.out.println();
}
```

Addition of two matrix

```
public class AddingTwoMatrices
{
        public static void main(String args[])
        {
                int a[][]={{1,2,3},{4,5,6},{7,8,9}};
                int b[][]={{1,1,1},{1,1,1},{1,1,1}};
                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(c[i][j]+" ");
            }
            System.out.println();
 }
}
}
```

To find & display the sum of diagonal elements of square matrix

```
import java.io.*;
class SumOfDiagonal
{
public static void main(String args[])throws IOException
{
int a[][],n,sum;
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter size of a square matrix");
n=Integer.parseInt(br.readLine());
a=new int[n][n];
System.out.println("Enter elements of a matrix);
for(int i=0;i<n;i++)
{for(int j=0;j<n;j++)
a[i][j]=Integer.parseInt(br.readLine());
sum=diagonalSum(a,n);
System.out.println("Sum of diagonal elements is:"+sum);}}
```

```
static int diagonalSum(int a[][],int n)
{
int sum=0;
for(int i=0;i<n;i++)
for(int j=0;j<n;j++)
{
if(i==j)
sum+=a[i][j];
}
return sum;
}
}
```

Matrix multiplication

```
for(i=0;i<3;i++)
{       for(j=0;j<3;j++)
        {
                c[i][j]=0;
                for(k=0;k<3;k++)
                        c[i][j]+=a[i][k]*b[k][j];
                System.out.println(c[i][j]+” ”);
        }
        System.out.println();
}
```

Strings

- An object that represents sequence of char values

Creating string:

String msg=“Hello”;

or

String msg=new String(“Hello”);

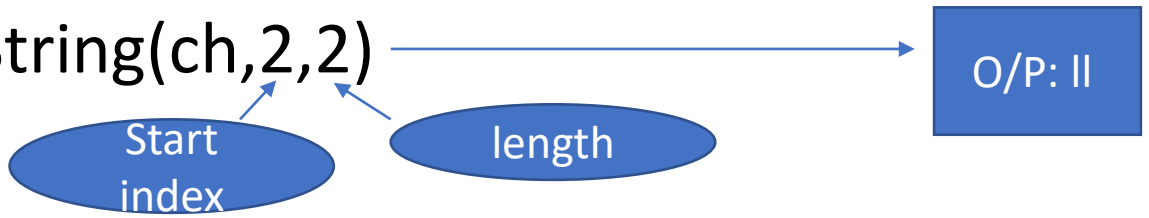
or

Char[] ch={‘h’,’e’,’l’,’l’,’o’}

String msg=new String(ch);

or

String msg=new String(ch,2,2)



String class methods

- **length()**

String txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

System.out.println("The length of the txt string is: " +txt.length());

o/p: The length of the txt string is: 26

- **concat()**

String first=“Patrick ”

String last=“Wilson”

System.out.println(first.concat(last));

o/p: Patrick Wilson

Note: + operator can also be used between strings to concatenate

- **toUpperCase() & toLowerCase()**

Converts all characters to a specified case.

```
String txt = "Hello World";
System.out.println(txt.toUpperCase());   // Outputs "HELLO WORLD"
System.out.println(txt.toLowerCase());   // Outputs "hello world"
```

- **indexOf()**

returns the **index** (the position) of the first occurrence of a specified text in a string.

```
String txt = "Please locate where 'locate' occurs!";
System.out.println(txt.indexOf("locate"));
// Outputs 7
```

- **equals() & equalsIgnoreCase()**

equals() - Case sensitive

eg:

String s1=“abc”, s2=“ABC”, s3=“def”, s4=“abc”;

System.out.println(s1.equals(s2));

System.out.println(s1.equals(s3));

System.out.println(s1.equals(s4));

System.out.println(s1.equalsIgnoreCase(s2));

o/p:

false

false

true

true

- **charAt()**

```
char ch;
String s1="Jack";
ch=s1.charAt(3);
System.out.println(ch);
```

o/p:
K

- **Substring()**

```
String s1="Hello World";
s2=s1.substring(0,4);
System.out.println(s2);
```

o/p:
Hello

- **replace()**

