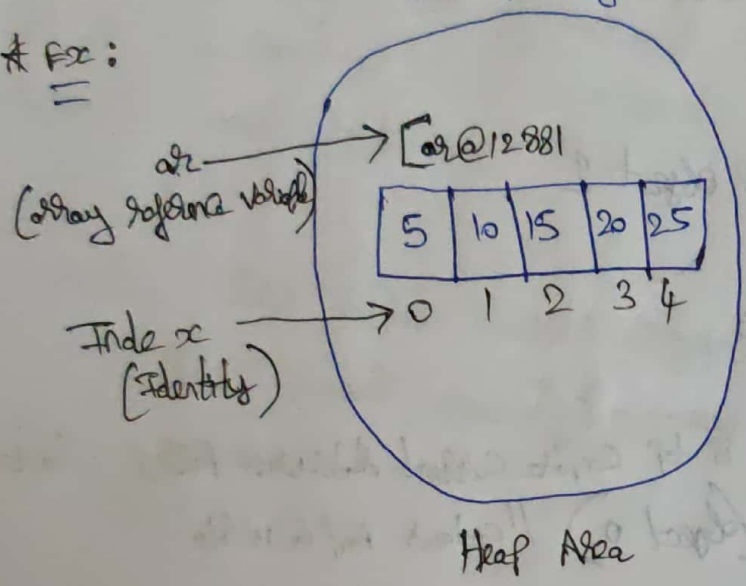
16/09/22

ARRAYS

* Array is a linear data Structure and it is a contigious (next to one another) Block of memory which is used to store homogeneous Data.

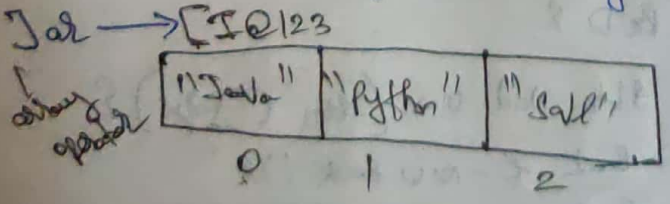
* Ex:



Characteristics of an Array

* It is a homogeneous collection of element.
* Size of the array is fixed, once, array is created we cannot able to Increase or decrease the size of the array dynamically.
* We can be able to access the elenat of an array with the help of index. The Index always starts from 0 and the the length of the array always starts from 1.

Ex:



length ⇒ 3

Index ⇒ 0 – 2

HOW TO CREATE AN ARRAY?

* In Java, we can create an array in two ways, (i.e)
  1. Dynamic Way
  2. Static Way

Syntax for Creating an Array Using Dynamic Way

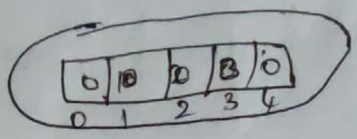
```
datatype[] VariableName = new datatype[size];
```

↓

Array Ref Variable

Ex: (Primitive datatype)

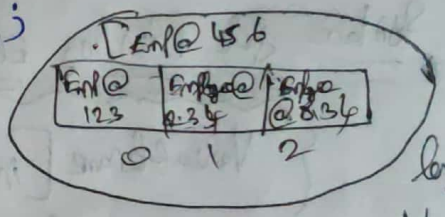
1. `int[] ar = new int[5];`



length => 5
Index => 0-4

(non-Primitive datatype)

2. `Employee[] er = new Employee[3];`



length => 3
Index => 0-2

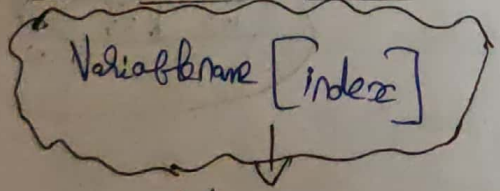
WAJP TO CREATE AN INTEGER ARRAY of SIZE 3 AND PRINT THE SIZE.

```
Public class A1 {
    Public static void main (String[] args) {
        int[] a = new int[3];
        S.o.println("first Array:" + a.length);
        double[] b = new double[8];
        S.o.println("Second Array:" + b.length);
    }
}
```

```
Public class A2 {
    Public Static Void main(String [] args) {
    double [] a = new double [4];
    S.o.Pln(a[0]);
    S.o.Pln (a[1]);
    S.o.Pln (a[2]);
    S:o.Pln (a[3]);
    S.o.Pln (a[-1]);   //RTE ATOBE
    S.o.Pln (a[5]);    //RTE ATOBE
    }
}
```

Note:

Syntax for Printing Elements

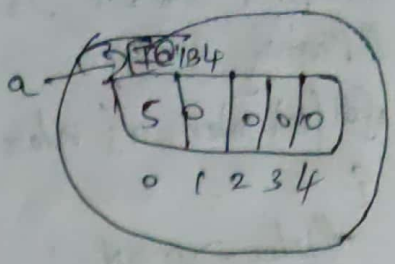


index = integer

WAJP TO CREATE AN ARRAY of SIZE 5 AND STORE THE ELEMENTS AND PRINT IT.

```
Public class A3 {
    Public Static Void main(String [] args) {
    int [] a = new int [5];
    //L->5    I->0..4
    a[0] = 20;
    a[1] = 30;
    a[2] = 40;
    a[3] = 50;
    a[4] = 60;
```

```
S.o.pln ("a[0]-->" + a[0]);
S.o.pln ("a[1]-->" + a[1]);
S.o.pln ("a[2]-->" + a[2]);
S.o.pln ("a[3]-->" + a[3]);
S.o.pln ("a[4]-->" + a[4]);
```



}
}

Assignment

1) WAJP TO CREATE STRING ARRAY OF SIZE5AND STORE THE ELEMENT AND PRINT USING FOR-Loop.

2) WAJP TO CREATE A ARRAY EMPLOYEE TYPE of SIZE 5 AND PRINT THE NAMES of THE EMPLOYEE.

3) WAJP TO CREATE AN ARRAY of FRUIT HAVING THE STATES TASTE AND PRICE. PRINT ALL THE ELEMENTS USING FOR Loop.

17/09/22

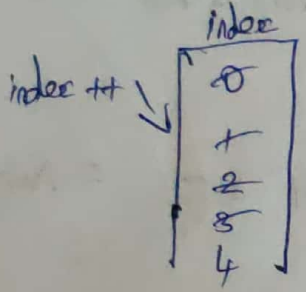
WAJP to create an integer array of size decided by the user and initialize the array by taking the input from the user.

```
import java.util.*;
Public class A4
{
    Static Scanner S=new Scanner(System.in);
    Public static void main (String [] args)
    {
        S.o.pln ("Enter the size of the array");
        int a[] = new int[S.nextInt()];
        // Initializing
        for(int index=0; index<a.length; index++)
        {
            //index -> 0 to N-1
            S.o.pln ("Enter the number");
            a[index] = S.nextInt();
```

```
}
S.o.Pln("****");
// Printing
for(int index=0; index<a.length ;index++)
{
  //index -> 0 -> N-1
  S.o.Pln("a["+index+"] -> "+a[index]);
}
}
```

Tracing part

length = 4 ; index = 3



0 < 4 -> True
a[0] = 2
1 < 4 -> True
a[1] = 3
2 < 4 -> True
a[2] = 6
3 < 4 -> True
a[3] = 9
4 < 4 -> false

O/P

a[0] => 2
a[1] -> 3
a[2] -> 6
a[3] -> 9

(**) WAP to copy one array element into an array..

```
import java.util.Scanner;
public class A5
{
  static Scanner s=new Scanner(System.in);
  public static void main(String [] args)
  {
    s.o.Pln("Enter the size of an array: ar");
    int [] ar = new int [s.nextInt()];
    int br[] = new int [ar.length];
    for(index=0; index < ar.length; index++)
int <  {
      s.o.Pln("enter the elements");
      ar[index] = ar[index];
    }
    s.o.Pln("*****");
```

```
for (int index=0; index<br.length; index++)
{
    S.o.Pln("br[" + index + "]-> " + br[index]);
}
}
}
```

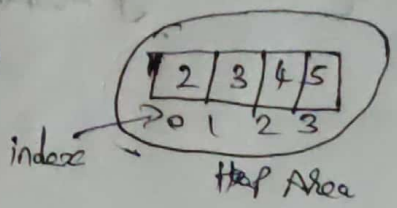
Static Way of Array Creation:

Syntax:

```
datatype[] variable = {data1, data2, data3, ....};
```

↑ PDT/NPDT

Ex: `int[] a = {2,3,4,5};`



Note:

* For dynamic way of array creation we can declare and initialize an array in a different line & Compile Time Successful.

```
Ex: int a[];  // declaration
    a = new int[4];  // CTS
```

* For static way of array creation we can't declare and initialize into different line. It is Compile Time Error. We have to declare & initialize in a same line.

```
Ex: int []b;  // declaration of an array
    b = {1,2,3,4};  // CTE.
    int []c = {2,4,5,6};  // CTS
```

WAP to create an array by using static keyword and print the elements.

```
Public class A6
{
    Public static void main(String [] args)
    {
        String s[] = {"SQL", "Java", "Testing", "JDBC"};
        // i) length ii) Print All the element
        S.o.Pln("The length is:" + s.length);
        for(index=0; index<s.length; index++)
        {
            S.o.Pln("The array element is:" + s[index]);
        }
    }
}
```

o/p

```
The length is: 4
The array element is: SQL
The array element is: Java
   "    "    "    : Testing
   "    "    "    : JDBC
```

WAP to print the array elements from the reverse order.

```
Public class A7
{
    Public static void main(String [] args)
    {
        double [] d = {1,2,3,4}; //L-4 ; index = 0-3
        for(int index = d.length-1; index>=0; index--)
        {
            //index = 3,2,1,0
            S.o.Pln("d[" + index + "] -> " + d[index]);
        }
    }
}
```

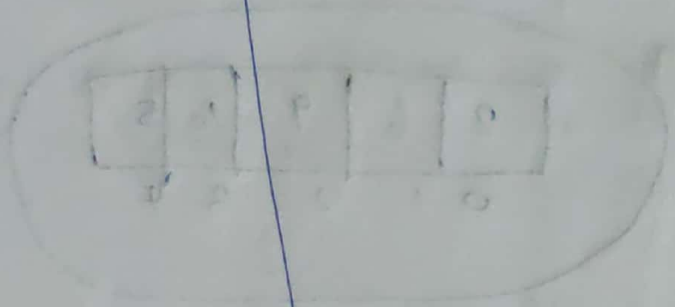
o/p

d[3] = 1

d[2] = 2

d[1] = 3

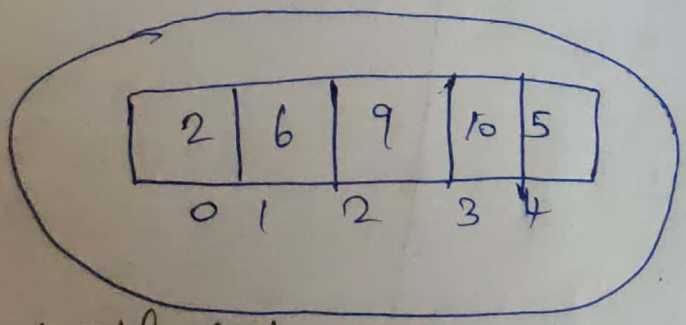
d[0] = 4



Reverse Accessing

```
public class A7 {
    public static void main (String [] args) {
        double [] d = {1, 2, 3, 4}; // L-4 Index -> 0-3
        for (int index = d.length - 1; index >= 0; index--) {
            // index = 3, 2, 1, 0
            S.o.Pln ("d[" + index + "] -->" + d[index]);
        }
    }
}
```

WAJP TO CREATE AN INTEGER ARRAY AND FIND SUM of THE ELEMENT AND ALSO FIND SUM of THE EVEN ELEMENT AND FIND PRODUCT of THE ELEMENT AND FIND AVERAGE of THE SUM of ELEMENT.



a.length = 5
index = 0 - 4

1. Total elevat sum -> 2+6+9+10+2
2. Total sum of even elevat -> 2+6+10
3. Product of elevat -> 2*6*9*10*5
4. Average

```
public class A8 {
    public static void main(String [] args) {
        int [] a = {2, 6, 9, 10, 5};
        int sum=0, e_sum =0, product=1, avg=0;
        for (int index = 0; index < a.length; index++) {
            //index --> 0, 1, 2, 3, 4
            sum += a[index];
            if (a[index] % 2 == 0)
```

e_sum+=a[index];
Product*=a[index];
}
avg=sum/a.length;
S.o.Pln("The Total sum:"+sum);
S.o.Pln("The Even Sum:"+e_sum);
S.o.Pln("The Total Product:"+product);
S.o.Pln("The Total Average:"+avg);
}

Assignment

1. WAJP TO CREATE CHARACTER ARRAY AND PRINT ITS ASCII VALUES.
2. WAJP TO CREATE 2 ARRAYS of DIFFERENT AND CREATE RESULTANT ARRAY WHICH INCLUDE FIRST ARRAY AND SECOND ARRAY.
3. WAJP TO FIND THE FIRST LARGEST ELEMENT IN THE GIVEN ARRAY.
&
FIRST SMALLEST ELEMENT
4. WAJP TO READ THE ELEMENT OR NUMBER FROM THE USER AND CHECK THE ELEMENT IS PRESENT OR NOT.

