(JAVA PROGRAMMING - 22412)

Unit II: Derived syntactical constructs in JAVA

(Weightage - 18 marks)

TRANSAL SAVE (SY-COMPS)

2.1 Constructors and method, types of constructors, nesting of methods, arguement passing the 'this' keyword, command line arguements, varargs : variable length arguements, garbage collection, finalize() method, the object class.

Questions

Q Give use of garbage collection in java (2m) .. S-23

Q Explain Constructor with its type. Give example of parameterized Constructor. (6M) ... S-23, S-22

Q Explain constructor with suitable example (2M) .. (W-22)

Q State use of finalize() method with syntax. (2M) (W-19) (S-19)

Constructor

* A constructor is a special member function which is used to allocate memory space and values to data members of that object.
* It is called constructor because it constructs the value of data member of class.
* Constructor is declared in side class
* A constructor have the same name as class itself.
* A constructor initializes an object immediately upon creation.
* Constructor is automatically called when an object is created.

Syntax

```
Constructor name ([arguments])
{
// body
}
```

Rules

1. Same name as class.
2. Constructor must have no explicit return type
3. It can be public, private or protected.
4. There can be different signature multiple constructors.

Types of Constructors

- Default constructors
- parameterized constructors

Default Constructor

- Default Constructor is also known as Empty constructor because it has no arguement.
- The constructor which does not accept any arguement is called default constructor.
- A Default constructor is used to initialise all the objects of class with same values.
- A default constructor can be called directly when an object of an class is created.
- If no constructor is there, compiler will create default constructor.

Syntax

```
class class_name
{
    public: //data members;
    class_name  //constructor
    {
    //body
    }
}
```

Example:

```
public class student
{
  string firstname, lastname;
   int age;
  public student()     //constructor
  {
    firstname = "ABC";
    lastname = "PQR";
     age = 100;
  }
public static void main(string args[])
  {
    student m = new student();
    System.out.println("Age : "+ m.student m.age);
  }
}
```

Parameterized Constructor

* A constructor that has parameters or arguements is known as parameterized constructor
* Sometime it is essential to initalize the various data elements of different object with different values when they are created.
* The constructor which accept any number of formal parameter are used for intalize the object are called parameterized constructor
* Using parameterized constructor, it is possible to inicatize object with different set of values at the time of their creation.
* These different set of values initialized to object must be passed as arrugement when constructor is invoked.

Syntax

```
class class_name
  {
    class_name (para1, para2)

    };
    class_name(constructor(parameter1, par2)

  }
```

Example

```
public class Edureka
{
string student_name;
int    studentAge;
Edureka (string name, int x) // constructor
   {
     student_name = name;
     studentAge = x;
   }
void display ( ) {
 System.out.println (studentname + " " +studentAge);
}
public static void main (string args[])
   {
    edureka object = new Edureka ("manas", 17);
     object.display ();

  }

}
```

Nesting of methods

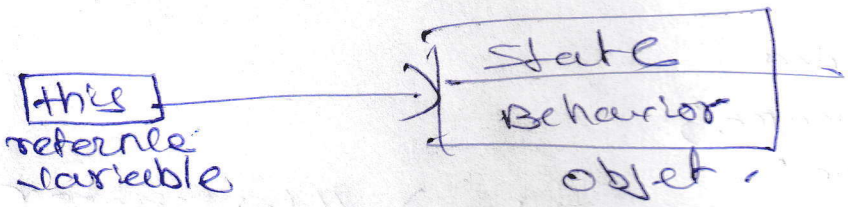
* If method in java calls a method in same class it is called nesting of methods.
* When a method calls the method in same class dot(.) operator is not needed.
* A method can call more than one method in same class
* Sucessive calls can be made means 1st can call 2nd and 2nd can 3rd and so on...

'this' keyword

* Many time it is necessary to refer to its own object in method or a constructor. To allow this JAVA defines the 'this' keyword.
* The 'this' is used inside the method or constructor to refer its own object.
* That is, 'this' is always a reference to object of current class type.