```
String s1="circle"
String s2=new String();
s2=s1.replace('c','t');
System.out.println(s2);
```

o/p:
tirtle

- **trim()**

```
String s1="    Hello World   ";
s2=s1.trim();
System.out.println(s2);
```

o/p:
Hello World

- **compareTo()**

s1.compareTo(s2);

if s1 > s2, it returns positive number

if s1 < s2, it returns negative number

if s1 == s2, it returns 0

eg:

String s1=“Hello”;

String s2=“Hemlo”

System.out.println(s1.compareTo(s2));

o/p:

-1

because "l" is 1 times lower than "m"

- **startsWith()**

boolean start= s1.startsWith(s2);

Returns true if a string starts with the substring s2

- **endsWith()**

boolean start= s1.endsWith(s2);

Returns true if a string ends with the substring s2

To check whether entered string is palindrome or not

```
import java.io.*;
public class Palindrome
{
static boolean isPalindrome(String str)
{
int i = 0, j = str.length() - 1;
while (i < j)
{
if(str.charAt(i) != str.charAt(j))
 return false;
 i++;
 j--;
 }
return true;
}
public static void main(String[] args)throws IOException
{
String str;
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the string");
str=br.readLine();
str=str.toLowerCase();
if(isPalindrome(str))
    System.out.print("Palindrome");
else
     System.out.print("Not a palindrome");
  }
}
```

To count vowels & consonants in a string

```
public class CountVowelConsonant
{
public static void main(String[] args)
{
        int vCount = 0, cCount = 0;
        String str = "This is a simple sentence";
        str = str.toLowerCase();
        for(int i = 0; i < str.length(); i++)
         {
            //Checks whether a character is a vowel
            if(str.charAt(i) == 'a' || str.charAt(i) == 'e'
|| str.charAt(i) == 'i' || str.charAt(i) ==          'o'
|| str.charAt(i) == 'u')
                        vCount++;
else if(str.charAt(i) >= 'a' && str.charAt(i)<='z')
          {
              cCount++;
           }
        }
System.out.println("Number of vowels: " + vCount);
System.out.println("Number of consonants: " + cCount);
    }
}
```

Sort the string alphabetically

```
import java.io.*;
public class StringOrdering
{
   public static void main(String[] args)throws IOException
   {
      int count;
      String temp;
      BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
      System.out.print("Enter number of strings you would
like to enter:");
      count = Integer.parseInt(br.readLine());
      String str[] = new String[count];
      System.out.println("Enter the Strings one by one:");
      for(int i = 0; i < count; i++)
         str[i] = br.readLine();
for (int i = 0; i < count; i++)
      {
         for (int j = i + 1; j < count; j++)
       {
            if (str[i].compareTo(str[j])>0)
            {
               temp = str[i];
               str[i] = str[j];
               str[j] = temp;
            }
         }
      } System.out.print("Strings in Sorted Order:");
      for (int i = 0; i <= count - 1; i++)
         System.out.print(str[i] + ", ");
   }}
```

Accept string upto 15 characters. Display elements of the string using their element no.

```
public class StringToCharArray
{
	public static void main(String args[])
	{
		String s1="hello !";
		char[] ch=s1.toCharArray();
		for(int i=0;i<ch.length;i++)
		{
			System.out.println("Character "+ch[i]+" is at position no. "+i);
		}
	}
}
```

Java Character Class

- Wraps the value of all the primitive type char into an object

Syntax:

Character ch = new Character('a’);

- If we pass a primitive char into a method that expects an object, the compiler automatically converts the char to a Character class object.
- Character class is immutable like String class i.e once it’s object is created, it **cannot** be changed.

Method	Description
`isUpperCase()`	Tests if character is uppercase
`toUpperCase()`	Returns the uppercase equivalent of the argument; no change is made if the argument is not a lowercase letter
`isLowerCase()`	Tests if character is lowercase
`toLowerCase()`	Returns the lowercase equivalent of the argument; no change is made if the argument is not an uppercase letter
`isDigit()`	Returns `true` if the argument is a digit (0–9) and `false` otherwise
`isLetter()`	Returns `true` if the argument is a letter and `false` otherwise
`isLetterOrDigit()`	Returns `true` if the argument is a letter or digit and `false` otherwise
`isWhitespace()`	Returns `true` if the argument is whitespace and `false` otherwise; this includes the space, tab, newline, carriage return, and form feed

Examples