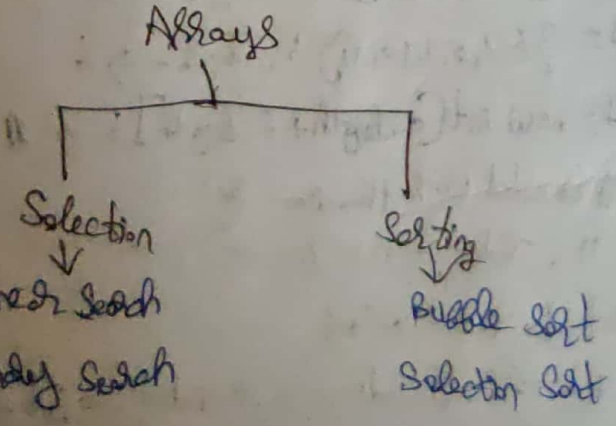
9/09/22

2. Public class A9 {
public static Void main (String [] args) {
int a[]={1,2,3,4,5}; // length -->5
int b[]={6,7,8,9,10,11}; // length -->6
int res[]=new int[a.length+b.length]; // 11
// one array created with the name res
// length--11, Index=0-10
// with respect to third array
for(int t_index=0, f_index=0, s_index=0; t_index<res.length; t_index++){
// t_index=0-10
if(t_index<a.length) // 5--5
{

```
        res[t_index] = a[f_index++];
    }
    else
    {
        res[t_index] = [$_index++];
    }
    S.o.pln("res[" + t_index + "] -> " + res[t_index]);
}
```

o/p

res[0] -> 1
res[1] -> 2
res[2] -> 3
res[3] -> 4
res[4] -> 5
res[5] -> 6
res[6] -> 7
res[7] -> 8
res[8] -> 9
res[9] -> 10
res[10] -> 11



LINEAR SEARCH

```
import java.util.Scanner;
//Linear Search
public class A1o{
    static String res="Not Present";
    static int key;
    static Scanner s=new Scanner(System.in);
    public static void main(String [] args){
        int a[]={4,2,3,6,9,10,15};
        S.o.Pln("Enter the Key Element");
        key=S.nextInt();
        S.o.Pln("The Element is :"+linearSearch(a));
    }
    public static String linearSearch(int a[])
    {
      for(int index=0;index<a.length;index++){
        if(a[index]==key){
            res="Present";
            break;
        }
      }
      return res;
}
```

WAJP to create an employee class having the data members employee id and the employee name and create and employee array and search the employee id to is present or not if it is present print employee details.

```
Public class Emp {
    //stubs
    String name;
    int emp_id;
    Emp()
    {

    }
    Emp(String name, int emp_id)
    {
        this.name = name;
        this.emp_id = emp_id;
    }
    @Override
    Public String toString()
    {
        return "name: " + name + "id: " + emp_id;
    }
}

import java.util.Scanner;
Public class EmpD {
    Static Scanner s = new Scanner(System.in);
    Public Static Void main(String[] args) {
        Emp[] e = {new Emp("Ram", 21), new Emp("Sita", 22),
                   new Emp("Bobby", 23)};
        S.o.Pln("Enter the Emp Id");
        int id = s.nextInt();
        for (int index = 0; index < e.length; index++) {
            //index -> 0-2
            if (id == e[index].emp_id)
            {
                //details
                S.o.Pln(e[index]); //toString
            }
        }
    }
}
```

BINARY SEARCH

```
import java.util.Scanner;
public class BS {
    static String res="Not Present";
    static int l, mid, h, key;
    static
```

* If we want to apply Binary Search algorithm, the element present in an array it should be sorted.
* We have to always note lower index value, middle index value and higher index value.
* To achieve Binary Search algorithm, we have to assume our key element is present in middle position.

Ex:

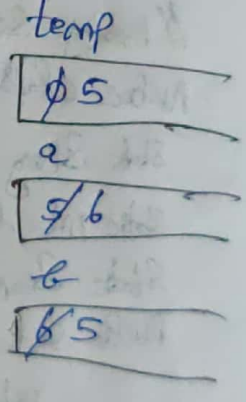
```
import java.util.Scanner;
// Binary Search
Public class BS {
 static String res="Not Present";
 static int l, mid, h, key;
 Static Scanner s=new Scanner(System.in);
 Public Static Void main(String [] args) {
        int [] a={1,2,3,4,5};
        l=0;
        h=a.length-1;
        mid=(l+h)/2;
        s.o.println("Enter the key element");
        key=s.nextInt();
        s.o.println(BinarySearch(a));
}
Public static String BinarySearch(int a[])
{  while(h>=l)
```

```
mid = (l+h)/2;
if (a[mid]==key)
{
    res="present";
    break;
}
else if (key > a[mid])
    l=mid+1;
else
    h=mid-1;
}
return res;
}
}
```

20/09/22

WAP for swapping the two numbers with the help of temporary variable.

```
int a=5;
int b=6;
int temp=0;
//swapping
temp=b;
b=a;
a=temp;
```




```
public class Swap {
    public static void main(String[] args) {
        int a=5, b=6;
        S.o.Pln("Before Swapping");
        S.o.Pln("A: ->"+a);
        S.o.Pln("B: ->"+b);
        S.o.Pln("*****");
        //swapping
        int temp=a;
        a=b;
        b=temp;
```

```
        S.o.Pln("After Swapping");
        S.o.Pln("A:=>"+a);
        S.o.Pln("B:=>"+b);
    }
}
```

BUBBLE SORT

Eg:

```
public class BubbleSort {
    public static void main(String [] args) {
        int [] a = {24, 23, 22, 21};
        //Length -=> 4
        //outer condition -=> (outer loop) - 3
        //Inner condition -- (Inner loop)
        int temp=0;
        for (int i=0; i<a.length; i++) {
            //0,1,2 (outer for loop)
            for (int j=0; j<a.length ; j++)
            {
                //j=0,1,2
                if (a[j] >a[j+1]) //true
                {
                    //Swapping
                    temp=a[j];
                    a[j]=a[j+1];
                    a[j+1]=temp;
                }
            }
        }

        for(int i=0; i<a.length ; i++) {
            S.o.Print(a[i]+" ");
        }
    }
}
```

o/p

21 22 23 24

SELECTION SORT

* In the selection sort algorithm, our intention is to find the smallest element index and swap that smallest element with the first element.
* Whose outer loop variable i is pointing.
* In selection sort, in each pass/cycle only one swapping will be there.

Ex:

```
public class SelectionSort {
    public static void main (String [] args){
        int index=0, temp=0;
        int a[] = {7,8,6,2,0,9,5};
        for (int i=0; i<a.length; i++) {
            index=i;
            for (int j=i+1; j<a.length; j++) {
                if (a[index] > a[j]) // true
                {
                    index = j;
                }
            }
            temp = a[index];
            a[index] = a[i];
            a[i] = temp;
            S.o.Print(a[i] + " ");
        }
    }
}
```

WAJP to find the first largest element present in an array.

```
public class Largest {
    public static void main (String [] args) {
    int [] a = {2, 8, 9, 16, 4, 3};
    int largest = a[0];
    for (int i=0; i < a.length ; i++) {
        //index --> 0-5
        if (a[i] > largest)
            largest = a[i];
    }
    S.o.pln ("Largest element is :" + largest);
    }
}
```

WAJP to find the first smallest element present in an array.

```
Public class Smallest {
    public static void main (String [] args) {
    int [] a = {2, 8, 9, 16, 4, 3};
    int smallest = a[0];
    for (int i=0; i < a.length ; i++) {

        if (a[i] < smallest)
            smallest = a[i];
    }
    S.o.pln ("Smallest element is :" + smallest);
    }
}
```

Assignment

1. WAP To find the nth largest element of an array. (** interview Que)
2. WAP To find the nth smallest element of an array.
3. WAP to find the frequency of duplicate element

21/09/22

String Class

* It is a class present in ~~java.util~~ java.lang Package.
* The declaration of String class in the lang package is given below,

public final class String extends Object implements Serializable, Comparable<String>, CharSequence

STRING

* String is a literal (data).
* It is a group of character that is enclosed within the double quote " ".
* It is a Non-Primitive data.
* In java, we can store a String by creating instance of the following classes.
  - → java.lang.String
  - → java.lang.StringBuffer
  - → java.lang.StringBuilder
* In java, whenever we create a String compiler implicitly create an instance for java.lang.String in String Pool Area / String Constant Pool (SCP).

STRING LITERAL

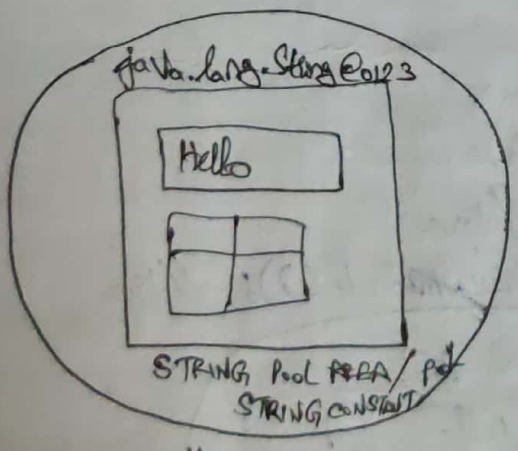
Anything enclosed within the double quotes " " in java is considered as String literal.

Characteristics of String Literal:

* When a String literals is used in java Program, an instance of java.lang.String class is created inside a String Pool.
* For the given String literal, if the instance of a String is already present, then new instance is not created instead the reference of a existing instance is given.

Ex: 1

```
class Demo
{
  public static void main (String [] args)
  {
    S.o.Pln ("Hello");  // String@0123
    S.o.Pln ("Hello");  // String@0123
  }
}
```



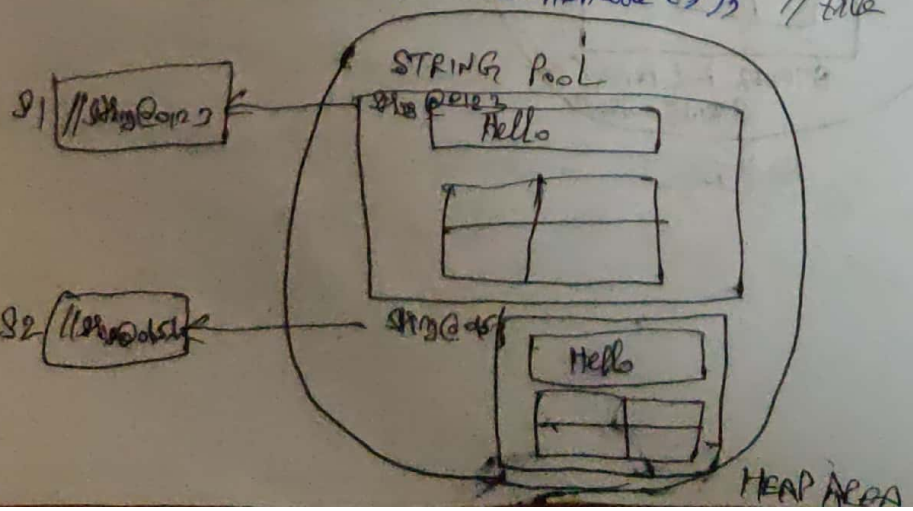
Heap Area

Ex 2:

```
class Demo
{
    public static void main(String [] args)
    {
        String s1,s2;
        s1="Hello";
        s2="Hello";
        S.o.Pln(s1);
        S.o.Pln(s2);
        S.o.Pln(s1==s2); // true
        S.o.Pln(s1.equals(s2)); // true
    }
}
```

Ex 3:

```
class Demo
{
    public static void main(String [] args)
    {
        String s1,s2;
        s1="Hello";
        s2=new String("Hello");
        S.o.Pln(s1);
        S.o.Pln(s2);
        S.o.Pln(s1==s2); //false
        S.o.Pln(s1.equals(s2)); //true
        S.o.Pln(s1.hashCode()==s2.hashCode()); //true
```



STRING CLASS

java.lang.String:

* String is a inbuilt class defined in a java.lang Package.
* It is a final class
* It inherits java.lang.object
* In a String class toString(), equals(), hashcode() methods of java.lang.object class are overridden.