Use

* this can passed as an arguement in method call
* this() can be used to invoke current class constructor
* It can be used to invoke current class method.
* It can be used to refer current class instance variable.




```
class Box
{
    int height, depth, length;
    Box(int height, int depth, int length)
    {
        this.height = height;
        this.depth = depth;
        this.length = length;
    }
}
```

Command Line Arguments

* Sometimes, we will want to pass information into a program when we run it. This accomplished by passing command-line arguements to main().
* A command-line arguement is information that directly follows the program's name on command line when it is executed. To access the command line arguements inside a JAVA program is quite easy - they are stored as string in string array passed to main().

* Any number of arguements can be passed to java program through commandline.
* while running the program anythings written after the name of class are command line arguements.
* If more than one arguements are present they are seperated by spaces.

java Filename arg1, arg2, arg3 ..... ,

Program

```
class command
{
    public static void main(String args[])
    {
        System.out.println("The first arguement: "
                           +args[0]);
        System.out.println("The 2nd arguement: "
                           +args[1]);
    }
}
```

O/P

The first arguement: Java.
The second arguent: Programming

Varargs: Variable Length Arguements

The varrags allows the method to accept zero or multiple arugements. Before varargs either we use overloaded method or take an array as the method parameter but it was not considered good because it leads to maintalence problem. If we don't know how many arrugement we will have to pass in the method, varargs is better approach.

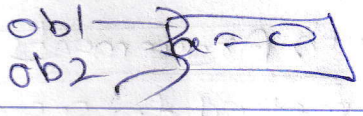
Garbage Collection

* In Java, garbage means unreferenced objects.
* when the object is not needed any more, the space occupied by such objects can be collected and used for later reallocation.
* JAVA performs release of memory occupied by unused objects automatically, it is called garbage collection.
* In other words "garbage collection is proces of reclaiming the runtime unused memory automatically".
* The garbage collector, or just collector, attempts to redalm garbage, or memory occupied by objects that are no longer in used by program.

```
class x
{
  int a;
  x()
  {
    a=0;
  }
}
class Y
{
public static void main (String args[])
  {
    x ob1 = new x();
    x ob2 = new x();
     ob1 = ob2;
  }
}
```

After execution

a=0



Finalize() method

- The finalize() method in java is called by the garbage collector before an object is garbage collected.
- This method can be used to perform any necessary clean up before the object is destroyed, such as releasing resources or detaching event listeners.
- To add a finalizer to a class, you define simply finalize() method.
- Syntax

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

2.2 Visibility Control : Public, Private, Protected, default friendly private protected access

Questions....

* Enlist any two access specifier with syntax. (2m)
* Explain any four visibility controls in JAVA. (4m)
* Explain the four access specifier in JAVA

Visibility Control

* Visibility modifiers are also known as access modifers or access specifiers.
* Access modifiers determine the accessibility of members of a class.
* An access specifier is a keyword that represents how to access a member of a class.
* Java provides a number of access modifiers to set access levels for classes, variables, methods and constructors.
* The four access levels are :

private : visible to the class only. private members of a class are not available outside class.

public : visible to the world. public members of class are available anywhere outside the class also inside.

protected : visible to package and all subclasses. protected members are available outside the class.

default : visible to package. If no access specifier is used then default specifier is used by java compiler. Default members are available outside the class. No modifiers are needed.

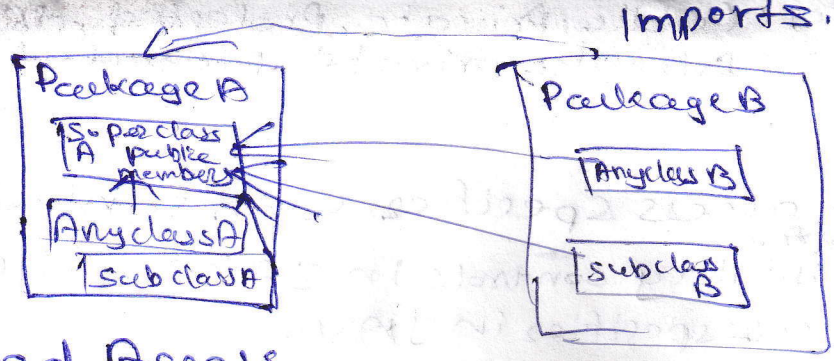
Public Access

* If we declare any variable and method as 'public' it will be accessible to all entire class and visible to all ~~other the class~~ the classes outside the class.
* It can be access outside package also.
* A variable or method declared as public has the widest possible visibility and accessible everywhere.

