Module - 4

⇒ Object class:

out of multiple predefine classes one of the classes name is object class.

It is default super class to all predefine and user define classes of java

object class is present in java.lang package.

** what is superanost class of java?

A. object class

javap java.lang.Object //javap packagename.classname

** List method of object?

Method of object class:

1. getclass():class - - - - - - → final
2. object1.equals(object2) : boolean
3. *** toString(): String

Array/string/Collection of framework / maps

4. hashcode : int
5. notify():void - - - - - → final
6. notifyAll():void - - - - → final
7. wait():void - - - - - → final
8. wait(long timeout): void - - - → final
9. wait(long timeout, int nanos): void - - → final

→ multithreading

10. finalise() - - - - - multithreading
11. clone()

```
class A
{
}
```

```
class A extends Object
{
    //methods || virtually present
}
```

Compiler + JVM

class A Contain what? ⟶ total || methods

Scanned with CamScanner

```
class Sample extends Object
{
  public static void main(String args[])
  {
    //through subclass object i call to method of superclass
     Sample s1 = new Sample();
     System.out.println(s1.toString());
     Sample s2 = new Sample();
     System.out.println(s2.toString());
    // another way to call toString() is by printing reference var
     Sample s3 = new Sample();
     System.out.println(s3); // ⟶ s3.toString()
  }
}
```

output:-
Sample@383534aa
Sample@6bc168e5
Sample@7b3300e5

*** toString() :-

* It is method of object class. It Returns Complete information of an object
* Complete Information Consists of packagename.classname@objectaddress

  Ex:- mypackage.Sample@22h33453

Syntax:-

```
public String toString()
{
  //return packagename.classname@Objectaddress
}
```

* object address is unique address given to Every object
* it Cannot be same for 2 objects
* whenever we print reference variable internally it Calls toString().

Scanned with CamScanner

Program:-

```
class Student extends Object
{
  String sname;
  //Constructor
  public Student(String sname)
  {
    this.sname= sname;
  }
  //@override toString()
  public String toString()
  {
    return "Studentname:"+sname;
  }
  public static void main(String args[])
  {
    Student s1 = new Student("Rohan")
    System.out.println(s1.toString());
  }
}
```

output:-

Student name: Rohan

Scanned with CamScanner

*** Exception Handling :

Exception :-

An exception is an unwanted or unexpected Condition which disturbs our normal flow of excution

Ex:- 1. CoronaException
2. LockDownException
3. BatteryDeadException

* once Exception occured remaining part of program will not be excuted

* An exception is an

```
class A
{
    statement1;
    statement2;
    statement3;
    statement4;
    statement5; (unexpected statement
    statement6;
    statement7;
    statement8;
    statement9;
    statement10;
}
```

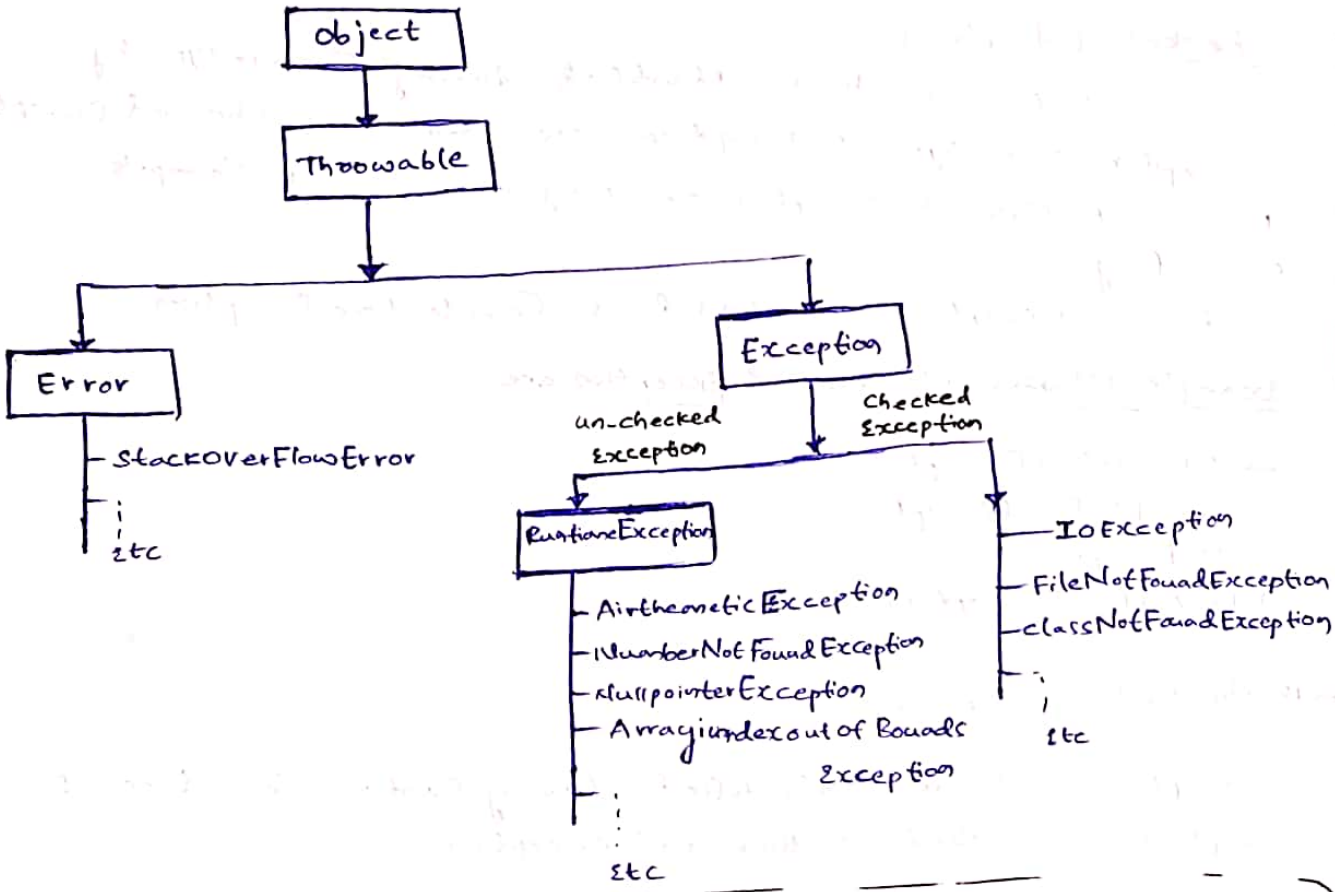
excution

exception Handle

exception

unexcuted

Scanned with CamScanner



* object class is a superanost class to all class.
* Throwable class is a superclass to Error class and Exception class
* RuntimeException is subclass to Exception class

* Exception Hierarchy

object class is a superclass to all the predefine and userdefine class of java

Throwable class is a superclass to "Exception" class and "Error" class and it is present in java.lang package.

Exception class is a superclass to RuntimeExceptionclass and other Exception classes.