It Implements:

* Comparable
* Serializable
* CharSequence

CONSTRUCTOR IN STRING CLASS

CONSTRUCTORS	DESCRIPTION
String()	Creates an empty String object
String (String literals)	Creates String object by initializing with String literals.
String (char[] ch)	Create String by converting character array into String
String (byte[] b)	Creates String by converting byte array into String
String (StringBuffer sb)	Creates String by converting StringBuffer to String.
String (StringBuilder sb)	Creates String by converting StringBuilder to String.

METHODS of STRING CLASS

IMPORTANT METHODS:

RETURN TYPE	METHOD NAME	DESCRIPTION
String	toUpperCase()	converts the specified String to Uppercase.
String	toLowerCase()	converts the specified String to LowerCase
String	concat(String s)	joins the specified Strings
String	trim()	Remove the space present before and after the String.
String	subString(int index)	extract a characters from a String object starts from specified index and ends at the end of a String.
String	SubString(int start, int end)	Extract a characters from a String starts from specified index and ends at end-1 index.
char	charAt(int index),	Returns character of the specified index from the String.
int	indexOf(char ch)	Return the index of the character specified if not return -1.
int	indexOf(char ch, int start_index)	Return the index of the character specified by searching from specified index if not return -1.
int	indexOf(CharSequence str)	Return the index of the specified String index if not return -1.
int	indexOf(CharSequence str, int start_index)	Return the index of the specified String by searching from specified index if not return -1.
int	lastIndexOf(char ch)	Return the index of the character which is occured at

		last in the original String.
int	length()	returns length of the String
boolean	equals(Object o)	compares states of two Strings
boolean	equalsIgnoreCase(String s)	compares two Strings by ignoring its case.
boolean	contains(String str)	returns true if specified String is present else it returns false.
boolean	isEmpty()	return true if String is empty else return false.
char[]	toCharArray(String str)	converts the specified String into character array.
String[]	split(String str)	Break the specified String into multiple String and returns String array.
byte[]	getBytes()	converts the specified String to byte value and returns byte array.

Characteristics of String

* Instance of String class is immutable in nature. (once the object is created then the state is not modified).
* If we try to manipulate (Modify) the state / data then new object is created and reference is given.

22/09/22

WAJP to convert char array into String.

```
Public class S2
{
    Public static void main (String [] args)
    {
        char [] ch = {'J', 'a', 'v', 'a'};
        // String -> char[]
        String res = new String (ch);
        S.o.Pln ("String -> " +res);
    }
}
```

Another way is,

```
Public class S2
{
    Public static void main (String [] args)
    {
        char [] ch = {'H', 'e', 'l', 'l', 'o'};
        String res = " ";
        for (int index=0 ; index < ch.length ; index++)
        {
            //index -> 0-4
            res = res + ch [index];
            S.o.Pln (res);
        }
    }
}
```

o/p

Hello

WAP to Print Each characters Present in a String.

```
public class S3
{
    public static void main(String[] args)
    {
        String s = "Java";
        for(int index = 0; index < s.length; index++)
        {
            //index -> 0-3
            ch = s.charAt(index);   //0, 1, 2, 3
            S.o.Pln(ch);
        }
    }
}
```

o/p

J
a
V
a

WAP to convert String into char array.

```
public class S4
{
    public static void main(String[] args)
    {
        String s = "Tabrez";   //index -> 0-5
        //char[] ch <- String
        char ch[] = new char[s.length()];   //length = 6
                                            index -> 0-5
        for(int index = s.length()-1; i=0; index>0; index--, i++)
        {
            //index -> 5-0
            //i -> 0-5
            ch[i] = s.charAt(index);
            S.o.Pln("ch["+i+"] -> "+ch[i]);
        }
    }
}
```

(or)

ch[0] - z
ch[1] - e
ch[2] - r
ch[3] - o
ch[4] - a
ch[5] - T

WAJP to Reverse a String and palindrome (or) not.

```
public class S5
{
    public static void main(String [] args)
    {
        String given = "Malayalam";
        String s = "Hello";
        String s1 = "Hello";
        String rev = "";  // Malayalam
        for(int i = given.length() - 1; i >= 0; i--)
        {
            rev = rev + given.charAt(i);
        }
        if(given.equals(rev))
            S.o.pln("palindrome");
        else
            S.o.pln("Not a palindrome");
    }
}
```

23/09/22

STRING BUFFER

java.lang.StringBuffer:

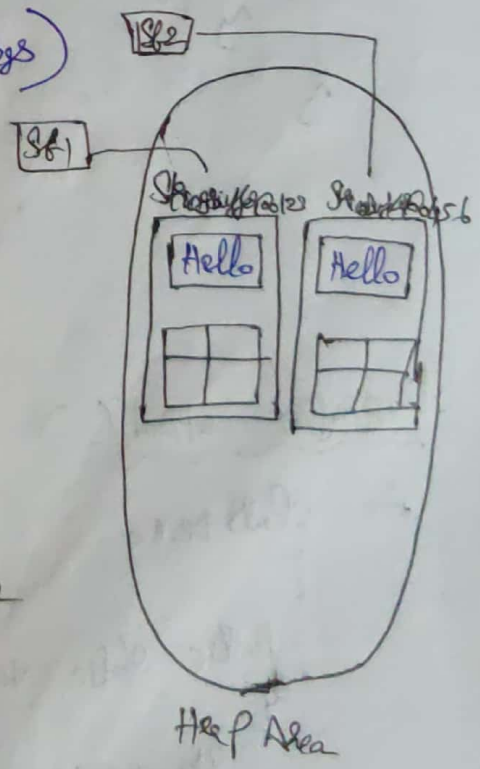
* It is a inBuilt class defined in a java.lang Package.
* It is a final class.
* It helps to create mutable instance of String.
* StringBuffer does not have SCP.
* It inherits java.lang.Object class.
* In StringBuffer equals(), hashCode() methods of java.lang.Object class are not overridden.

It implements:

* Serializable
* CharSequence

Ex1:

```
Class Demo
{
    Public static void main (String [] args)
    {
        StringBuffer sb1, sb2;
        sb1 = new StringBuffer ("Hello");
        sb2 = new StringBuffer ("Hello");
        S.o.Pln (sb1);
        S.o.Pln (sb2);
        S.o.Pln (sb1 == sb2); // false
        S.o.Pln (sb1.equals (sb2)); // false
    }
}
```



WAJP to convert String into StringBuffer.

```
Public class S8
{
    Public static Void main (String[] args)
    {
        String s = "Laddu";
        // StringBuffer <--- String
        StringBuffer res = new StringBuffer(s);
        S.o.Pln (res);
    }
}
```

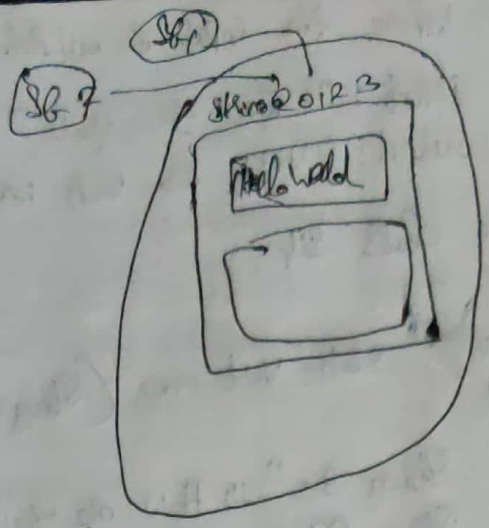
WAJP to convert StringBuffer into String.

```
Public class S9
{
    Public static Void main (String [] args)
    {
        StringBuffer sb = new StringBuffer ("Laddu");
        S.o.Pln (sb);
    }
}
```

StringBuffer append()

```
Ex: class Demo
{
    Public static void main (String [] args)
    {
        StringBuffer sb1, sb2;
        sb1 = new StringBuffer ("Hello");
```

```
sb2 = sb1;
S.o.Pln(sb1); // Hello
S.o.Pln(sb2); // Hello
sb1.append("World");
S.o.Pln(sb1); // Hello World
S.o.Pln(sb2); // Hello World
S.o.Pln(sb1==sb2); // true
S.o.Pln(sb1.equals(sb2)); // true
}
}
```



NOTE:

* We can create mutable objects with respect to StringBuffer and StringBuilder.

WAJP where the input & output is given below,

input : Hi how are you?

output : you? are how Hi

```
Public class S13
{
    Public static void main(String [] args){
        String s = "Hi how are you?";
        String res[] = s.split(" ");
        for(int i=res.length-1; i>=0; i--)
        {
            S.o.Pln(res[i] + " ");
        }
    }
}
```

O/P

you? are how Hi

WAP where the input & output given below,

input : Hi how are you ?
output : ? uoy era woh iH

```
Public class Sif
{
    Public static void main (String[] args)
    {
        String s = " Hi How are You? ";
        String [] res = s.Split(" ");
        String f_res = " ";
        for (int i = res.length-1 ; i>=0 ; i--)
        {
            // index -> 3-0
            f_res += reverse(res[i]) + " ";
        }
        S.o.Pln ("INPut:" + s);
        S.o.Pln ("outPut:" + f_res);
    }
    Public static String reverse (String s)
    {
        String rev = " ";
        for (int i = s.length()-1 ; i>=0 ; i--)
            rev += s.charAt(i);
        return rev;
    }
}
```

CONSTRUCTORS

CONSTRUCTORS	DESCRIPTION
StringBuffer()	Creates empty String with initial capacity 16.
StringBuffer(String str)	Creates String buffer with the specified String.

IMPORTANT METHODS of STRINGBUFFER:

RETURN TYPE	METHOD NAME	DESCRIPTION
int	capacity()	Returns current capacity
int	length()	Returns length of the String.
char	charAt(int index)	Returns the character of the specified index.
StringBuffer	append(String s)	Join Strings. (overloaded method).
StringBuffer	insert(int index, String s)	insert a specified String into original String of specified index. (overloaded method).
StringBuffer	delete(int begin, int end)	Delete String from specified beginning index to end-1 index.

RETURN TYPE	METHOD NAME	DESCRIPTION
StringBuffer	deleteCharAt(int index)	Delete the character present in the specified index.
StringBuffer	reverse()	Reverse the String
StringBuffer	setLength(int length)	only specified length in a String is exact remaining get removed.
StringBuffer	subString(int begin)	Returns the substring from the specified beginning index.
StringBuffer	subString(int begin, int end)	Returns substring from specified beginning index to end-1 index.
StringBuffer	replace(int begin, int end, String s)	Replace a specified String from the beginning index to end-1 index.
Void	trimToSize()	Removes the Unused capacity or set the capacity till length of the String.
Void	setCharAt(int index, char new char)	Replace the new character in a String of specified index
Void	ensureCapacity(int Capacity)	Sets capacity for storing a String.
—	—	—

DIFFERENCE BETWEEN STRINGBUFFER AND STRINGBUILDER.