For example,

```
public int val;
public void display();
```

* Example. The public members of superclass A can be accessed from any class and subclass ~~from~~ the same package and any class and subclass from another package. So variable or method declared as public has widest possible visibility and accessible anywhere.



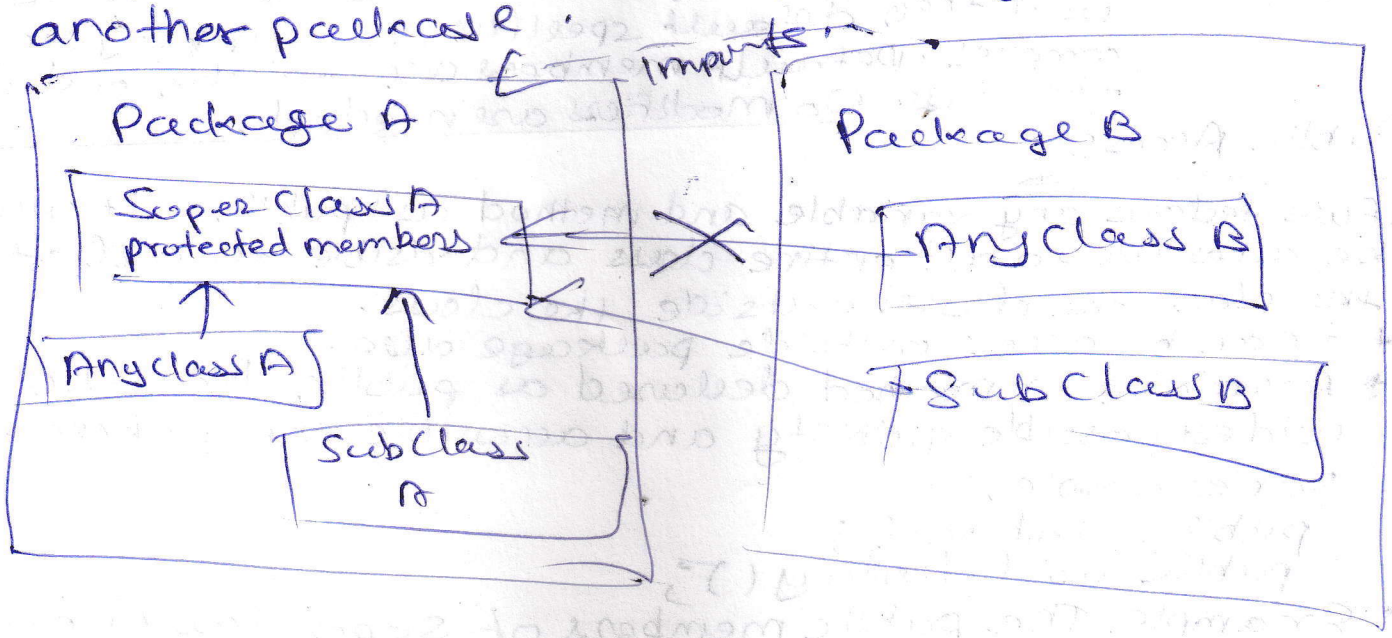
Protected Access

* The visibility level of protected field lies in between the public access and friendly access.
* A protected member is accessible in all classes in the package containing its class and by and by all subclass of its class in any package where the class is visible.
* In other words, non-subclasses in other packages cannot access protected members from other packages. Means visibility control is strongly related to inheritance.