Depending on Hierarchy Exceptions are divided into 2 types

1. checked Exception
2. unchecked Exception

Scanned with CamScanner

Checked Exception:-

Exception which are Identified during Compile type by Compiler such type of exceptions are called as Checked Exceptions

In case of Checked exception our program will not Compile successfully

Checked Exceptions are called as Compile time Exception

Examples (Classes) of checked Exception are

→ InterruptedException
→ ClassNotFoundException
→ SQLException
→ FileNotFoundException

⇒ Unchecked Exception :-

Exception which are Identified during Runtime such time of exception are called as Unchecked Exception

Incase of Unchecked Exception our program will atleast Compiles successfully

Unchecked Exception are also called as runtime Exception

RuntimeExceptionClass is a superclass to all UncheckedException classes

Examples (classes) of unchecked Exception are:-

→ ArithmeticException
→ ArrayIndexOutOfBoundsException (Array prog)
→ NullPointerException (String prog)
→ StringIndexOutOfBoundsException (String prog)
→ ClassCastException (Downcasting)
→ InputMismatchException (Scanner)

Scanned with CamScanner

```
class Demo
{
  public static void main(String args[])
  {
    int a=10, b=0, c;
    c = a/b;
  }
}
```

name : AE
DES: /by zero
Location:
Demo.main

JVM — programmer! — Handle (try/catch)
yes

In above program excention begins from main method there is an unexpected statement c = 10/0 when this statement is encountered JVM creates an Exception object which includes

Name:
description:
location:

and handover it to JVM Now JVM will checks is there any exception Handling Code present in main()

Exception handling Code present in main() also JVM will calls to default

exception handler printStackTrace and that default exception handler provides the Complete information of Exception

Default Exception handler printStackTrace() is a method throwable class

display our exception information line name: description and location

Exception in thread "main" java.lang.ArithmeticException: /by zero
at Exceptionprograms.Demo.main(Demo.java:11)

* once exception occured Remaining part of the program will not be excented
* So if we want the above program to be excented we have to handle the Exception
* Exception Can be handled by using try and Catch block

Scanned with CamScanner

```
class Demo
{
  public static void main(String args[])
  {
    int a=10, b=0, c;
    System.out.println("main starts");
    c = a/b
    System.out.println("main ends");
  }
}
```

output:- main starts
Exception in thread "main" java.lang.ArithmeticException: / by zero
at Demo.main(Demo.java:7)

* try and catch blocks:-

```
try
{
  //risky statements which causes exception
}
catch(Exceptionname refvar)
{
  //Alternate solution
}
```

* try block is used to keep a code which causes an exception
* once Exception occured in try block remaining part of code will not be excuted from try block
  so we should keep only statements which cause exceptions under try block.
* once Exception occured in try block JVM immediately make a search for corresponding catch block
* catch block is where we will caught the exception
* under catch block we provide some statements which works as alternate solution for Exception

Scanned with CamScanner

```
class Demo
{
  public static void main(String args[])
  {
    int a=10, b=0, c;
    System.out.println("main starts");
    try
    {
      System.out.println("try starts");
      c = a/b
      System.out.println("try ends");
    }
    catch(Exception e) //Exception e = new ArithmeticException()
    {
      System.out.println("this is catch block");
      System.out.println(e.toString());
    }
    System.out.println("main ends");
  }
}
```

output:-
main starts
try starts
this is catch block
java.lang.ArithmeticException:/by zero
main ends

Rules:-

1. try block is followed by either catch block or finally block
2. between try and catch block there should not be any statement
3. For one try block we can have multiple Catch blocks but sequence should be most specific catch block to most general Catch block
4. It is better to create seperate try and Catch block for every excention
5. when we dont know Exception type we can mention it as ExceptionClass type because Exception is a super class to all the class and during upcasting we studied that superclass can hold reference of subclass object

   ex:- Exception e = new ArithmeticException

Scanned with CamScanner

```
class Demo
{
 public static void main(String args[])
 {
   int a=10,b=0,c;
   System.out.println("main starts");
   try
   {
     c = a/b;
   }
   catch(ArithmeticException e)
   {
     System.out.println(e.toString());
   }
   catch(Exception e)
   {
     System.out.println(e.toString())
   }
 }
}
```

output:-

main starts
java.lang.ArithmeticException: / by zero

```
class Demo
{
  public static void main(String args[])
  {
    int a=10,b=10,c;
    System.out.println("main starts");
    try
    {
      System.out.println("try starts");
      c=a/b;
      System.out.println("try ends");
    }
    catch(Exception e) //Exception e = new ArithmeticException()
    {
      System.out.println("this is catch block");
      System.out.println(e.toString())
    }
    System.out.println("main ends");
  }
}
```

output:-

main starts
try starts
try ends
main ends

Scanned with CamScanner

* finally block :-

finally is a block of Code which will get executed irrespective of

Case:-1 exception did not occured

```
int a=10, b=10, c;
try
{
  c = a/b;
}
catch(Exception e)
{
  System.out.println("catch block");
}
finally
{
  System.out.println("finally block");
}
```

Case:-2 exception occured and handled

```
int a=10, b=0, c;
try
{
  c = a/b;
}
catch(ArithmeticException e)
{
}
finally
{
  System.out.println("finally block");
}
```

Scanned with CamScanner

Case:-3 Exception occured and not handled

```
int a=10, b=0, c;
try
{
  c = a/b;
}
finally
{
  System.out.println("finally block");
}
```

Basically it used to keep an important code which should not skip like closing of data base connection or closing of opened file etc.

following combination are valid for try, catch and finally block

1.
```
try
{
}
catch()
{
}
```

2.
```
try
{
}
finally
{
}
```

3.
```
try
{
}
catch()
{
}
finally
{
}
```

throws keyword :-

It is a keyword which is used to transfer the exception control to caller

Syntax:-

MethodHeader MethodSignature throws Exceptionname

Ex:- public void a() throws ArithemeticException

Ex:- you are going and registering for Interview - try/catch

* you are not going rather u friend to register on behalf of you - throws

Scanned with CamScanner

```
Class Sample
{
  public static void main(String args[])
  {
    try
    {
      calculate();
    }
    catch(AirthemeticException e)
    {
      System.out.println("catch block");
    }
  }
  public static void calculate() throws AirthemeticException
  {
    int a=10, b=0, c;
    c=a/b
  }
}
```

output:-
Catch block

throw Keyword:-

It is used to create an exception object explicitly based on programer choice, Exception object can be created

Syntax:- throw new Exceptionname(description)

```
class Vote
{
  public static void validate(int age)
  {
    if(age<18)
    {
      throws new AirthemeticException("under Eligibility");
    }
    else
    {
      System.out.println("u can vote");
    }
  }
  public static void main(String args[])
  {
    Validate(17)
  }
}
```

output:-
Exception in thread "main" java.lang.AirthemeticException: under Eligibility age
at Vote.Validate(sample.java:52)
at Vote.main(Sample.java:61

Scanned with CamScanner

try - - - - - risky statement
catch - - - - → caught the exception by keeping an alternate solution
finally - - - - → important code that to be executed surely
throw - - - - → create an exception object explicitly
throws - - - - → To transfer an exception

Differences between THROW AND Throws

THROW	THROWS
→ Throw keyword is used to create Exception object explicitly	It is used to transfer the excution
→ Throw keyword is used inside method	Throws keyword is used with method declaration
Ex:- throw new Exceptionname[Expdescription] throw new ArithmeticException("exception")	Ex:- method declaration throws Exceptionname public void fly() throws InterruptedException
→ Throw keyword is mainly used for userdefine exception	Throws keyword is mainly used for checked Exception
→ using throw keyword we can create only one exception object at a time	→ using throws keyword we can transfer multiple exception at a time
throw new MyBalException	pv checked() throws interruptedException (SQL Exception)