STRING BUFFER	STRINGBUILDER
*All the method present in StringBuffer is Synchronized.	*All the Method present in StringBuilder is non-Synchronized.
*At a time only one thread is allowed to access String Buffer object. Hence it is Thread Safe.	*At a time multiple thread is allowed to access StringBuilder object. Hence it is Not Thread Safe.
*Threads are required to Wait to operate a StringBuffer object. Hence Relatively Performance is low.	*Threads are not required to Wait to operate a StringBuilder object. Hence Relatively Performance is high.
*Less efficient than StringBuilder	*Efficiency is high compared to StringBuffer.
*introduced in 1.0V	*Introduced in 1.5V

24/09/22

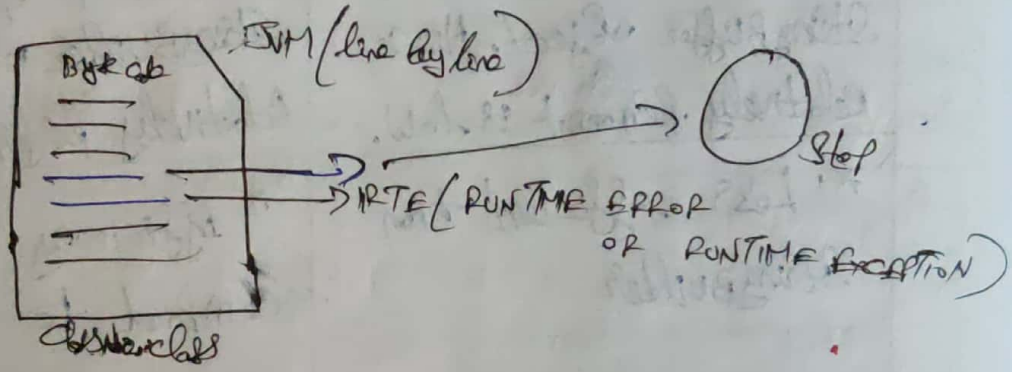
EXCEPTION HANDLING

i) Exception

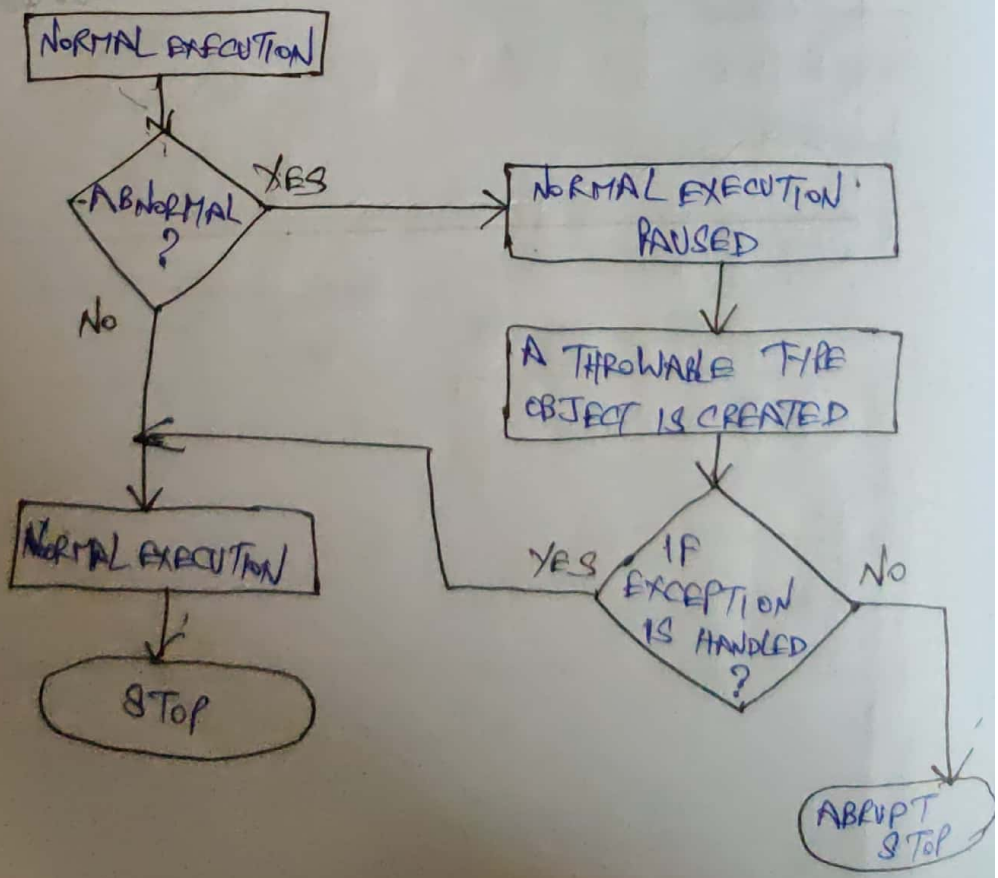
* The Exception is a Problem that occurs during the execution of a Program (Runtime).
* When an exception occurs, the execution of the Program stops abruptly (Unexpected Stop).

Note:

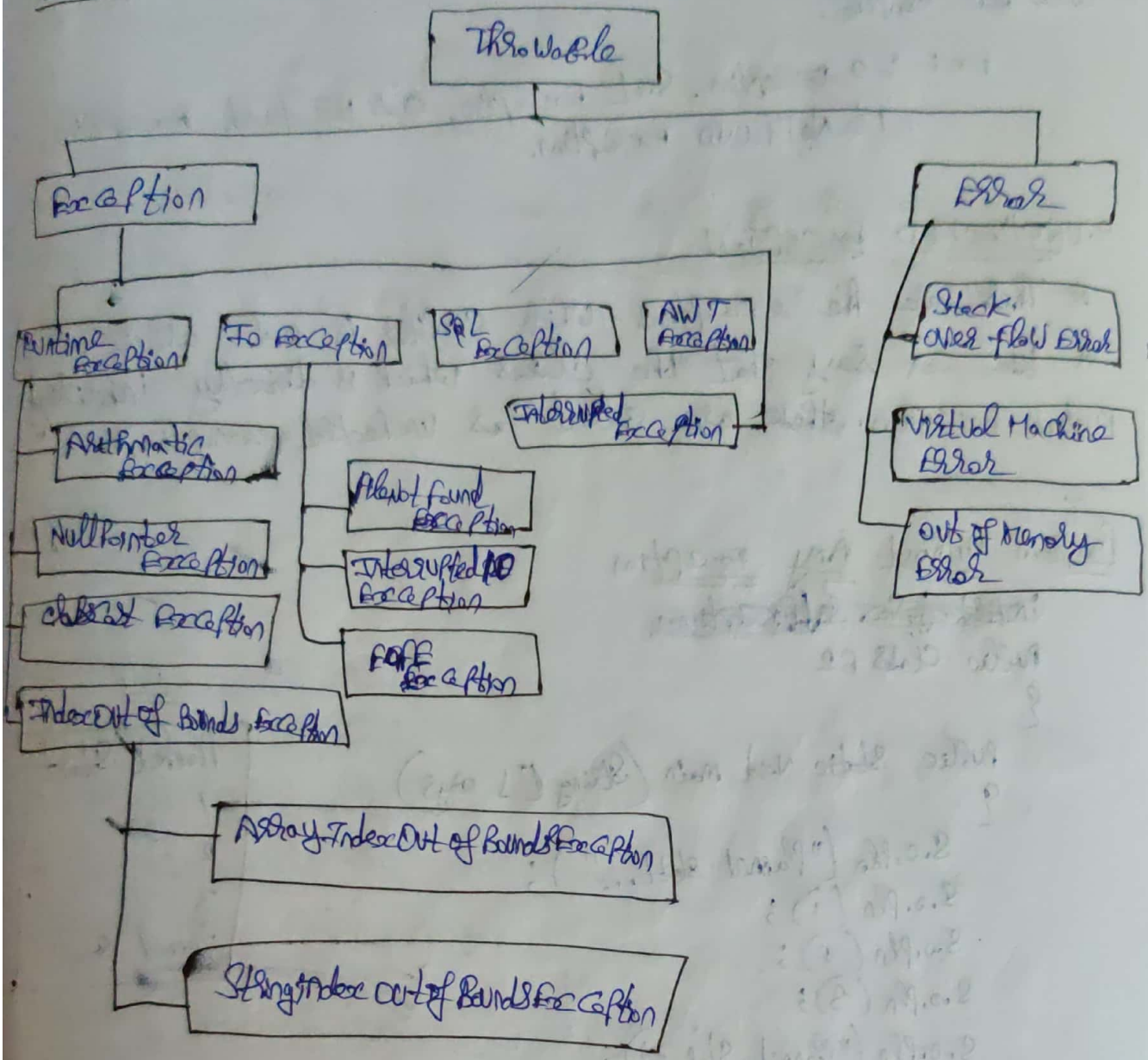
Every exception in Java is a class of 'Throwable type'.



WHAT HAPPENS IF AN EXCEPTION OCCURS?



HIERARCHY OF EXCEPTION CLASS



Types of Exception :

* We are having two types of Exception,
  1. Checked Exception
  2. Unchecked Exception

1. Checked Exception

* Compiler aware exception is called as checked exception.
* We know the hierarchy of exception classes, the classes which is directly inheriting ~~Throwable~~ Exception class except RunTime Exception and error class is called as checked exception.

* Checked exception if it is present in the Program, Compiler will not compile.

Ex: IO exception, SQL Exception, class NOT found exception, File NOT FOUND Exception.

2. UNCHECKED EXCEPTION

* These are the exceptions which compiler is not aware.
* We can say that the classes which is directly inheriting Runtime Exception., these are called as Unchecked exception.

Program without Any Exception

```
Public class EE
{
   Public static void main (String [] args)
   {
      S.o.Pln ("Planned start...");
      S.o.Pln (1);
      S.o.Pln (2);
      S.o.Pln (3);
      S.o.Pln ("Planned Stop...");
   }
}
```

O/P

```
Planned Start
1
2
3
Planned Stop
```

Program Along with the Exception

```
Public class EE {
   Public static void main (String [] args) {
      S.o.Pln ("Planned start");
      S.o.Pln (1);
      S.o.Pln (2/0);
      S.o.Pln (3);
      S.o.Pln (4);
      S.o.Pln ("Planned Stopped");
   }
}
```

O/P

```
Planned start
1
```

* As we can see that the above program is not meeting with Planned Start due to Exception.

WHAT IS EXCEPTION HANDLING?

Maintaining the normal flow of the application by using try and catch block is called as exception handling.

26/09/22

try { }

* Inside the try block we will write those statements which can be the reason for the abrupt stop of my application.
* We cannot use try block alone, it should be always followed by catch block.

catch (Throwable t) { }

* Catch block only it will execute if catch is useful (Runtime created objects has to be stored in throwable type of reference).

Example Program by Maintaining the Normal flow of the Application.

```
Public class E2 {
    Public static Void main (String [] args) {
    S.o.Pln (1);
    try {
        S.o.Pln (2);
        S.o.Pln (3/0); //stop -> throw new A.E()
        S.o.Pln (4);
    }
    catch(ArithmaticException e){
        S.o.Pln(e);
        S.o.Pln("I Told you Na U can't divide by zero");
    }
    S.o.Pln ("Bye");
    }
}
```

Example Program Without Maintaining the Normal flow of the Application.

```
Public class E4{
    Public static Void main(String [] args) {
        S.o.Pln ("start");
        String .S = null;
        S.o. Pln(1);
        S.o.Pln (S.charAt(2)); //RTException
        S.o.Pln("Bye");
        S.o.Pln ("End...");
    }
}
```

Program By Handling NullPointerException.

```
public class E4 {
    public static void main(String[] args) {
        S.o.Pln("start");
        String s=null;
        try {
            S.o.Pln(1);
            S.o.Pln(s.charAt(2));
            S.o.Pln("Hi"); // Ignored
        }
        catch(NullPointerException e) {
            S.o.Pln("calling Non-static member with null ref");
        }
        S.o.Pln("Bye");
        S.o.Pln("End");
    }
}
```

O/P

```
start
1
calling Non-static member with null ref
Bye
End
```

Can one try block can be followed by multiple catch blocks.