Forexample,

```
protected int val;
protect void display();
```

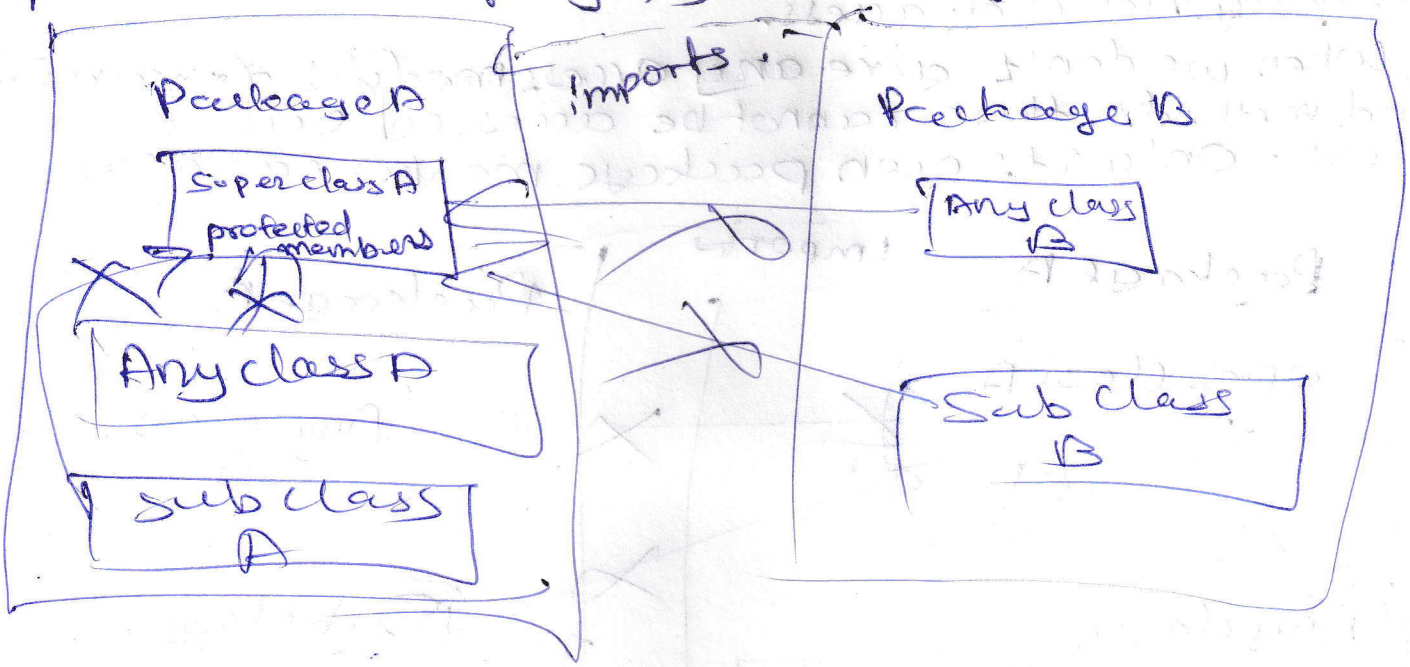
* Forexample The protected members from super class A can be accessed from subclass and other class from same package as well as sub class from another package.
But it cannot accessed from Any class from another package.



Private Access

* Private fields have highest degree of protection.
* This is most restrictive than all other visibility controls, Private members are not accessible from any other classes.
* They can not be accessed even from any subclass or any class from same package.
* In order to declare a member as private in front of member as.

```
private int val;
private void display();
```