```
public class Test {
 public static void main(String args[]) {
System.out.println(Character.toUpperCase('c'));
}
}
```

o/p:

C

```
public class Test {
  public static void main(String args[]) {
      System.out.println(Character.isLetter('c'));
      System.out.println(Character.isLetter('5'));
   }}
```

o/p:

True

false

```
public class Test {

   public static void main(String args[]) {
     System.out.println(Character.isWhitespace('c'));
     System.out.println(Character.isWhitespace(' '));
     System.out.println(Character.isWhitespace('\n'));
     System.out.println(Character.isWhitespace('\t'));
   }
}
```

o/p:

False

True

True

true

StringBuffer class

- StringBuffer class is used to create mutable (modifiable) string.
- It provides much of the functionality of strings.
- String represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences.
- Meaning, objects of class StringBuffer can be modified even after they have been created
- Eg: Inserting characters in the middle or at the end of objects
- Defined in java.lang package

StringBuffer constructors

- **StringBuffer( ):** It reserves room for 16 characters without reallocation.

```
StringBuffer s=new StringBuffer();
```

- **StringBuffer(int size):** It accepts an integer argument that explicitly sets the size of the buffer.

```
StringBuffer s=new StringBuffer(20);
```

- **StringBuffer(String str):** It accepts a **String** argument that sets the initial contents of the StringBuffer object and reserves room for 16 more characters without reallocation.

```
StringBuffer s=new StringBuffer(“OOP Class");
```

StringBuffer methods

1. **length():** object length
2. **Capacity():** total allocated capacity

Example:

```
import java.io.*;
class LengthCapacity
{
public static void main(String args[])
{StringBuffer s1 = new StringBuffer();
StringBuffer s2=new StringBuffer(20);
StringBuffer s3=new StringBuffer(“OOP Class");
System.out.println("Length of string s1 is: ”+s1.length()+” and capacity is:”+s1.capacity() );
System.out.println("Length of string s2 is: ”+s2.length()+” and capacity is:”+s2.capacity() );
System.out.println("Length of string s3 is: ”+s3.length()+” and capacity is:”+s3.capacity() ); }}
```

3. **append()**

eg:

StringBuffer sb=new StringBuffer(“Hello”);

sb.append(“ World”);

o/p: Hello World

4. **insert(int index, String str)**

eg:

sb.insert(sb.length(),” 2020”)

o/p: Hello World 2020

5. **reverse()**

eg: sb.reverse()

o/p: 0202 dlrow olleh

6. **delete(int startIndex, int endIndex)**

eg: sb= Hello World

sb.delete(5,10)

o/p: Hello

7. **deleteCharAt(int index)**

eg: sb="Hello World!"

sb.deleteCharAt(11)

o/p: Hello World

8. **replace(int startIndex, int endIndex, String str)**

eg: sb="Hello World!"

sb.replace(6,10,"Earth")

o/p: Hello Earth!

9. toString()

- Converts StringBuffer object to String object
- We can use string methods if required

sb= sb.toString();