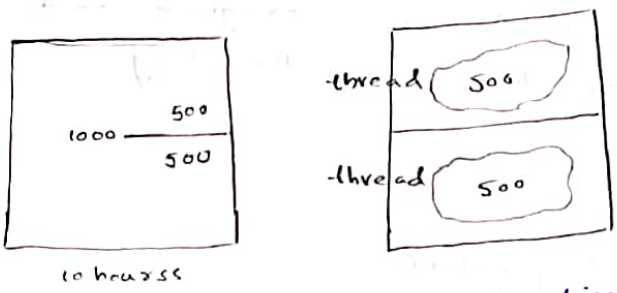
Scanned with CamScanner

⇒ Multi threading :-

multitasking:- performing more than one task parallely at a time

Thread based multitasking:-

The process of executing more than one task parallely where each task is independent part of same program



→ Excution time is more
→ Effiecency is less
→ cpu utilisation is less

→ excution time is less
→ effiecency is more
→ cpu utilisation is more

program:-

```
class Sample extends Thread
{
  public void run()
  {
    //task of any thread
    for(int i=1; i<=3; i++)
    {
      System.out.println("This is any thread");
    }
  }
}
public class User_T
{
  public static void main(String args[])
  {
    Sample s1 = new Sample();
    s1.start();
    //task for main thread
    for(int i=1; i<=3; i++)
    {
      System.out.println("This is main thread");
```

main thread

Go to Sample I create method through once it available

JVM → Thread scheduler

thread, main

output:-
This is main thread
This is any thread
This is any thread
This is any thread
This is main thread
This is main thread

```
    }
  }
}
```

Thread is defined as a small part of our program

or

Thread is defined as light weight process

or

Thread is defined as flow of excution

→ process of creating multiple threads in a program and excuting them parallely is called as Multithreading

Thread can be created in 2 ways:-

pre requsite:- inheritance, overriding, interface

1. By extending thread class
2. By extending Runnable interface

1. By extending Thread class:-

→ create our class which should extends Thread class

→ Thread is pre define class of java which is present in java.lang package

→ override run() which is predefine method of Thread class because run() indicates job of our thread

public void run()

→ In order to make our thread available for excution make a call to start()

→ start is a predefine method of Thread class

public synchronized void start()

Scanned with CamScanner

```
class Sample extends Thread
{
    public void run() //overriding of run()
    {
        //task of any thread
        for(int i=1; i<=3; i++) //job of any thread
        {
            System.out.println("This is any thread");
        }
    }
}
public class User
{
    public static void main(String args[])
    {
        Sample s1 = new Sample();
        s1.start();
        //task of for main thread
        for(int i=1; i<=3; i++) //job of any thread
        {
            System.out.println("This is main thread");
        }
    }
}
```

Implementing Runnable Interface

Runnable is a predefine Interface which is present in java.lang package

```
public interface java.lang.Runnable
{
    public abstract void run();
}
```

Scanned with CamScanner

```
class Sample implements Runnable
{
  public void run() // overriding of run()
  {
    //task of any thread
    for(int i=1; i<=3; i++) // job of any thread
    {
      System.out.println("This is any sample thread");
    }
  }
}
public class User_T
{
  public static void main(String args[])
  {
    Sample s1 = new Sample();
    Thread t1 = new Thread(s1);
    t1.start(); // ——→ it will call to run() of Sample class
    // task for main thread
    for(int i=1; i<=3; i++) // job of any thread
    {
      System.out.println("This is main thread");
    }
  }
}
```

Internal Implementation:-

Runnable is a predefine interface which is present in java.lang package

```
public interface java.lang.package
```

```
Interface Runnable
{
  public abstract void run()
}
class Thread implements Runnable
{
```

Scanned with CamScanner

```
    public void run()
    {

    }
}
```

out of 2 ways preferred by runnable interface

class A extends Student implements Runnable ——→ possible

class A extends Thread, Student — — → not possible

A thread can be always be in any of one stage

1. new ——— → it will be new state when you create object of our thread class

2. Ready/Runnable —→ it is in Ready state when start() get called

3. Running — — — → it is in running when thread scheduler start Executing

4. Blocker — — — → it is in Blocked state if we call to sleep() or wait()

5. Terminated — —→ when run() is Completed

Scanned with CamScanner

primitive data type:-

8 primitive data types

byte, short, int, long, float, double, long, boolean, char

int a = 100;

At a time I can store only one value into this variable

what if I wanted to store group of value

Arrays:

→ An Array is used to store collection of/group of homogenous (similar) data stored in continuous manner

→ we will use array when we are working with group of values

→ An Array is represented with []

→ In Java an array is created as an object where every value is stored with one index position starting from 0

Steps for using Array

1. create an array
2. store the value into an array
3. print array values using index position

1. create an array:-

Syntax:→ arraytype arrayname[] = new arraytype[size];

(or)

→ arraytype[] arrayname = new arraytype[size];

(or)

→ arraytype []arrayname = new arraytype[size];

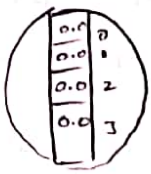
→ An array cannot be infinite, every array is some finite size

→ Array size cannot be negative or decimal

Scanned with CamScanner

Ex:- int a[] = new int[3] — a

double [] d = new double[4]; — d



String[] s = new String[3]; - - S - →

→ when we created an array and we didnot store any values then default values will be there in array

2. Store the values into an array

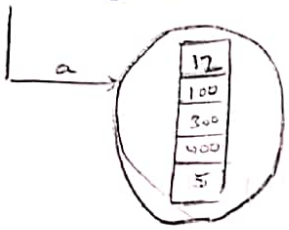
Syntax:- arrayname[indexnum] = value;

a[0] = 100;
a[1] = 200;
a[2] = 300;
a[4] = 500 → ArrayIndexOutofBoundsException

Note:-

When we already knows our array elements we can create an array object directly

int a[] = {12, 100, 300, 400, 5};



3. printing the array values

1. for loop
2. for each loop

length variable:-

It is a predefine variable, if we print length variable with an array refereace we will get size of our array

Ex:- String s[] = {"java", "Adv java", "j2ee", "java selenium"};

Sop(s.length); - - → 4

Scanned with CamScanner

Ex:-

```
char []c = new char[3]
Sop (c.length); ---> 3
```

```
public class Student
{
    public static void main(String args[])
    {
        String names[] = new String[5];

        names[0] = "Mr John";
        names[1] = "Mr Rahul";
        names[2] = "Mr Sohan";
        names[3] = "Mr. Rohan";
        names[4] = "Mr Vishal";

        for(int i=0; i<names.length; i++)
        {
            System.out.println(names[i]);
        }
    }
}
```