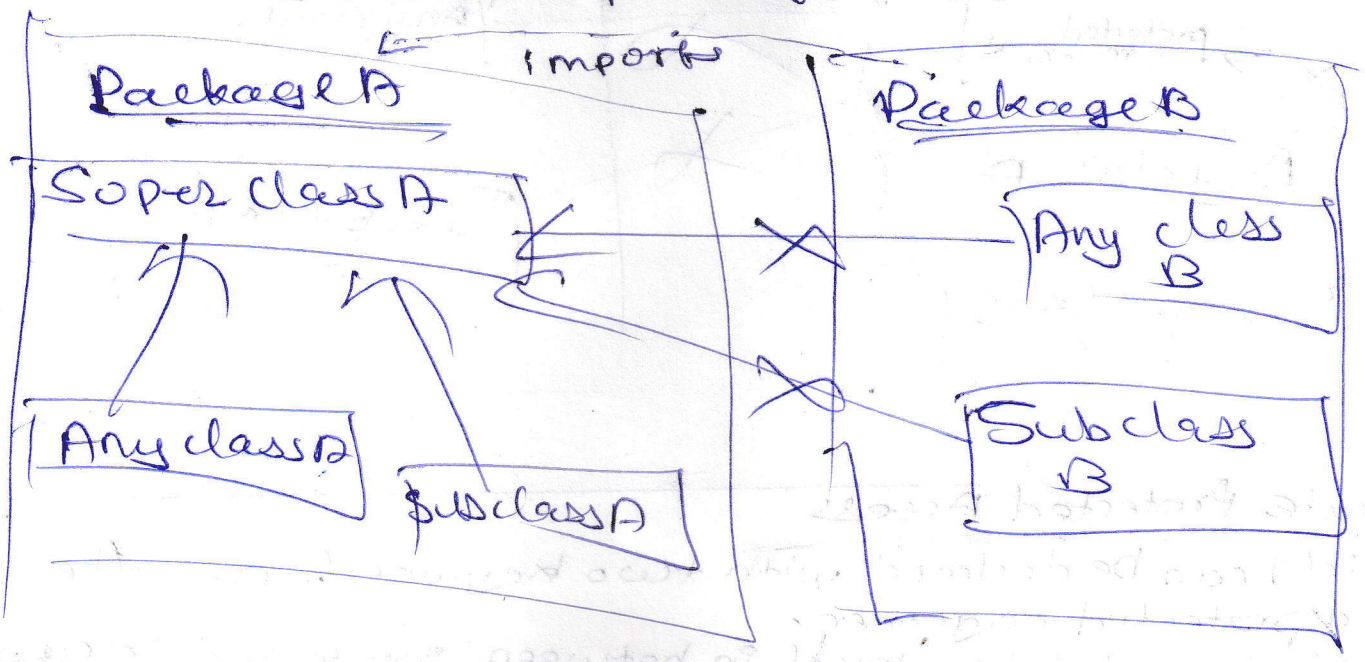


Private Protected Access

* A field can be declared with two keywords private and protected together.
* This gives visibility level in between "protected" access and "private" access.
* This modifier makes the field visible in all subclass regardless of what package they are in.
* Remember, these fields are not accessible by other classes in same package

Default Access

* The Default Access can also be called as package access or friendly Access.
* When no member accessibility modifier is specified, the member is only accessible inside its own package only.
* Even if class is visible in another package, the member is not accessible there.
* Default member accessibility is more restrictive than the protected members accessibility.
* When no access modifier is specified, the members default to a limited version of public accessibility known as "friendly" level of access.
* When we don't give any access modifiers to variables and methods, these cannot be accessed outside the class. Only its own package members can access it.



2.3 : Array and strings: Types of arrray, creating an array, strings, string class, and string buffer. vectors, wrapper classes, enumerated types.

Questions

Q. Differiate between string and string buffer class. (4m)

Q. Difference between vectors and array (4m)

Q. Explain any four methods of vector class with example. (4m)

Q. Compare array and vector. Explain elementAT() and addElement() methods

Q. Write JAVA 1-D array Bubble sort program. (4m)

Q. Write a program to check whether the string provided by the user in palindrome or not. (6m)

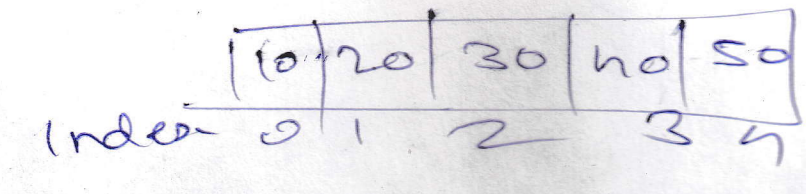
Q. Explain any four methods of string class. (4m)

Q. List any four methods of string class and state use of each. (4m)

Q. Name the wrapper class methods for following. (2m)
i) To convert string objects to primitive int.
ii) To convert primitive int to string objects

Array

* An array is a collection of similiar type of elements in which has contigos memory location
* Java array is object which contains elements of similiar data type.
* we can store only a fixed set of elements in Java array.
* Array in Java is index-based, the first element of array is stored at 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++.