Yes, one try block can be followed by multiple catch blocks.

Ex:

```
Public class E5 {
    Public static void main(String [] args) {
    ...try {
            S.o.Pln("Hi");
            String s = null;
            S.o.Pln(4/0);
            S.o.Pln(s.charAt(6));
        }
        Catch(NullPointerException e)
        {
            S.o.Pln("Dance");
        }
        catch(ArithmeticException e) {
            S.o.Pln("Sing");
        }
        S.o.Pln("Bye_Bye");
    }
}
```

o/p

Hi
Sing
Bye_Bye

Tracing part

try {
6 — JVM ✓ RTS
7 — s=null ✓ RTS
8 (4/0) Stop → throw A.E()
9 — ⊗ } Abnormal stop
10 — ⊗ }
}
catch(N.P.E ref)
{ => }
catch(A.E ref)
{
}

finally { }

* It is a block which has to be followed by the try { }.

* Inside the finally block we will write those statements which must & should execute even exception occured or not occured.

Program with finally block with no exceptions

```
public class Ex6 {
    public static void main (String [] args) {
        S.o.Pln("1");
        S.o.Pln(2);
        try {
            S.o.Pln(3);
        }
        catch (ArithmeticException e) {
            S.o.Pln ("Hi From Pinga");
        }
        finally {
            S.o.Pln ("Hi an VIP Statement");
        }
        S.o.Pln ("Bye");
    }
}
```

o/p

```
1
2
3
Hi Im VIP Statement
Bye
```

Program along with exception and finally block execution.

```
public class E7 {
    public static void main(String[] args) {
    S.o.Pln("Java");
    try {
        S.o.Pln("SQL");
        S.o.Pln(4/0); // throw
        S.o.Pln("Testing");
    }
    catch (Throwable e) {
        S.o.Pln("Can't Divide By 0");
        try {
            String s=null;
            S.o.Pln(s.length());
        }
        catch (NullPointerException A) {
            S.o.Pln("Null Ref");
        }
        finally {
            int a[] = {4,5,6,5};
            S.o.Pln("JDBC");
            try {
                S.o.Pln(a[5]);
            }
            catch (ArithmeticException e)
            {
                S.o.Pln("1");
            }
            catch(NullPointerException ee) {
                S.o.Pln(3);
            }
            catch(ArrayIndexOutOfBoundsException e) {
                S.o.Pln(2);
            }
```

```
        S.o.Pln(2);
    }
    S.o.Pln("Hihihi");
}
S.o.Pln("Bye");
}
}
```

o/p

27/09/22

throw

* It is a keyword
* It is used to throw an exception manually.
* By using throw we can throw checked exception, unchecked exception. It is mainly used to throw the custom exception.

Syntax:

```
throw exception;
throw new CustomException("String"); // Exception class created by Programmer
                (or)
throw new ExceptionClass();
```

Program for creating manual exceptions by using throw keyword

```
public class E9 {
    public static void main(String [] args) {
    S.o.Pln(12);
    S.o.Pln(13);
    try {
        throw new ArrayIndexOutOfBoundsException();
    }
    catch (NullPointerException e) {
        S.o.Pln(e);
    }
    finally
    {
    }
    catch (ArrayIndexOutOfBoundsException e) {
        S.o.Pln("object is catched");
    }
    finally {
        S.o.Pln("Dinga");
        try {
            throw new NullPointerException();
        }
        catch (Exception e) {
            S.o.Pln("Dingi");
        }
    }
    S.o.Pln("Bye");
}
}
```

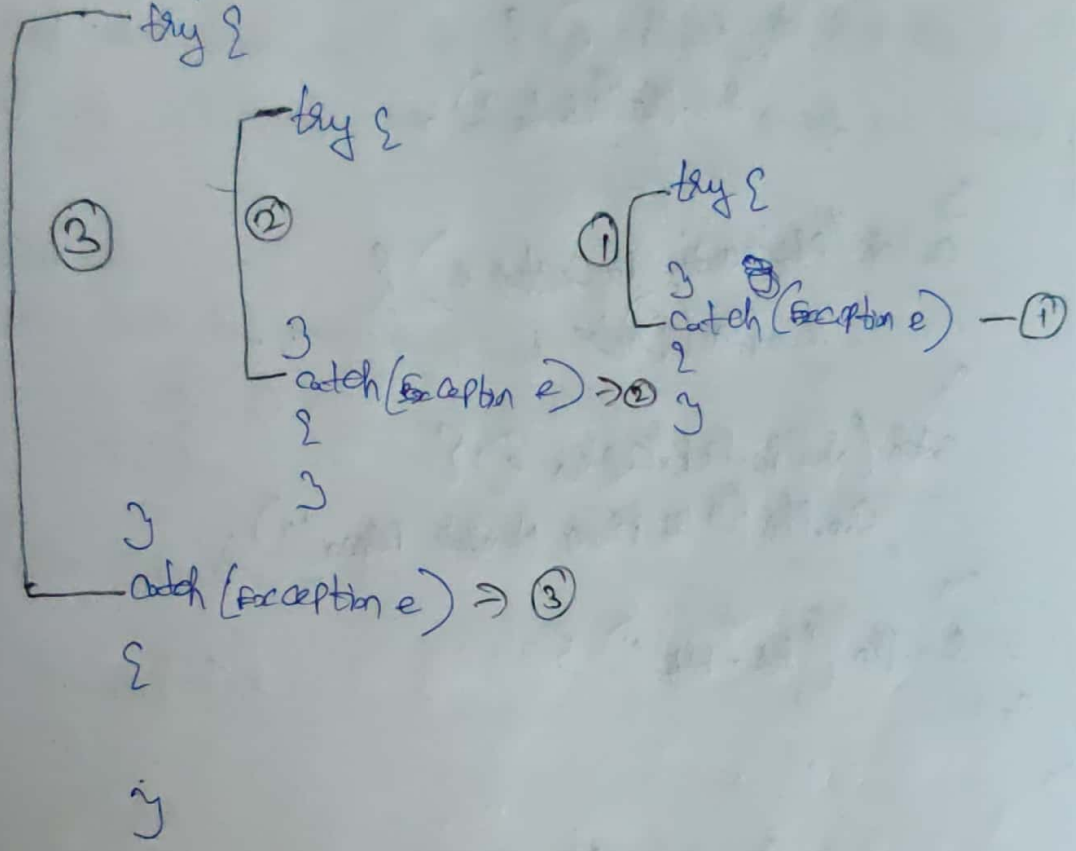
o/p

12
13
object is catched
Dinga
Dingi
Bye

Nested try { }

one try and catch block present inside another try block is called as Nested try Block.

Ex:



Example Program

```
public class Eio {
    public static void main(String [] args) {
        try {
            S.o.Pln("one");
            String s="cat";
            S.o.Pln(s.length()); //3
            S.o.Pln("Hi");
            try {
                S.o.Pln("This is Laddu");
                S.o.Pln(4/0); //Exception obj
                S.o.Pln("Ram");
```

```
try {
    S.o.Pln("This is My Rasagullah");
}
catch (Exception e) {
    S.o.Pln("JK");
}
}
catch (Exception e) {
    S.o.Pln("Sita");
}
}
catch (ArithmaticException e) {
    S.o.Pln(e);
}
catch (NullPointerException e) {
    S.o.Pln("Hi From NumberFormat");
}
S.o.Pln("Bye-Bye");
}
}
```

o/p

one
3
Hi
Thats it Laddu
Sita
Bye-Bye

28/09/22

OBJECT PROPAGATION IN EXCEPTION HANDLING

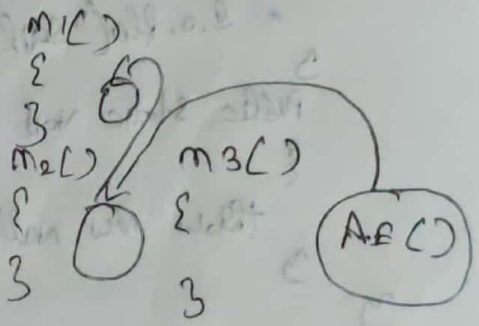
The process of moving exception object from one place to another is called as object propagation.

object Propagation with respect to Unchecked exception

* Unchecked exceptions will be Propagated Implicitly.
* It will only Propagate to the caller if it is not handled.

Ex:

```
Public class E14 {
    Public Static Void main(String [] args) {
        S.o.Pln (1);
        m1 ();
    }
    Public Static Void m1()
    {
      S.o.Pln(2);
      try {
      m2();
      }
      catch (Exception e) {
          S.o.Pln ("ateched in m1()");
      }
    }
    Public Static Void m2()
    {
      S.o.Pln(3);
      m3();
    }
    Public Static Void m3()
    {
      S.o.Pln(4);
      S.o.Pln(4/0); // new ArithmaticException()
    }
}
```



o/p

1
2
3
4
ateched in m1()

Example Program for Unchecked Exception & manually creating exception.

```
Public class E15{
    Public static void main (String [] args){
        S.o.Pln ("Hai");
        test1 ();
        S.o.Pln ("Bye");
    }
    Public static void test1()
    {
        S.o.Pln ("Java");
        try {
            test2();
        }
        catch (Exception e) {//catching
            S.o.Pln ("Bye");
        }
        S.o.Pln("Hello");
    }
    Public static void test2()
    {
        throw new NullPointerException(); //Ball
    }
}
```

OBJECT PROPAGATION WITH RESPECT TO CHECKED EXCEPTION

* Checked exception will be not propagated implicitly, we have to propagate it explicitly with the help of 'throws' keyword.

throws

* It is a keyword which has to be used in the method declaration.

* If we are declaring throws in the method declaration we should declare along with their respective exception class.

We can declare more than one exception with the help of throws keyword.

Can We explicitly Propagate Unchecked exception?

Yes, We can explicitly Propagate unchecked exception.

Example Program without exception but Declaring the Exception (throwing to the caller)

```
Public class E182
    Public static void main(String[] args) throws Exception {
        S.o.Pln("Hi");
        Sleep(100);
    }
    Public static Void Sleep(int a) throws InterruptedException
    {
        S.o.Pln("Hi Iam Ref");
    }
}
```

Customized Exception

* The Exception classes (or) the exception type classes created by the Programmer is called as Customized exceptions.

Steps to create customized Exception

1. Create Exception type of class. extending Exception class.
2. Create Parameterized method in the Exception type of class which will accept String data.
3. Call the Parameterized Constructor of exception class from the child class constructor with the help of super call statement and Pass the message.

Ex:

```
import java.util.Scanner;
Public class MatrimonyA {
    static Scanner s=new Scanner(System.in);
    Public static void main(String [] args) {
        S.o.Pln("Enter the Age");
        Int age = s.nextInt();
        try{
            isEligible(age);
        }
        Catch(ICanUnderStand e){
            S.o.Pln(e);
        }
    }
    Public static Void isEligible(int age) throws ICanUnderstand
    {
        if(age>=18)
        {
            S.o.Pln("Congratulations Welcome to Hell");
        }else
        {
            // Create an Exception
            throw new ICanUnderStand(  );
        }
    }

Public class ICanUnderStand extends Exception{
    Public ICanUnderStand() {
        //Todo Auto generated Constructor stub
    }
    Public ICanUnderStand(String str)
    {
        super(str);
    }
}
```

29/09/22

Wrapper class

* Java is not 100% object oriented Language, we will store the data even in Primitive type of container.

Ex:

Stored in Primitive type of container

```
int a = 5;        // Not stored as object
a.toString();
```

Stored as object

```
Student s = new Student();
s.toString();
```

* The Wrapper class in java provides mechanism to wrap the primitive into an object.

* for every primitive data type corresponding class is declared known as a Wrapper class.

* There are eight Wrapper classes declared in java.lang Package which provides several methods to convert Primitive into an object.

Wrapper classes

Primitive datatypes	Wrapper classes
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

Note: Among the Wrapper classes Byte, Short, Integer, Long, Float, Double are subclasses of Number class.