Starting index

⇒ output:-

```
Mr John
Mr Rahul
Mr Sohan
Mr Rohan
Mr Vishal
```

program:-

1. write a program to create an Integer array and display

```
public class Interarray
{
    public static void main(String args[])
    {
        int a[] = new int[5];

        a[0] = 10;
        a[1] = 20;
        a[2] = 30;
        a[3] = 40;
        a[4] = 50;

        for(int i=0; i<a.length; i++)
        {
            System.out.println(a[i]);
        }
    }
}
```

output:-

```
10
20
30
40
50
```

Scanned with CamScanner

(2) write a program to create an Integer ~~ARRAY~~ of size 7 and do the following

a. sum of all numbers of an array

b. average of all numbers of an

```
public class Array1
{
  public static void main(String args[])
  {
    int a[] = {10, 15, 20, 25, 30, 35}
    int sum=0;
    for(int i=0; i<a.length; i++)
    {
      sum = sum + a[i]; // 0 + a[0] + a[1] + a[2] + a[3] + a[4] + a[5]
    }
    System.out.println(sum);
    float average;
    average = sum/a.length;
    System.out.println(average);
  }
}
```

⇒ output:-
135
22.0

(3) write a program to print Even number from array

```
import java.util.Scanner
public class Array
{
  public static void main(String args[])
  {
    Scanner s = new Scanner[System.in];
    int n;
    System.out.println("Enter the size of an array:");
    n = s.nextInt();
    // create an array
    int a[] = new int[n];
    // for asking array values
    for(int i=0; i<n; i++)
    {
      a[i] = s.nextInt();
    }
```

Scanned with CamScanner

```
System.out.println("---- Even number from array are :- ---");
// i need to check all array number from 0 to last index
   ---> i - index , a[i] --> value at ith index from array
for(int i=0; i<n; i++)
{
    // Condition
    if(a[i] % 2 == 0)   // if (i % 2 == 0)
    System.out.println(a[i]);
}
}
}
```

output:-

```
Enter the size of an array
5
25
12
22
---- Even number from array are -
12
22
```

(4) Write a program to print even position number of an array?

```
import statement we should write
public class Array1
{
  public static void main(String args[])
  {
    Scanner s = new Scanner(System.in)
    int n;
    System.out.println("Enter the size of an array:");
    n = s.nextInt();
    //create an array
    int a[] = new int[n];
    // for asking array values
    for(int i=0; i<n; i++)
    {
      a[i] = s.nextInt()
    }
    System.out.println("---- Even index position number
                                from array are: ");
```

Scanned with CamScanner

// i need to check all the array numbers from 0 to last index --- i-index, a[] → value of ith index from array

```
for(int i=0; i<n; i++)
{
    // Condition
    if(i%2==0)
    System.out.println(a[i]);
}
}
}
```

⑤ Write a program to print maximum or greatest number from an array!

```
public class Array1
{
    public static void main(String args[])
    {
        int a[] = {22, -11, 85, 7, 4};
    // logic : will assume 1st num as maximum and Compare
with Remaining num in an array

        int max = a[0];
        for(int i=1; i<a.length; i++)
        {
            if(a[i] > max)
            {
                max = a[i];
            }
        }
        System.out.println("The maximum value of an array is:"
                                +max);
    }
}
```

output:-

The maximum value of an array is:
85

Scanned with CamScanner

⑥ write a program to print minimum number from an array

```
import java.util.Scanner
public class Array1
{
  public static void main(String args[])
  {
    Scanner s = new Scanner(System.in)
    int n;
    System.out.println("Enter size of an array:");
    n = s.nextInt();
    //create an array
    int a[] = new int[n];
    //for asking array values
    for(int i=0; i<n; i++)
    {
      a[i] = s.nextInt();
    }
    //logic i will assume 1st num as minimum and compare
    with Remaining num in an array
    int min=a[0];
    for(int i=1; i<a.length; i++)
    {
      if(a[i] < min)
      {
        min = a[i];
      }
    }
    System.out.println("The minimum value of an array
                        is :" + min);
  }
}
```

* output:-

```
Enter size of an array
5
88
98
65
-7
The minimum value of an
array is: -7
```

Scanned with CamScanner

(8) write a program to print second highest number from an array

```
public class Arrays
{
  public static void main(String args[])
  {
    int a[] = {11, 25, 89, 6, 78};
    int max = a[0];
    int secmax = a[0];
    for(int i=1; i<a.length; i++)
    {
      if(a[i] > max)
      {
        secmax = max;
        max = a[i];
      }
      else if(a[i] > secmax)
      {
        secmax = a[i];
      }
    }
    System.out.println("The maximum value of an array is :" +max);
    System.out.println("The second maximum value of an array is :" +secmax);
  }
}
```

output:-

The max value of an array is 89
The second max value of an array is 78

(9) write a program to print second minimum value of an array

```
public class Array
{
  public static void main(String args[])
  {
    int a[] = {11, 25, 89, 6, 78}
    int min = a[0];
    int secmin = a[0];
    for(int i=1; i<a.length; i++)
    {
      if(a[i] < min)
      {
        secmin = min;
        min = a[i];
      }
      else if(a[i] < secmin)
      {
        secmin = a[i];
      }
    }
    System.out.println("The minimum value of an array is :" +min);
    System.out.println("The second minimum value of an array is :" +secmin);
  }
}
```

output:-

6
11

Scanned with CamScanner

(10) WAP to Sort(arrange) and array in increasing order

{ 11, 8, 95, 162, 4 }

take 2 for loops

1. for loop for taking 0 to a.length

2. for loop for taking next nber for comparing

```
public class Array
{
  public static void main(String args[])
  {
    int a[] = {11, 8, 95, 162, 4};
    int temp;
    for(int i=0; i<a.length; i++)
    {
      for(int j=0; j<a.length-1; j++)
      {
        if(a[j] > a[j+1])
        {
          temp = a[j];
          a[j] = a[j+1];
          a[j+1] = temp;
        }
      }
```

```
//for loop for printing array
for(int k=0; k<a.length; k++)
{
  System.out.println(a[k]);
}
}
}
```

output:-

4
8
11
95
162

(11) WAP to Sort arrange and array in descending order

```
public class Array
{
  public static void main(String args[])
  {
    int a[] = {11, 8, 95, 162, 4};
    int temp;
    for(int i=0; i<a.length; i++)
    {
      for(int j=0; j<a.length-1; j++
      {
        if [a[j] < a[j+1]
        {
          temp = a[j];
          a[j] = a[j+1];
          a[j+1] = temp;
        }
      }
    }
```

Scanned with CamScanner

```
//for loop for printing array
for(int k=0; k<a.length; k++)
{
    System.out.println(a[k]);
}
}
}
```

output

Command Line Arguments:-

String args[]

arraytype arrayname[]

main() is containing an argument as array of String type while executing we can pass any type of values and those values will be stored in array args

```
public class CMD
{
    public static void main(String args[])
    {
        System.out.println(a[0]);
    }
}
```

output:-

```
java CMD 25
25
java CMD java
java
```

Scanned with CamScanner

Q. Is java an 100% object oriented language?

A. No

String — object
Exception — object
array — object
Collection → object
thread → object
Primitive datatype → memory block X

It is nearly 100% object oriented language because PDT doesnot Relate to object

In order to represent these PDT as object we use wrapper classes.