Creating of Array

type array_name [];

type [] array_name;

Memory

array_name = new type[size];

Types of Array

- one / Single Dimensional Array. ⇒ 
- Two / multidimensional Array ⇒ matrices.

Array of object

obj name [ ] = new classname ( ) ;

Question ⇒

Write a program to copy all elements of one array into another array. (6M)

⇒

```
import java.util.Scanner;
public class ArrayCopy
{
  public static void main (String args[])
    {
     Scanner scanner = new Scanner (System.in);
     System.out.println ("Enter size of array = ");
      int size = scanner.nextInt();
   int[] sourceArray = new int[size];
    int[] destinationArray = new int[size];
System.out.println("Enter elements for source Array");
    for( int i=0; i<size ; i++)
        {
        System.out.println("Enter element
        sourceArray[i] = scanner.nextInt();
        }
    //logic for copying
    for(int i=0; i<size ; i++)
      {
      destinationArray[i]=sourceArray[i];
      }
   System.out.println("Elements in destination array");
     for( int i=0; i<size; i++)
       {
       System.out.print(destinationArray[i]+" ");
       }
}
}
```

Question

Java program to sort 1-d array in ascending order using bubble sort. (um)

⇒

```
import java.util.Scanner;
public class BubbleSort
{
public static void main(String args[])
{
Scanner scanner = new Scanner(System.in);
System.out.println("Enter size of Array");
int n = scanner.nextInt();
int [] a = new int[n];
System.out.println("enter elements of array");
for(int i=0; i<n; ;++)
{
a[i] = scanner.nextInt();
}
//Perform BubbleSort
for(int i=0; i<n-1; ;++)
{
for(int j=0 ;j<n-i-1; ;++)
{
if(a[j] > a[j+1])
{
int temp = a[j];
a[j] = a[j+1];
a[j+1] = temp;
}
}
}
System.out.println("sorted array");
for(int i=0; i<n; ;++)
{
System.out.println(a[i]+" ");
}
}
}
```

Question

Program for addition of 3x3 matrices.

```
Class matrices
{ public static void main (String args[])
  {
    int x[][] = { {8,5,6},
                  {1,2,1},
                  {0,8,7} };
    int y[][] = { {4,3,2},
                  {3,6,4},
                  {0,0,0} };
  System.out.println("first matrix");
   for(i=0;i<3;i++)
     {for(j=0;j<3;j++)
       {
     }}System.out.print("\n"+x[i][j]);
  }
 System.out.println("second matrix");
for(int j=0;i<3;i++)
   {for(int j=0;j<3;j++)
     {
      S.O.P("\n"+y[][]);
   }}
System.out.println("Addition");
 for(int i=0;i<3;i++)
   {
    for(int j=0;j<3;i++)
     {
    System.out.prnt(" "+x[i][j]+y[i][j]);
     }
   }
}
```

Strings

String is collection of characters.
In Java, the string can be defined using two commonly used methods
- Using string class
- Using string Buffer class

String class

Syntax ⇒ string string_variables ;

Example,

```
class stringDemo
{
    public static void main(String args[])
    {
    String s= "Welcome123";
    S.O.P(" "+s);
    }
}
```

Methods of string class

Question ⇒ Explain any four methods of string class.

(1) length()
This method is used to find total no. of characters from string.
Syntax : int length()
Example : ~~int len =~~ s.length();

(2) concat()
This is used to join two strings.
Syntax : string concat(string str)
Example
```
string s = "First";
String t= "second";
s.concat(t);
```

(3) s1.equal(s2)
If s1 & s2 are equal then return true.

(4) s1.charAt(position)
Return the character present at index position.

(5) s1.compareto(s2)
s1 < s2 ⇒ positive
s1 > s2 ⇒ negative

(8) trim()
(9) replace()
s1 = s2 ⇒ zero.