WAP to convert int to Integer.

```
Public class Test2 {
    Public static void main (String [] args) {
    //Boxing
    //NPDT <-- Short
    [redacted].
    Integer [redacted] = Integer.valueOf(25);
    S.o.Pln(Ref); //UnBoxing
    int [redacted] = i.intValue();
    S.o.Pln(res);
    }
}
```

30/09/22

BOXING

What is Boxing?

* The Process of converting one Primitive datatype into another Non-Primitive datatype is called as Boxing.
* We can achieve Boxing with the help of a method valueof() method.

valueof() Method

We can Wrap a Primitive value to corresponding Wrapper class object by using a valueof() method.

Declaration:

```
Public static Wrapper class valueof(String)
Public static Wrapperclass valueof(Primitive)
```

WAJP to convert all Primitive datatypes into Non-Primitive datatype

```
class Demo
{
  public static void main(String[] args)
  {
    //Primtive data
    byte b=10;
    short s=20;
    int i=30;
    long l=40;
    float f=10.0f;
    double d=2005;
    char ch='a';

    // Converting Primitive to object
    Byte obj=Byte.valueof(b);
```

```
Short obj2 = Short.valueOf(s);
Integer obj3 = Integer.valueOf(i);
Long obj4 = Long.valueOf(l);
Float obj5 = Float.valueOf(f);
Double obj6 = Double.valueOf(d);
Character obj7 = Character.valueOf(c);
String str = String.valueOf(i);
}
}
```

UNBOXING

* The Process of converting Non-Primitive datatype back to Primitive datatype is called as UnBoxing.
* We can achieve UnBoxing with the help of one Non-static method that is called as Value() method.

WAJP to achieve UnBoxing.

```
class Demo1
{
    public static void main(String[] args)
    {
        //Boxing
        Integer i = Integer.valueOf(25); // 25 is stored as NPDT
        Character ch = Character.valueOf('A');
        Boolean b = Boolean.valueOf(false);

        //Boxing
        int res = i.intValue();
        S.o.Pln(res); // PDT
        char res1 = ch.charValue();
        S.o.Pln(res1); // PDT
        boolean res2 = b.booleanValue();
        S.o.Pln(res2);
```

AUTO-BOXING

The Process of implicitly converting a Primitive data type into corresponding Wrapper class (object) is known as Auto-Boxing.

Ex:

int - Integer
byte - Byte
boolean - Boolean, etc.,

AUTO-UNBOXING

The Process of implicitly converting a object into Primitive datatype is known as auto-Unboxing.

Ex:

Integer - int
Byte - byte
Boolean - boolean, etc.,

WAJP for Auto-Boxing & Auto Un-Boxing.

```
Public class {
    Public static void main(String [] args) {
    // Auto-Boxing
    Integer ref1 = 1; // Integer.valueof(1); (Compiler will add)
    Character ref2 = 'A';
    Short ref3 = 45;
    Boolean ref4 = true;
    Double ref5 = 4.3;
```

```
//Auto-unBoxing
int a = ref1;
char b = ref2;
double d = ref5; //ref5.doubleValue(); (Compiler will add)
S.o.Pln(ref1 + "-NPDT");
S.o.Pln(ref2 + "-NPDT");
S.o.Pln("****");
S.o.Pln(a + "---PDT");
S.o.Pln(b + "---PDT");
}
}
```

Conversion of String

If we want any Primitive Datatype into String which is Non-Primitive Datatype, we have call the valueof() method of the String class.

Ex:

```
Public class Wrt {
    Public static void main (String [] args) {
        int a = 5;
        // String <--- PDT
        String ref = String.valueof(a);
        S.o.Pln(ref);
        byte b = 6;
        String ref2 = String.valueof(b);
        S.o.Pln(ref2);
        char ch = 'A';
        String ref3 = String.valueof(ch);
        S.o.Pln(ref3);
    }
}
```

Parsing

The Process of Converting String NPDT ~~into~~ back to PDT is called as Parsing.

Note:

Parse method is present in all the Wrapper except character and String.

WAJP to convert String into Primitive Datatype

```
class Demo
{
    public static void main(String [] args)
    {
        String str = "ten";
        byte b = Byte.ParseByte(str);
        short s = Short.ParseShort(str);
        int i = Integer.ParseInt(str);
        float f = Float.ParseFloat(str);
        double d = Double.ParseDouble(str);
    }
}
```

01/10/22

COLLECTION FRAMEWORK

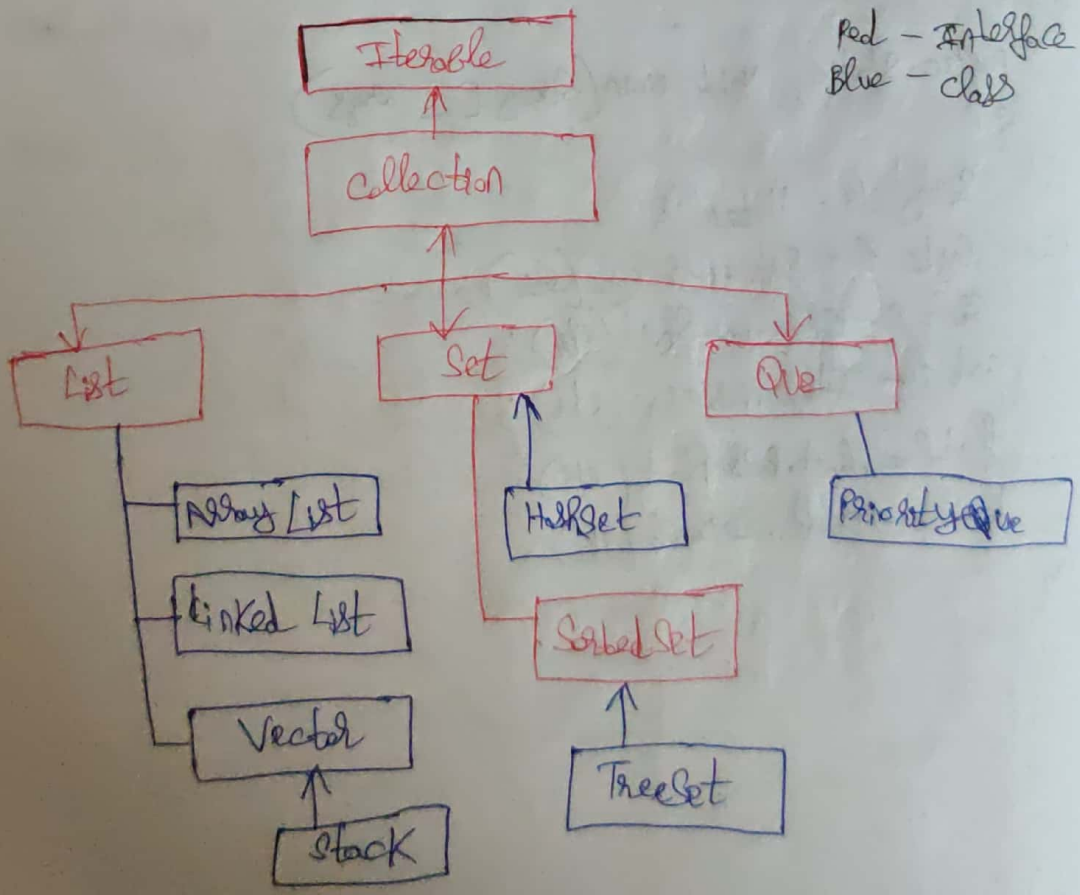
What is collection?

The word collection specifies storing the group of objects together.

What is framework?

It is the readymade architecture which will help the programmer to perform operations such as Adding the elenent, Removing an elenent, Searching an elenent, and ~~deleting~~ Sorting an elenent ~~through~~ to the collection.

What is collection framework?



* Collection Framework is the combination of set of classes and Interface together.
* It is a readymade architecture which will help to create a collection and to store the objects in to the collection and also to perform CRUD operations for the collection.

NEED of COLLECTION FRAMEWORK

* We go for Collection FrameWork, to overcome the disadvantage of an array.

Disadvantages of an array

* Size of an array is fixed, we cannot increase or decrease dynamically
* Array is a collection of homogeneous element
* If we want to perform any manipulations such as searching, sorting, adding, removing and deleting (CRUD) operations, is complex.
* Therefore, there is a need of collection frameworks.
* Therefore In order to overcome the disadvantage of an array, we will use the Data structures such as List, Set, Que and Maps / dictionaries.

COLLECTION INTERFACE

1. Collection is an interface defined in java.util.Package.
2. Collection interface Provides the mechanism to store group of elements (objects) together.
3. All the elements in the collection are stored in the form of objects (i.e. only Non-Primitive datatypes are allowed).

4. Collection interface provides the abstract methods to the programmer to perform the following task.

a. Add an element into the collection
b. Search an element in the collection
c. Remove an element in the collection
d. Access/read the elements present in the collection.

Methods of Collection interface:

Purpose	Return type	Method Signature
Add an element		1. add(object) 2. addAll(collection)
Search an element		1. contains(Object) 2. containsAll(collection)
Remove an element		1. remove(Object) 2. removeAll(collection) 3. retainAll(collection) 4. clear()
Access an element	Iterator	1. iterator() 2. for each loop
Miscellaneous		1. size() 2. isEmpty() 3. toArray() 4. hashCode() 5. equals()

List Interface:

* List interface is defined in java.util package.
* List is a sub interface of collection interface. Therefore it has all the methods inherited from collection interface.

Characteristics of List interface:

* List is a collection of objects.
* List maintains the insertion order of elements. The element inserted first, it will be maintained in the 1st position.
* It allows duplicate entries. We can add same object (element) multiple times.
* List supports indexing. We can insert, search, remove, access elements with the help of index.
* The index starts from zero upto (list.size-1).

Methods of List Interface:

List interface will have all the inherited abstract methods from iterable and collection interfaces.

Purpose	Return-type	Method Signature
Add elements		add(Object) addAll(Collections) ========== add(int index, Object) addAll(int index, Collection)
Search an element	boolean boolean int	contains(Object) containsAll(Collection) ========== indexof(Object)
Removing an element	boolean boolean boolean	remove(Object) removeAll(Collection) retainAll(Collection) clear() ========== remove(int index)

Access an element	Iterator	iterator() get(int index) listIterator() listIterator(int) We can access elements using for each loop
Miscellaneous		size() isEmpty() toArray() hashCode() equals()

ArrayList:

1. It is a concrete implementing class of List interface.
2. Characteristics of ArrayList is same as List.
3. ArrayList is defined in java.util Package.

Note:

1. All the abstract methods of List and Collection interface is implemented.
2. We can create an object (instance) ArrayList.

Constructors:

ArrayList()	Creates an empty ArrayList Object.
ArrayList(Collection c)	Creates an ArrayList Object, and copies all the elements present in the given Collection into the ArrayList created.

03/10/22

WAP to create a collection of List type and Prove the characteristic of list data structure and print the size.

```
Package arraylist;
import java.util.ArrayList;
import java.util.List;
Public class A4 {
    Public static void main (String [] args) {
        List ls = new ArrayList();
        ls.add ("Tabrez");
        ls.add (1.35);
        ls.add (true);
        ls.add (null);
        ls.add (1.35);
        S.o.Pln ("Size:" + ls.size());
        S.o.Pln(ls);
    }
}
```

o/p

```
Size:5
[Tabrez, 1.35, true, null, 1.35]
```

WAP to add one collection into another collection.

```
Package arraylist;
import java.util.ArrayList;
import java.util.List;
Public class A5 {
    Public static void main (String [] args) {
```

```
List raju = new ArrayList();
List rani = new ArrayList();
// Raju Bag contains The Element
raju.add("Java");
raju.add("Sql DataBase");
// Adding Books into Rani Bag
rani.add("IDBC");
rani.add("Block Chain");
//Adding books into r
S.o.Pln("Raju_" + raju);
S.o.Pln("Rani_" + rani);
// Adding Raju collection into Rani's c
rani.addAll(raju);
S.o.Pln("After Adding Raju collection into Rani collection");
S.o.Pln("Rani's _" + rani);
}
}
```

Note:

If we want to Add only elements of other collection we have to use addAll().