Wrapper Classes:-

These are predefined classes which are present in java.lang package

for every primitive datatype, we have Corresponding wrapper class

datatype	Wrapperclass
byte	Byte
boolean	Boolean
char	Character
short	Short
float	Float
int	Integer
double	Double
long	Long

long → Memory block
long → object

* Every wrapper class is a final class so we cannot Inherit it than

* If you want to see what these classes Contain

cmd: javap packagename.classname

ex: javap java.lang.Integer

Scanned with CamScanner

The major application of these wrapper classes in Collection of framework Concepts

Note :-

byte b = 85; — 85 (1 byte)

byte b1 = new Byte(85); (or) Byte b1 = 85;

85

Using method of wrapper class we can perform Boxing and unboxing

⇒ Boxing:- Converting primitive datatype as an object is called as boxing

Boxing can be performed by using valueOf()

Ex:- int i = 35000;

Integer obj = Integer.valueOf(i); // boxing

public static java.lang.Integer valueOf(int);

unboxing:- Converting object type as a primitive type is called unboxing

* unBoxing can be performed by xxxValue() (xxx → primitive datatype)

Ex:- Integer i = new Integer(100) or Integer i = 100

Int pt = i.intValue();

for storing data:-

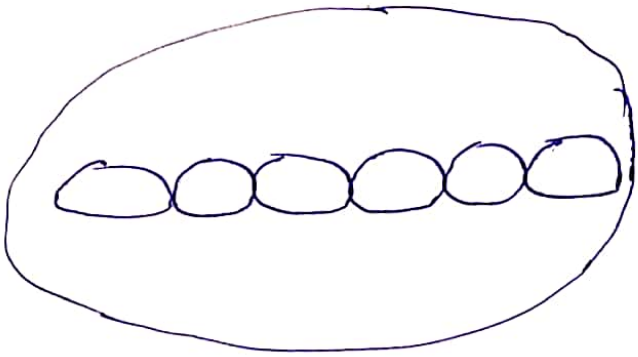
* Primitive datatype Variables — [ ] — of specific size — only one value

* Arrays — → only homogeneous type
→ finite size
→ every thing is value
→ code for every operator

Scanned with CamScanner

Collections

→ both homo + heterogeneous
→ no size limit
→ Every thing is objects (100%)
→ Every operators predefine methods

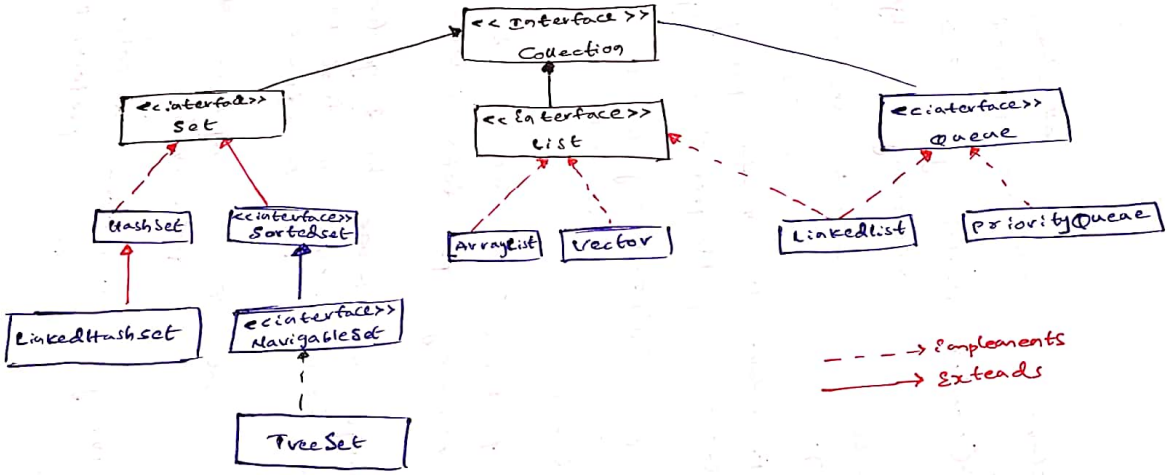


	Arrays	Collections
1.	Arrays are used to store	Collection is used to store
2.	Collection of homogenous (similar) data	2. Heterogeneous (different) as well as Homogenous objects
3.	Array are finite in size int a[] = new int[3];	3. Collection are numerous (growable in size)
4.	Arrays can deal with primitive as well as wrapper classes	4. Collections purely deals with Wrapper classes (objects)
5.	Array doesnot have any underlying data structure (physical Structure of storing data)	5. Every classes of Collection have datastructure
6.	Array doesnot Contain predefine method (add, sorting, removing replacing)	6. In Collection 80% support is pre-define Interface
7	we Should use arrays when we already knows the data in advance	7. we should go for Collection when we dont know the data in advance
8.	Memory wise array is not preferred	8. memory wise Collection is preferred
9.	performance wise arrays are prefferred	9. performance wise Collection is not preferred

Scanned with CamScanner

Collection:- Collection is nothing but a group of objects represents as Single object. Its main purpose is to store huge amount of homogenous as well as heterogenous data.

Collection Interface



Scanned with CamScanner

Collection Framework :-

It is an architecture or template

It provides the group of classes and Inheritance which is used to store multiple object as Single object

CollectionFramework ⟶ Concept

Collection ⟶ Root interface

All classes and Interface of CollectionFramework are present in java.util.package

Collection Hierarchy :-

Collection is the root Interface of CollectionFramework

Collection Interface is extended by 3 different interfaces

1. List < interface >
2. Set < interface >
3. Queue < interface >

Learning about Collection mean learning about classes and it Characteristic

Different class of Collection are :-

1. ArrayList
2. LinkedList
3. Vector

⟶ Implement List Interface

4. priority Queue
5. LinkedList

⟶ Implements Queue Interface

6. Tree Set
7. HashSet
8. LinkedSet

⟶ Implements SetInterface

Scanned with CamScanner

Charateristics to study for every class:-

- Version
- homogenous or heterogenous
- Null object is possible or not
- duplicate allowed or not
- Insertion order is preserve or not
- data structure
- Cursors.

Useful Methods:-

* add(object): used to add objects
* addAll(Collection): used to Copy one Collection objects into another
* remove(Object): used to Removes object
* remove(Index): used to remove the object at given index
* set(Index, Object): it Replaces the old object with new object at given Index
* get(index): Retrives the object at given index
* isEmpty(): check whether Collections empty or not
* Contains(object): checks whether object is present or not
* Size(): provides size of Collections (always calculates from 1)
* Collections.sort(): Sort the Collection in ascending order (In case of char's as per unicode order)
* Collections.reverse(): Sort in reverse way:

List Interface:-

* List is an interface which extends Collection interface

List is implemented by 3 different class

1. ArrayList
2. LinkedList
3. Vector(class Vector extends Stack implements List)

List follows Index Based process

Scanned with CamScanner

ArrayList:-

ArrayList is a class which implements list Interface

Charateristics of ArrayList:-

* It is Introduced in 1.2V
* ArrayList stores Heterogenous data
* It is possible to add Null object in Arraylist
* It allow duplicate objects
* In arraylist Insertion order is preserved the way we added objects the way it will be printed
* It follows data structure as growable size array
* Iterator and ListIterator Cursor are used

Array are Typesafe but Collection are not safe

```
Ex:- int a[] = new int[3];
        a[0] = 100;
        a[1] = "java"; // CTE
Ex:- Arraylist a = new Arraylist()
        a.add("java");
        a.add(100);
        a.add('A');
```

program:-

```
package List.In
import java.util.ArrayList;

class List_1
{
  public static void main(String args[])
  {
    ArrayList l1 = new Arraylist();
    l1.add("core java"); //String
    Integer i1 = 500;
    l1.add(i1);   //int   //Integer
```