(6) toLowercase()
(7) toUppercase()

StringBuffer

The stringBuffer is 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 more components to existing string or modify the existing string but in case of String class once the string is defined then it remains fix.

following are somesimple methods used for StringBuffer

Name of Method	Description
append(String str)	Appends the string to buffer.
capacity()	It returns the capacity of string buffer.
charAt(int index)	It returns a specific character from the sequence which is specified by index
delete(int start, int end)	It delete the characters from string specified by the starting & ending index
insert(int offset, char ch)	It inserts character at position specified by offset.
length()	It returns length of string buffer.
setCharAt (int index, char ch)	The character specified by index the string buffer is set to ch.
setLength(int new_len)	It sets length of String buffer.
toString()	It converts the string representing data in string buffer.
replace(int star, int end, String str)	It replace the characters specified by new string.
reverse()	The character sequence is reversed.

Question:

Write a simple java program to find the reverse string and check whether the entered string is Palindrome or not. (6M)

```
import java.util.*;
class ReversePalindrome
{
    public static void main(String args[])
    {
        String s1;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter string");
        s1 = sc.nextLine();
        StringBuffer s2 = new StringBuffer(s1);
        s2.reverse();
        System.out.println("orginal string "+s1);
        System.out.println("reversed string "+s2);
        System.out.println("checking Palindrome");
        if(s1.equals(s2.toString()))
        {
            System.out.println(s1+" Palindrome");
        }
        else
        {
            System.out.println(s1+" not palindrome");
        }
    }
}
```

String class	String Buffer class
The length of string object is fixed.	The length of String Buffer can be increased.
The string object is immutable, that means we can not modify the string once created.	The String Buffer class is immutable.
It is slower in performance	It is faster in performance
It consumes more memory	It consumes less memory
Thread Safe	Not Thread Safe
It represents immutable character sequence	It represent mutable sequence of character.
slower during concatenation	faster during concatenation
Less efficient	more efficient

Wrapper classes

* Wrapper classes that allow primitive datatypes to accessed as objects.
* The wrapper class is one kind of wrapper around primitive datatype.
* The wrapper classes represent the primitive data types in instance of class.

Following table shows various primitive datatypes and co-responding wrapper classes

Primitive data type	Wrapper class
boolean	Boolean
byte	Byte
char	Char
double	Double
float	Float
int	Integer

Uses of wrapper classes

* They 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 retrive value of object as primitive datatype.

i) To convert string objects to primitive int.

```
String str = "5";
int value = Integer.parseInt(str);
```

ii) To convert primitive int to string objects.

```
Int value 5;
String str = Integer.toString
             (value);
```

Vector

* The vector is class in java that implements 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 can store integer, string, double or any of the data type elements together.

Advantage of vectors over array.

- Vectors contains elements of varying data type.
- The size of vector can be changed whenever required.
- It can store simple objects.

Vector created like this :-

```
Vector vectobj = new vector();
Vector vectobj = new vector(5);
```

Methods

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.

Vector

* The vector is class in java that implements 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 can store integer, string, doceble or any othe data type elements together.

Advantage of vectors over array.

- Vectors contains elements of varying datatype.
- The size of vector can be changed whenever required.
- It can store simple objects.

Vector created like this :-

```
Vector vectobj = new vector();
Vector vectobj = new vector(5);
```

Methods

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.

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());
}
}
```

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

```
import java.util.Scanner;
import java.util.Vector;
public class VectorOperations {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    Vector<Object> vector = new Vector<>();
// Adding integers, strings, and floats to the vector
    vector.add(5);         // Integer
    vector.add("Hello");   // String
    vector.add(3.14f);     // Float
    vector.add(10);        // Integer
    vector.add("World");   // String
    vector.add(2.718);     // Double (automatically converted to Float)

    // Displaying the vector
    System.out.println("Vector elements: " + vector);

    // Removing an element specified by the user
    System.out.print("Enter the index of the element to remove: ");
    int indexToRemove = scanner.nextInt();
   if (indexToRemove >= 0 && indexToRemove < vector.size()) {
      vector.remove(indexToRemove);
      System.out.println("Element at index " + indexToRemove + " removed.");
    } else {
      System.out.println("Invalid index. No element removed.");
    }

    // Displaying the updated vector
    System.out.println("Updated vector: " + vector); } }
```

Output:-

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

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