WAP to retain one collection elements into another collection.

```
Package arrayList;
import java.util.ArrayList;
Public class AG {
    Public static void main(String[] args) {
        ArrayList l1 = new ArrayList();
        ArrayList l2 = new ArrayList();
        l1.add("A");
        l2.add("B");
        l1.add("C");
```

```
l1.add("D");
S.o.pln("L1-"+l1);
l2.add("A");
l2.add("B");
l2.add("C");
l2.add("D");
l2.add("E");
l2.add("F");
S.o.pln("L2-"+l2);
S.o.pln("********");
l2.retainAll(l1); //l2 is modified only keeping l1 object
S.o.pln(l2);
}
}
```

o/p

L1- [A B C D]

L2- [A B C D E F]

****

A B C D

WAP to create a collection of cookies and check the whether cookies. what user entered present or not?

```
Package arraylist;
import java.util.ArrayList;
Public class A7 {
    static Scanner s = new Scanner(System.in);
    Public static void main(String [] args) {
        ArrayList cookies = new ArrayList();
        cookies.add("Parle-G");
        Cookies.add("Hide And Seek");
```

```
cookies.add("Good Day");
cookies.add("Happy Happy");
S.o.Pln("Enter the cookie name to search");
String name = s.nextLine();
If (cookies.contains(name))
    S.o.Pln("Present");
else
  S.o.Pln("Not Present");
}
}
```

04/10/22

WAJP TO REMOVE THE OBJECT FROM THE COLLECTION.

```
Package arraylist;
import java.util.ArrayList;
import java.util.List;
Public class A9 {
    Public static void main (String [] args) {
        List ls = new ArrayList();
        ls.add(1); //POT --- Integer --- object
        ls.add("Tobby");
        ls.add("cat");
        ls.add("Java");
        ls.add(5);
        ls.add(0);
        S.o.Pln(ls);
        // Remove
        S.o.Pln("\n*** After Removing ***\n");
        ls.remove(Integer.valueof(1));
        S.o.Pln(ls);
        ls.clear()
        S.o.Pln(ls);
    }
}
```

Accessing The Collection

The elements of an ArrayList can be accessed in the following Way:

1. get method
2. Iterator
3. ListIterator
4. for each loop / Advanced for-loop

1. get(index)

* Get method is used to access the elements present in the ArrayList.
* It accepts index as an argument.
* It returns the Object type.

Program related to get method

```
Package arraylist;
import java.util.ArrayList;
import java.util.List;
Public class Alo {
        Public static Void main (String [] args) {
                List ls = new ArrayList();
                ls.add ("Java");
                ls.add ("Sql");
                ls.add ("Maven");
                Object o = ls.get(2);
                S.o.Pln (o);
        }
}
```

For each Loop / Advanced for-loop

Syntax:

```
for(type Variable : reference Variable of the collection/array)
{
    Statements;
}
```

Example program for For each loop

```
Package arraylist;
public class Employee
{  //States
   String name;
   int id;
   //constructors
   Employee(String name, int id)
   {
      super();
      this.name = name;
      this.id = id;
   }
   @override
   Public String toString()
   {
     return "" +name+ " : " +id;
   }
```

```
package arraylist;
import java.util.*;
public class Company {
    static String res = "Not Present";
    static Scanner s = new Scanner(System.in);
    public static void main (String [] args) {
        List test_tantra = new ArrayList();
        test_tantra.add (new Employee ("Tabbz", 1468));
        test_tantra.add (new Employee ("Bharath", 5648));
        test_tantra.add (new Employee ("Avinash", 1467));
        S.o.Pln ("Enter The Key Element");
        String key = s.nextline();
        for (Object o ; test_tantra)
        {
            Employee e = (Employee) o;
            If (e.name.equals (key))
            . res = "Present"
        }
        S.o.Pln (res);
    }
}
```

06/10/22

Iterator

To access the elements of an ArrayList using iterator

* iterator() is a method which belongs to Iterable interface.
* iterator() method creates an Iterator type object and returns the reference.
* iterator() method returns the reference in Iterator (interface) type.

java.util.Iterator:

Iterator interface has 2 important methods to perform iteration.

1. hasNext():

It checks whether there is an element to be accessed from the collection, if present it returns true, else it returns false.

2. next():

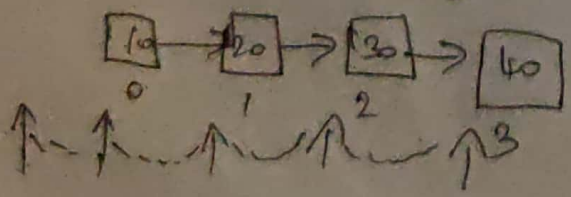
* It is used to access the element, and moves the cursor to the next element.
* The return type of next() is object.

Ex:

Step 1:

```
IS = [10, 20, 30, 40]
Iterator il = IS.iterator();
```

il.next()
il.next()
il.next()
il.next()
il.next() => NoSuchElementException



NoSuchElementException :

In the Iterator, when we try to access an element using next() method and if there is no element present, we get NoSuchElementException.

Example Program for Iterator using next() :

```
Public class A12 {
    Public static void main (String [] args) {
        List ls = new ArrayList ();
        ls.add ("Cat");
        ls.add (12);
        ls.add (14);
        ls.add ("Java");
        Iterator i = ls.iterator (); //Upcasting//Design
        S.o.Pln (i.next());
        S.o.Pln (i.next());
        S.o.Pln (i.next());
        S.o.Pln (i.next());
    }
}
```

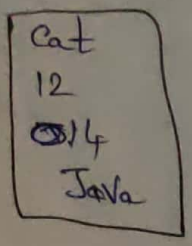
hasNext()

* In order to overcome NoSuchElementException, we have to call hasNext() method.
* It will return true if there is next element, else it will return false.

Example Program for Iteration using hasNext();

```
Public class A12 {
    Public static void main (String [] args) {
        List ls = new ArrayList();
        ls.add ("cat");
        ls.add (12);
        ls.add (14);
        ls.add ("Java");
        Iterator i = ls.Iterator(); // Design
        While (i.hasNext())
            S.o.Pln (i.next());
    }
}
```

O/P



Disadvantages of Iterator

1. We can iterate only once
2. We cannot access the elements in reverse order.
3. We cannot remove an element from the collection, during iteration. We get an Exception (ConcurrentModificationException).

07/10/22

LIST ITERATOR

ListIterator is a sub interface (child) of Iterator interface, it is defined in java.util.Package.

Methods of ListIterator:

hasNext()	To check whether next element to be accessed is present or not.
next()	It gives the element, and moves cursor forward
hasPrevious()	To checks whether previous element to be accessed is present or not
previous()	It gives the previous element pointed by the cursor, and moves the cursor backward.
remove(object)	It removes the current object pointed, from the collection, which is under iteration.
add(object)	It is used to add a new object into the collection.

Example Program

```
Package arraylist;
import java.util.*;
Public class A13 {
    Public static void main(String [] args) {
        List ls = new ArrayList();
        ls.add(10);
        ls.add("Java");
        ls.add(20);
        ls.add("PHP");
        //Create a design for Accessing
        ListIterator l = ls.listIterator();
        //forward Iteration
        While (l.hasNext()):
        {
          S.o.Pln (l.next());
        }
        S.o.Pln ("*****");
```

```
//Backward Direction
while(l.hasPrevious())
    s.o.println(l.previous());
}
}
```

TYPES of COLLECTIONS:

We are two types of collection. They are,

1. Homogeneous collection / Generic collection
2. Heterogeneous collection / Non-Generic collection

1. Homogeneous collection / Generic collection

It is a homogeneous (same type) collection of elements.

2. Non-Generic collection

* It is a heterogeneous (different type) collection of elements.
* Every element getting stored is converted to java.lang.Object type.

Syntax to create reference variable for Generic collection:

```
Collection_type<WPDT> variable;
ArrayList<Integer> al;
```

Syntax to create Generic collection object:

```
new collection_name<data_type>();
new ArrayList<Integer>();
```

In this ArrayList we can store only Integer.

From jdk 7 onwards,

```
ArrayList<Integer> al = new ArrayList();
```

WAJP TO CREATE A COLLECTION of STUDENT OBJECTS AND PRINT STUDENT DETAILS WHO IS HAVING MARKS LESSER THAN 85.

```
Package arrayList;
import java.util.*;
public class H1 {
    Public Static Void main (String [] args) {
        ArrayList<Student> ls = new ArrayList();
        ls.add(new Student (34, "Ram", 2));
        ls.add (new Student (75, "Sita", 3));
        ls.add (new Student (39, "Harish", 5));
        ls.add (new Student (47, "Karthik", 7));
        for (Student s : ls)
        {
            if (s.marks < 35)
                s.o.pln (s);
        }
    }
}
```

SORTING THE COLLECTION

* We can sort the collections by using one method which is present in collections class (i.e) collections.sort(list);

Rules for sorting

* We have to follow following given rules if we want to sort the collections.
* collection should be homogeneous
* The elements present inside the collection should be of comparable type.
* If we don't follow these rules, we will get classcast exception.

Example Program for LinkedList

```
import java.util.LinkedList;
Public class L1
{
   Public static void main (String [] args)
   {
      LinkedList ls = new LinkedList ();
      ls.add (56);
      ls.add (48);
      ls.add (56);
      ls.add (null);
      ls.add ("Tollu");
      S.o.println (ls);
      S.o.println ("Laddu");
      S.o.println (ls);
   }
}
```

o/p

```
[56, 48, 56, null, Tollu]
Laddu
[56, 48, 56, null, Tollu]
```

```
Public static void main (String [] args)
{
  ArrayList <Student> ls = new ArrayList ();
  ls.add (new Student (34, "Prabhu", 2));
  ls.add (new Student (75, "Ram", 3));
  ls.add (new Student (39, "Sita", 5));
  ls.add (new Student (47, "Hari", 7));
  for (Student s : ls)
  {
    if (s.marks < 35)
      s.o.pln (s);
  }
}
```

o/p

Program for Sorting Laptop based on price

```
Package arrayList;
Public class Laptop implements Comparable
{
  //States
  double price;
  String name;

  Public Laptop (double price, String name)
  {
    Super ();
    this.price = price;
    this.name = name;
  }
```

Student marklist Based on Id:

```
Package arraylist;
Public class Student
{
  //States
  double marks;
  String name;
  int id;
  //Constructors
  Public Student (double marks, String name, int id)
  {
    super();
    this.marks = marks;
    this.name = name;
    this.id = id;
  }
  @Override
  Public String toString()
  {
    Return "Name:" + name + " : Marks:" + marks + "id;
  }
}

Package Arraylist;
Import java.util.ArrayList;
Public class All {
  @Override
  Public String toString()
  {
    Return "Name :" + name + " Price :" + Price;
  }
  @Override
```

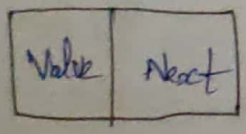
```
public int CompareTo (object o)
{
  Laptop l = (Laptop) o;
  if (this.Price == l.Price)
     Return 0;
  else if (this.Price > l.Price)
     return 1;
  else
    return -1;
}
}

import java.util.Arraylist;
Public class Amazon
{
   Public Static Void main (String[] args)
   {
     Arraylist<Laptop> ls = new Arraylist();
     ls.add (new Laptop (26000, "dell"));
     ls.add (new Laptop (45000, "Acer"));
     ls.add (new Laptop (28000, "Hp"));
     Collection.sort(ls);
     S.o.Pln (ls);

   }
}
```

Linked List:

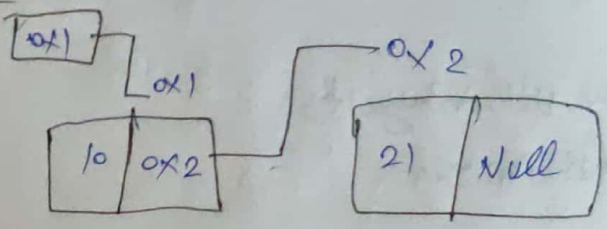
* Linked list is one of the implementation class of List interface.
* It is present Since JDK 1.2.
* The data will be stored in the linked list in the form of nodes.
* Each node will have two segment where one segment will stores the value and other segment stores the address of another node.