Scanned with CamScanner

```
l1.add(true);   // boolean   // Boolean
l1.add(785.25f); // float   // Float
l1.add('R');  // char   // character
l1.add(null);  // null
System.out.println(l1.toString); // print Content of object
    l1.remove(true);
    l1.remove(1); // l1.remove(object at 1st index)
System.out.println("After removing:" + l1); l1.toString()
System.out.println("object at 3rd index:" + l1.get(3));
System.out.println("Number of objects:" + l1.size());
    l1.add(2, "Core java"); // adding at 2nd index
    System.out.println("After adding at 2nd position:" + l1);
    l1.set(1, "python"); //
    System.out.println("After replacing:" + l1);
    System.out.println("--- Array List objects are ----");
    // for loop
    for(int i=0; i<l1.size(); i++
    {
      System.out.println(l1.get(i));
    }
  }
}
```

output:-

[core java, 500, true, 785.25, R, null]

After removing: [core java, 785.25, R, null]

object at 3rd index: null

Number of objects: 4

After adding at 2nd position: [Core java, 785.25, Corejava, R, null]

After Replacing: [Core java, python, Core java, R, null]

--- Array List objects are ---

Core java

python

Core java :

R :

null :

Generics:-

Since Collections are not safe i.e programmer can added any type of objects there is no restrictions for that, so to overcome this drawback java has given Generics Concept where u can create an object of specific type, if we add any other type of object than specified we will get CTE

```
import java.util.*;
public class List_2
{
  public static void main(String args[])
  {
    ArrayList<String> l2 = new ArrayList<>();
    l2.add("java");  //l2.add(500);
    ArrayList<Integer> l21 = new ArrayList<>();
    //l21.add("java");
    l21.add(500);
    ArrayList<Float> l22 = new ArrayList<>();
    //l22.add("java");
    l22.add(500.78f);
    ArrayList<Character> l8 = new ArrayList<>();
    l8.add('A')
    //l8.add(500);
  ArrayList<Object> l = new ArrayList<>();
    l.add("java");
    l.add('A');
    l.add(500.89f);
    l.add(true);
```

Scanned with CamScanner

```
    System.out.println(');
  }
}
```

output:-

[Java, A, 500.89, true]

⇒ **Cursors (movable object)**

This is the special characteristic given for Collection Concepts

using Cursors we can retrieve the objects from Collection one by one from left to right or right to left

Cursors is actually an Interface

2 types:-

1. Iterator
2. ListIterator
3. Enumerator (outdated)

1. **Iterator:-**

Iterator is an Interface which is used to traverse (retrieve, print) the Collection in forward direction)

Basically it provides the prevailage to access the objects without using index

```
Interface Iterator
{
  public boolean hasNext();
  public object next();
  public void remove();
}
```

⇒ * hasNext():→ It returns true if there is object available

⇒ * next():- It return Current object and move the Cursor to next object

Scanned with CamScanner

Since we cannot create an object of Iterator interface Collection framework gave

iterator() = new Iterator()

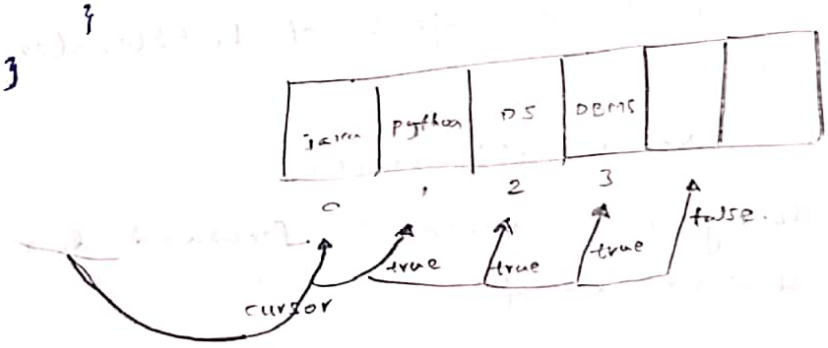
iterator() provides us an object of Iterator interface

```
import java.util.ArrayList;
import java.util.Iterator;

class Cursors
{
public static void main(String args[])
{
ArrayList<String> a1 = new ArrayList<>();
    a1.add("java");
    a1.add("python");
    a1.add("Data structures");
    a1.add("DBMS");
    //toString() and ref var and for loop
    // cursors
    // Iterator i1 = new Iterator();
    // Iterator i1 = new Iterator();
    Iterator i1 = a1.iterator();
    while(i1.hasNext()==true)
    {
        System.out.println(i1.next());
    }
}
}
```

output:-

```
java
python
Data structures
DBMS
```



Scanned with CamScanner

List Iterator:

List Iterator is an interface which provides the facility to traverse the list in forward as well as backward direction

```
interface ListIterator
{
    public boolean hasNext();
    public object next();
    public void remove();
    public boolean hasPrevious();
    public object previous();
    public void add();
}
```

hasPrevious:-

Return true if object is available to iterate from previous direction

previous():-

points the current object and move cursor to next object in previous direction

hasNext():-

It Returns true if there is object available in forward direction

next():-

it returns CurrentObject and move the cursor to next object

Since we cannot create an object of ListIterator interface they give as

ListIterator() ---> new ListIterator()

Note:- It is compulsory to traverse in forward direction first then in backward direction

Scanned with CamScanner

```
import java.util.ArrayList
import java.util.ListIterator;

class Cursors
{
  public static void main(String args[])
  {
    ArrayList<String> a1 = new ArrayList<>();
      a1.add("java");
      a1.add("python");
      a1.add(null);
      a1.add("Data structure");
      a1.add("DBMS");

      //toString() and ref var and for loop
      // cursors
      ListIterator<String> i1 = a1.listIterator();
      System.out.println(".... forward direction ----");
      while(i1.hasNext() == true)
      {
        System.out.println(i1.next());
      }
      System.out.println("---- Backward direction ----");
      while(i1.hasPrevious() == true)
      {
        System.out.println(i1.previous());
      }

  }
}
```

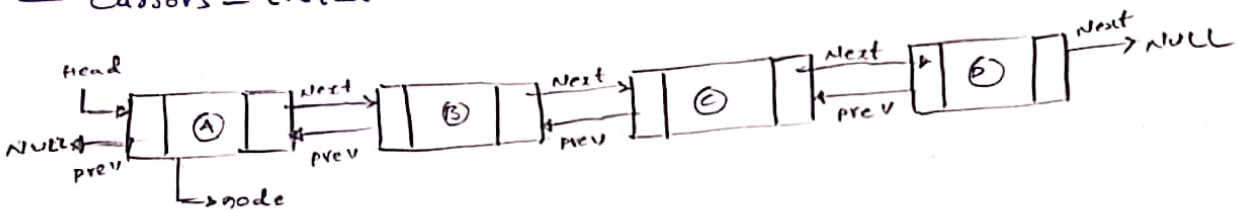
*

```
---- Forward direction ---
Java
python
null
Data structures
DBMS
---- Backward direction ---
DBMS
Data structures
null
python
Java
```