```
import java.io.*;
class PalindromeBuffer
{
public static void main(String args[])throws IOException
{
		BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
		System.out.println("Enter the String");
		String str = br.readLine();
		String temp=str;
		StringBuffer obj=new StringBuffer(str);
		obj.reverse();
		str=obj.toString();
		if(temp.equalsIgnoreCase(str))
			System.out.println("Palindrome");
		else
			System.out.println("Not a palindrome");
}
}
```

Vectors

- Vector is like the dynamic array which can grow or shrink its size
- Defined in java.util package
- It can vary in size
- It can hold objects of any type & number

Syntax:

Vector<type> vec=new Vector<type>();

- We can only store objects (cannot store simple data type directly)
- We can convert simple types to objects using wrapper classes

Vector constructors

Sr.No.	Constructor & Description
1	**Vector( )** This constructor creates a default vector, which has an initial size of 10.
2	**Vector(int size)** This constructor accepts an argument that equals to the required size, and creates a vector whose initial capacity is specified by size.
3	**Vector(int size, int incr)** This constructor creates a vector whose initial capacity is specified by size and whose increment is specified by incr. The increment specifies the number of elements to allocate each time that a vector is resized upward.

Vector methods

Void addElement (Object o)	Adds object to end of vector, increases vector size by one
int capacity()	Returns capacity of vector
Object elementAT (int i)	Returns object at index i
int indexOf (Object o)	Finds first occurrence of object; uses equals method
void insertElementAt(Object o, int i)	Inserts object at index i
boolean isEmpty()	Returns true if vector has no objects, false otherwise
void removeElementAt(int i)	Deletes object at index i
void setElementAt(Object o, int i)	Sets vector element at index I to be the specified object
int size()	Returns size of vector

Vector v =new Vector(5,3);

1. v.size() //returns 0
2. v.capacity(); //returns 5
3. v.addElement(Integer.valueOf(4))
4. v.addElement(Integer.valueOf(9));
5. v.addElement(Integer.valueOf(10));
6. v.capacity() //returns 5
7. v.addElement(Double.valueOf(2.1));
8. v.addElement(Double.valueOf(3.1));
9. v.addElement(Double.valueOf(4.1));
10. v.capacity(); //returns 8

Example

```
import java.util.*;
import java.io.*;
class CreateVector {
      public static void main(String[] arg)
   {
       // create default vector
        Vector<Object> v1 = new Vector<Object>();
          // Add elements using add() method
        v1.add(Integer.valueOf(1));
        v1.add(2);
        v1.add("Java");
        v1.add(3);
// print the vector to the console
 System.out.println("Vector v1 is " + v1);
 System.out.println("Vector v1 size is " + v1.size());
 System.out.println("Vector v1 capacity is " + v1.capacity());

    }
}
```

```
import java.util.*;
public class AnimalsVector {
    public static void main(String args[]) {
      //Create an empty vector with initial capacity 4
      Vector<String> vec = new Vector<String>(4);
      //Adding elements to a vector
      vec.add("Tiger");
      vec.add("Lion");
      vec.add("Dog");
      vec.add("Elephant");
      //Check size and capacity
      System.out.println("Size is: "+vec.size());
      System.out.println("Default capacity is: "+vec.capacity());
      //Display Vector elements
      System.out.println("Vector element is: "+vec);
      vec.addElement("Rat");
      vec.addElement("Cat");
      vec.addElement("Deer");
      //Again check size and capacity after two insertions
      System.out.println("Size after addition: "+vec.size());
      System.out.println("Capacity after addition is: "+vec.capacity());
      //Display Vector elements again
      System.out.println("Elements are: "+vec);
      //Checking if Tiger is present or not in this vector
       if(vec.contains("Tiger"))
       {
         System.out.println("Tiger is present at the index " +vec.indexOf("Tiger"));
       }
       else
       {
         System.out.println("Tiger is not present in the list.");
       }
       //Get the first element
      System.out.println("The first animal of the vector is = "+vec.firstElement());
      //Get the last element
      System.out.println("The last animal of the vector is = "+vec.lastElement());
    }
}
```

```
import java.util.*;
import java.io.*;
public class ShoppingVector {
public static void main(String args[])throws IOException {
Vector<String> vec = new Vector<String>(5);
int i,choice;
 String str;
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Enter 5 items");
for(i=0;i<5;i++)
{
str=br.readLine();
vec.addElement(str);
}
 do
 {
 System.out.println("1. Add item\n2. Delete item \n3. Display items \n4.
Exit");
 choice=Integer.parseInt(br.readLine());
 switch(choice)  {
case 1:
System.out.println("Enter name of item to be added:");
str=br.readLine();
vec.addElement(str);
break;
case 2:
System.out.println("Enter name of item to be removed:");
str=br.readLine();
vec.removeElement(str);
break;
case 3:
System.out.println("Current List");
for(i=0;i<vec.size();i++)
System.out.println(vec.elementAt(i));
break;
case 4:
break;
}
}while(choice!=4);
}
}
```