Node:

Ex1:



Ex2:



* Linked list implements marker interface such as Serializable, Cloneable. But not random access.
* In order to store the data the data structure used is Doubly linked list.
* Whenever object is created for linkedlist there is no memory with current capacity is created.

Characteristics of linkedlist

* Allows duplicate element into the collections.
* Null elements are allowed (null value).
* Indexing is present, we can access the element with respect to the index.

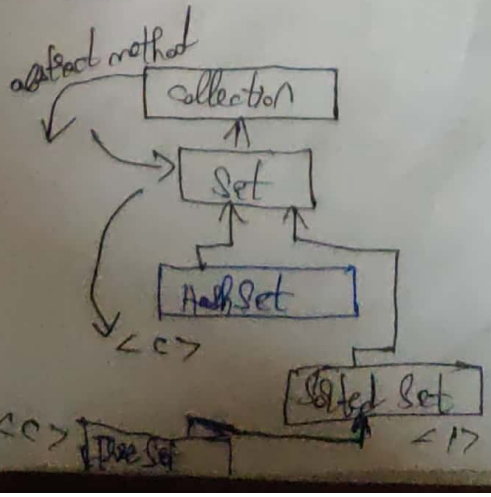
10/10/22

DIFFERENCE BETWEEN ARRAY LIST AND LINKED LIST.

ARRAY LIST	LINKED LIST
* Stores the Data in the form of Array.	* Stores the Data in the form of Nodes.
* Data Structure Used is Growable Array	* Data Structure Used is doubly LinKed list.
* 3 overloaded constructors are present.	* 4 overloaded constructors are Present.
* As Shift operations	* As No Shift operations
* Memory is contiguous	* Memory may not be contiguous

SET

* It is an interface which extends collection interface.
* It represents Unordered collection of elements.
* Whenever any element is being added for the Set type of collection internally Unique Key is Generated and the data is considered as Value. Based on the combination of Value and Key hashcode is generated.



Characteristics of Set:

*. It is an unordered collection of elements. (the order of insertion is not followed).

2. Set does not allow duplicate elements.

3. Set does not have indexing. Therefore, we cannot access, insert or remove elements based on index.

4. We can access the elements of Set by using:
   i) Iterator()
   ii) for each

Concrete Implementing class for Set interface:

1) HashSet:

* It is a concrete implementing class of Set interface.
* It has all the methods inherited from Collection interface.
* HashSet implements the interface such as Cloneable, Serializable & marker interface

Characteristics:

1. It is Unordered
2. No DUPLICATES
3. No index
4. Single Null element is added

WAP to create Set type of collections and check the characteristics of HashSet.

```
Package Set;
import java.util;
public class S1 {
    Public static void main(String[] args) {
        Set s = new HashSet();
        s.add(null);
        s.add(45);
        s.add("Tabrez");
        s.add("Laddu");
        s.add(null);
        s.println(s);
    }
}
```

o/p

[null, 45, Tabrez, Laddu]

Note:

* Whenever Object is created for the HashSet the current capacity will be 16.
* Present since JDK 1.2.
* It is having four overloaded constructors are present.

Linked HashSet

* It is a implementing class of Set Interface Present since JDK 1.4.
* Linked HashSet is similar to HashSet but it maintains order of insertion.

TreeSet

* TreeSet is a Concrete implementing class of Set interface.
* It is present since JDK 1.2.

Characteristics of TreeSet:

1. The elements will be in Sorted order by default.
2. The elements to be added in the TreeSet must be Comparable type, else we get class CastException.
3. All the elements in TreeSet should be of same type (Homogeneous), if not we get ClassCastException.
4. Duplicates are not allowed.
5. No Indexing, therefore adding and removing the elements is not possible with index.

Note:

* We cannot add null elements for TreeSet, we will NullPointerException.
* TreeSet implements the data structure which is called as Balanced Tree.
* We are having constructors present in the TreeSet such as,

Constructors:

TreeSet()	To create an empty TreeSet object.
TreeSet(collection)	It creates a treeSet object, and copies all the elements from the collection in the treeSet.

* For custom sorting, we have to go for Comparator.

Example Program

```
Package Set;
import java.util.TreeSet;
Public class S4 {
    Public static void main (String[] args) {
        TreeSet ts = new TreeSet ();
        ts.add ("Tabrez");
        ts.add ("Afroze");
        ts.add ("Bharath");
        ts.add ("AVinash");
        S.o.Pln (ts);
    }
}
```

O/P

[Afroze, AVinash, Bharath, Tabrez]

11/10/22

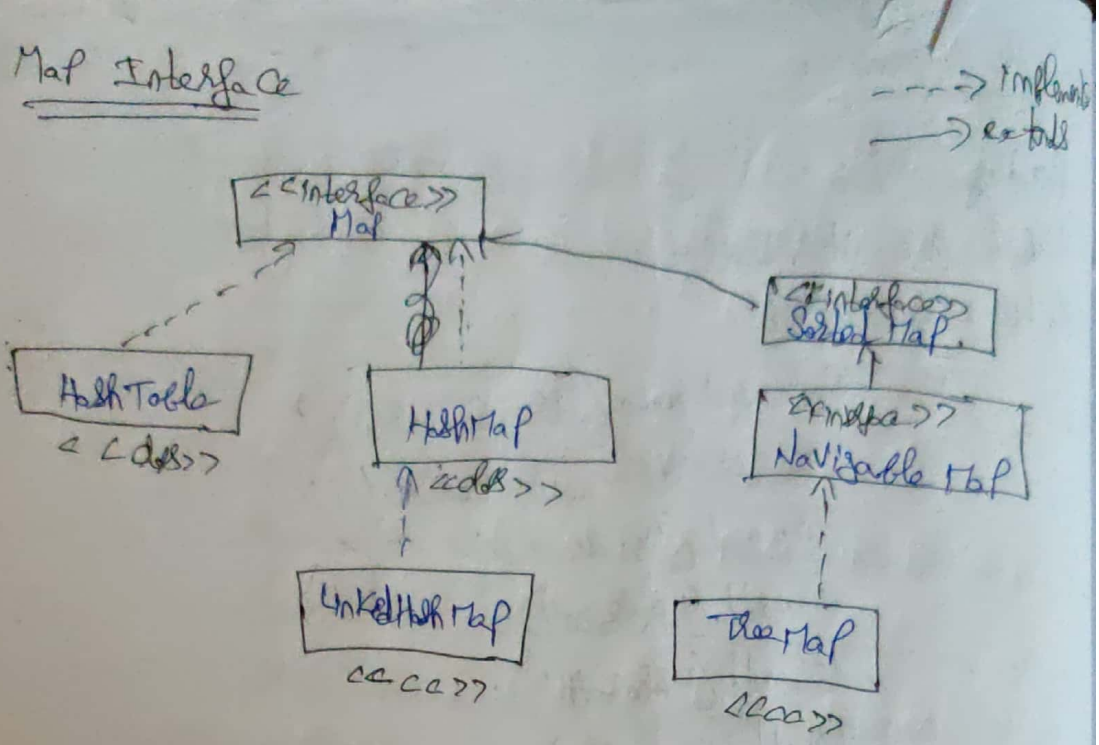
Maps

* Map is a data structure, which helps the programmers to the store the data in the form of key value pairs.
* Where every value is associated with a unique key.

Note:

1. Key cannot be duplicate
2. one key can be associated with only one value.
3. Maps helps us to access the values easily with the help of its associated key.

Map Interface



* Map is an interface present in java.util Package.
* Similar to collections we can also create generic maps.
* Whenever we are having any object into the map we will be having with Unique Key and Value Pair.

What is called as Entry?

One Key Value Pair together is called as Entry or Mapping.

We can obtain three different Views of a Map,

1. We can obtain a list of Values from a map
2. We can obtain a Set of Keys from a map
3. We can obtain a Set of Key-Value Pairs.

Ex: Map

{ 8-Smith, 3-John, 12-Tobez, 7-Smith }
↓ obj1 ↓ obj2 ↓ obj3 ↓ obj4

3 diff values

List to Key

{8, 3, 12, 7}

List of values

{Smith, John, Tobiez, Smith}

Entries

{8-Smith, 3-John, 12-Tobiez, 7-Smith}

Methods of Map interface:

Method Signature	Purpose
Put (Key, Value)	1. add an entry to the map (Key-Value Pair) 2. replace the old value with a new value of an existing entry in the map.
PutAll (Map)	It will copy all the entries from the given map into the current map.
containsKey (Key)	If the key is present returns true else returns false.
contains Value (Value)	If the value is present it returns true, else it returns false.
Remove (Key)	if the key is present the entry is removed from the map and the value is returned. If the key is not present nothing is removed and null is returned.
clear ()	it removes all the entries in the map.

12/6/22

get(key)	It is used to access the value associated with a particular key. If the key is present it returns the value. If the key is not present it returns null.
values()	it returns a collection of values present in the map. return type Collection<V>.
keySet()	it returns a set of keys present in the map return type Set<K>.
entrySet()	it returns a Set of all the entries present in the map. return type Set<<K,V>>
size()	
isEmpty()	
equals()	
hashCode()	

HashMap:

It is a concrete implementing class of Map interface.

characteristics

1. Data is stored in the form of key=value pair.
2. order of insertion is not maintained.
3. key cannot be duplicate, values can be duplicate.
4. key can be null
5. value can be null

Example Program

```
Package HashMap;
Import java.util.HashMap;
import java.util.Map;
Public class H1 {
    Public static void main (String[] args) {
        HashMap hs = new HashMap();
        hs.Put(12, "RaSgullah");
        hs.Put(13, "Laddu");
        hs.Put('A', "Pani Poori");
        hs.Put(51.3, "RaSgullah");
        hs.Put(null, null);
        S.o.Pln("obtaining 3 Diff views");
        S.o.Pln("***Keys***");
        S.o.Pln(hs.KeySet());
        S.o.Pln("***Values***");
        S.o.Pln(hs.Values());
        S.o.Pln("Key and values");
        S.o.Pln(hs.entrySet());
        S.o.Pln("Printing Map");
        S.o.Pln(hs);
    }
}
```

TreeMap

It is a concrete implementing class of Map Interface.

Characteristics :

1. It also stores data in key = value pairs.
2. It will sort the entries in the map with respect to keys in ascending order.
3. The key in TreeMap must be Comparable type. If it is not Comparable type we get ClassCastException.
4. In TreeMap key cannot be null, If it is null we get NullPointerException.
5. A value in TreeMap can be null.

Ex :

```
import java.util.TreeMap;
Public class T1 2
    Public static void main (String [] args) {
        TreeMap ts = new TreeMap();
        ts.Put (10, "Java");
        ts.Put (3, "Sql");
        ts.Put (4, "Manual");
        ts.Put (8, "JDBC");
        ts.Put (7, null);
        ts.Put (null, "Selenium"); // [illegible] CTE
        S.o.Pln (ts);
    }
}
```

HashTable:

It is a concrete implementing class of Map interface.

Characteristics:

1. It is used to store data in Key=Value format.
2. In hashtable the insertion order is not maintained.
3. In hashtable both Key and Value cannot be Null. If it is null we get NullPointerException.

Example Programs

1. To convert ArrayList to HashSet

```
ArrayList ls = new ArrayList();
ls.add(10);
ls.add(10);
ls.add(20);
ls.add(10);
ls.add(30);
Set s = new HashSet(ls);
S.o.pln(s);
```

2. Map to Set

With the help of entrySet() method of Map interface

```
Map m1 = new HashTable();
m1.put(1,"hello");
m1.put(2,"bye");
S.o.pln(m1); // {1=hello, 2=bye}
```

```
// Map to set
Set s1 = m1.entrySet();
s.o.println(s1); // [1=hello, 2=bye]
```

To convert A collection into an Array

We can convert a collection into an Array with the help of toArray() method present in collection interface.

The return type of toArray() method is Object[]

Ex:

Design a method to convert an Array into an ArrayList.

input : array
task : convert array to ArrayList
return : ArrayList

```
static ArrayList toList(int[] arr)
{
  ArrayList ls = new ArrayList();
  for(int j=0; j<arr.length; j++)
  {
    ls.add(arr[j]);
  }
  return ls;
}
```