Scanned with CamScanner

* LinkedList :-

- Intro in 1.2V
- default capacity of linked list 0
- Heterogeneous object are allowed
- Null object are allowed
- Insertion order is preserved
- duplicate objects are allowed
- data structure is doubley linked list
- Cursors – ListIterator and Iterator




```
import java.util.LinkedList;
import java.util.ListIterator;
Class Cursors
{
  public static void main(String args[])
  {
    LinkedList<String> a1 = new LinkedList<>();
        a1.add("java");
        a1.add("python");
        a1.add(null);
        a1.add("Data structures");
        a1.add("DBMS");
    //toString() and ref var and for loop
    // cursors
    ListIterator<String> i1 = a1.listIterator();
    System.out.println("--- forward direction ----");
    while(i1.hasNext()==true)
    {
      System.out.println(i1.next());
    }
    System.out.println("--- Backward direction ---");
    while (i1.hasPrevious()==true)
    {
```

Scanned with CamScanner

```
        System.out.println(l1.previous());
    }
  }
}
```

// Same process to write program only in place of arraylist object create object of linked list

--- forward direction

Java
Python
null
Data structures
DBMS

--- Backward direction ---

DBMS
Data structures
null
Python
Java

* Vector :-

Intro in 1.0 v so vector is also referred as legacy class

Default capacity is 10
Heterogenous objects are allowed
Null objects are allowed
Insertion order is preserved
duplicat objects are allowed
data structure is growable array.
cursors - ListIterator and Iterator

// program are same only in place of Arraylist use Vector

legacy classes are classes which is introduced in 1.2 v of java

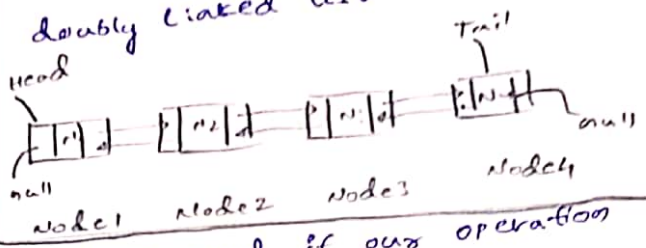
Difference between ArrayList and Vector

ArrayList	Vector
1. Default Capacity of Arraylist is 10	Default Capacity of Vector is 10
2. 1.2 so Non Legacy class	2. 1.0 so Legacy class
3. when it Reaches it saturation point size is increase by (Capacity * 3/2 + 1) ex: $\frac{10 \times 3}{2} + 1 = 16$	3. when it Reaches it saturation point size increases By its equal size ex:- 10 + 10 = 20
4. performance is good	4. performance is poor Compare with arraylist.

Scanned with CamScanner

***

Difference between ArrayList and LinkedList

ArrayList	LinkedList
1. Default size/capacity of ArrayList is 10	1. Default size of LinkedList is 0
2. Its data structure is growable size array Ex:- 0 1 2 3 4 5 6 7 8 9	2. Its data structure is doubly linked list 
3. It is good if our operation is retrieval	3. It is good, if our operation is insertion and deletion
4. While adding data in a middle, ArrayList is slower	4. LinkedList is faster, when we add object in middle
5. ArrayList is faster, when we add objects in the end	5. LinkedList is slower, when we add objects in the end

**

Difference between ArrayList and Array

Array	ArrayList
1. only Homogenous	1. Homo + Heterogenous
2. finite in size	2. growable in size
3. we will use array when we already knows elements	3. we will use arraylist when we donot elements
4. if we add more than give size we will get AIOB Exception	4. if we add more than default size, memory will grow by $\frac{3}{2}+1$
5. Array doesnot have pre define methods supports both primitive and wrapper classes	5. ArrayList has multiple pre define methods support only wrapper classes Arraylist supports iterator as well as list iterator Cursors

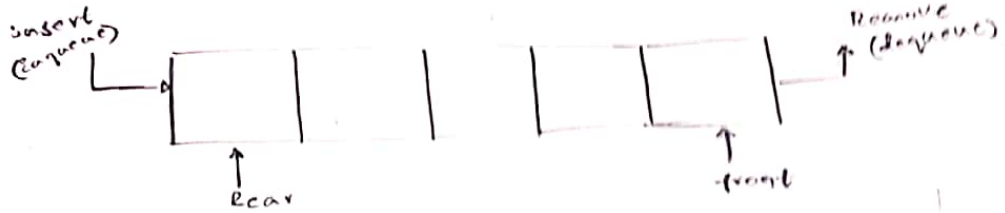
Scanned with CamScanner

Queue Interface :-

Queue is an Interface which is implemented by 2 different classes

1. priorityQueue
2. LinkedList

Queue is a data structure which usually follows FIFO principle (First In First out)



Queue data structure

1. priorityQueue :-

* Introduced in 1.2 version
* Default Capacity is 11
* It allows only homogeneous objects
* Basically data structure of queue is FIFO but in priority
* Queue internal sorting will happen
* Null objects are not allowed
* for retrieving of data, they have given the special methods called as peek() and poll()

peek() :-

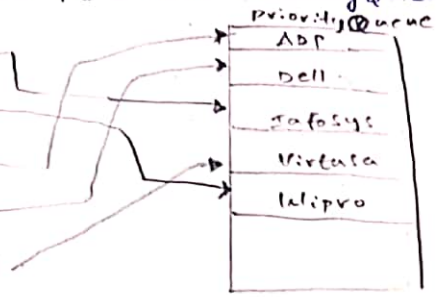
* It retrives the head element (top element) without deleting it from queue
* It Retrieves null if there is no head element present

poll() :-

* It Retrives head element (First element) and delete it from queue

Scanned with CamScanner

```
import java.util.PriorityQueue;
public class Queue1
{
    public static void main(String args[])
    {
        priorityQueue<Object> P1 = new PriorityQueue<>();
        P1.add("Infosys");
        P1.add("Wipro");
        P1.add("ADP");
        P1.add("Dell");
        P1.add("virtusa");
```




```
        System.out.println(P1.peek());
        System.out.println(P1.contains("ADP"));
        System.out.println(P1.poll());
        System.out.println(P1.contains("ADP"));
    }
}
```

output:-
ADP
true
ADP
false

* Null objects are not allowed

If we passed null value Exception:- NullPointerException

```
import java.util.PriorityQueue
public class Queue
{
    public static void main(String args[])
    {
        priorityQueue<object> P1 = new priorityQueue<>();
        P1.add("Infosys");
        P1.add("Dell");
        P1.add("Wipro");
        P1.add("ADP");
        P1.add("Innovax");
        P1.add("Dell");
        P1.add("virtusa");
```

Scanned with CamScanner

```
System.out.println(p1.peek());
System.out.println(p1.Contains("ADP"));
System.out.println(p1.poll());
System.out.println(p1.Contains("ADP"));
System.out.println(p1.poll());
System.out.println(p1.poll());
System.out.println(p1.poll());
System.out.println(p1.poll());
System.out.println(p1.poll());
System.out.println(p1);
}
}
```

output:-

```
ADP
true
ADP
false
Dell
Dell
Infosys
Innovate
Virtusa
[Wipro]
```

2. LinkedList:-

LinkedList is a class which implements List as well as Queue Interface

Class LinkedList implements List, Queue

All characteristics are same as linked list for List Interface

Compared to priority queue:-

Heterogeneous allowed

output in insertion order

null objects allowed

data structure is doubly linked list

```
import java.util.LinkedList;
public class Queue
{
  public static void main(String args[])
  {
    LinkedList<object> p1 = new LinkedList<>();
    p1.add("Infosys");
    p1.add(1235);
    p1.add("Wipro");
    p1.add(null);
    p1.add("Innovat");
```

Scanned with CamScanner

```
P1.add("Dell");
P1.add(1235);
```

* output:-
[Infosys, 1235, Wipro, null, Innovax, Dell, 1235]

```
*/ System.out.println(P1.poll());
System.out.println(P1.poll());
System.out.println(P1.poll());
System.out.println(P1.poll());
System.out.println(P1.poll());
System.out.println(P1.poll());
System.out.println(P1.poll()); /*
System.out.println(P1);
}
}
```

Set Interface:-

set is an interface which is implemented by 3 different classes

1. Hashset
2. Treeset
3. LinkedHashset

1. HashSet:-

It is a class which implements set Interface

features:-

* Introduced in 1.2V
* Heterogenous object are allowed
* Duplicate objects are not allowed In case if I add I wont get Compile time error
  it will just add once
* only one Null object is allowed
* Data structure is hastable
* Default capatity of hashtable is 11
* output is in random order (depend on hashcode number)
* Set is only unidirectional so it supports only Iterator

Scanned with CamScanner

```
import java.util.HashSet;
import java.util.Iterator;
public class Set1.
{
  Public static void main(String args[])
  {
    HashSet<String> s1 = new HashSet<>();
      s1.add("samsung");
      s1.add("Redmi");
      s1.add("Realme");
      s1.add("Oppo");
      s1.add("Lenovo");
      s1.add("oppo");
      s1.add(null);
      s1.add(null);
    // toString, refereace, for loop, Cursor, or for Each loop
    Iterator<String> i1 = s1.iterator();
    while (i1.hasNext() == true)
    {
      System.out.println(i1.next());
    }
  }
}
```

output:-

Lenovo
null
Oppo
Samsung
Realme
Redmi

2. LinkedHashset:-

It is a class which extends Hashset and implements set interface

features:-

→ Introduced in 1.2V
→ Heterogenous objects are allowed
→ Duplicate are not allowed In case if we add we wont get Compile time error it will add once
→ Only one Null Object is allowed
→ Data Structure is Linkedlist

Scanned with CamScanner

* Insertion order is preserved
* set is only uni-directional so it supports only Iterator

```
import java.util.LinkedHashset;
import java.util.Iterator;
public class Set1
{
  public static void main(String args[])
  {
    LinkedHashset<String> s1 = new LinkedHashset<>();
      s1.add("Samsung");
      s1.add("Redmi");
      s1.add("Realme");
      s1.add("oppo");
      s1.add("Lenovo");
      s1.add("oppo");
      s1.add(null);
      s1.add(null);
      //toString, refereace, for loop, cursor or for each loop
      Iterator<String> i1 = s1.iterator();
      while(i1.hasNext()==true)
      {
        System.out.println(i1.next());
      }
  }
}
```

output:-
Samsung
Redmi
Realme
oppo
Lenovo
null

3. Treeset :-

It is a class which implements Set Interface

features:-

* Introduced in 1.2V
* Heterogeneous objects are not allowed if we add we will get class Exception
* Duplicates are not allowed In case if i add i won't get Compile time error
  it will add just once

Scanned with CamScanner

* No null object is allowed
* Data structure is tree
* output in Sorted order
* Set is only uni-directional So it supports only iterator

```
import java.util.Treeset;
import java.util.Iterator;
public class Set1
{
  public Static void main(String args[])
  {
    TreeSet<object> s1 = new Treeset<>();
        s1.add("Samsung");
        s1.add("Redmi");
        s1.add("Realme");
        s1.add("oppo");
        s1.add("Lenovo");
        s1.add("oppo");
      // s1.add(1256); -- classCastException
      // s1.add(null); -- -> NullPointerException
  // toString, reference, for loop, cursor or for each loop
      Iterator<Object> i1 = s1.iterator();
      while(i1.hasNext() == true)
      {
          System.out.println(i1.next());
      }
  }
}
```

* output :-

Lenovo
Oppo
Realme
Redmi
Samsung

Scanned with CamScanner

*CollecctionFramework Complete theory - Notepad

File Edit Format View Help

Features	ArrayList	LinkedList	Vector	PriorityQ	Hashset	LinkedHashset	Treeset
1.Version	1.2	1.2	1.0	1.2	1.2	1.2v	1.2v
2.Heterogeneous (Allowed/not)	allowed	allowed	allowed	not	allowed	allowed	not
3.data structure	growable size array	double linked list	growable array	internal sorting	hastable	linked list	tree
4.duplicate objects (Allowed/not)	allow	allowed	allowed	allowed	allowed	not (add once)	not (add once)
5.null Objects (Allowed/not)	allowed	allowed	allowed	not	one null object alloed	one null object allowed	not
6.Output Order	insertion order	insertion order	insertion order	sorted order	insertion order	insertion order	sorted
7.For retriving Objects what we can use	toString(),Iterator,ListIterator forloop						

Scanned with CamScanner

hashcode() :-

It is a method of object class

public int hashCode()

* It returns a hashcode number for every object
* hashcode number is a unique number generated for every object based on object address.

```
package List;
public class HashCodeSample
{
  public static void main(String args[])
  {
    HashCodeSample * s = new HashCodeSample();
    // HashCodeSample s = new HashCodeSample();
    String ob = s.toString();
    int hn = s.hashCode();
    System.out.println(ob);
    System.out.println(hn);
  }
}
```

output:-

List.HashCodeSample@4cc77c2e

1288141870

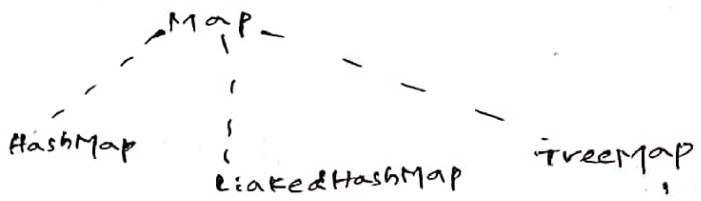
Scanned with CamScanner

Map Interface :-

* Map is an Interface which is implemented by 3 classes
* Each class will allow us to store the objects as a key,value pair is called as Entry
* So Map is Referred as Collection of Entry objects
* whenever we want to Represents objects as Keyvalue we go for Maps

Map Interface is implemented by 3 different classes

1. HashMap
2. LinkedHashMap
3. TreeMap



Ex:-

Key	Value
barcode	Amazonds packet

Ex:-2

Key	Value
batchcode	Subject.

Useful Methods are:-

put(K,V) :- used to add objects as Key value pair into Map

Keyset() :- Return all Key's available in Map

values() :- Return all values available in Map.C

ContainsKey(key) and ContainsValue(value) :-

Checks key and value is present or not respectively

Scanned with CamScanner

* HashMap:-

HashMap is a class which implements Map interface

features:-

Introduced in 1.2V
heterogeneous data allowed
Data structure is hastable
duplicate keys are not allowed
duplicate values are allowed
output will be in random order based on hashcode
only one null key is allowed and multiple null values are allowed.

```
import java.util.HashMap
public class HashMap
{
  public static void main(String args[])
  {
    HashMap<Integer, String> m1 = new HashMap<>();
    m1.put(1005, "one plus");
    m1.put(7225, "samsung");
    m1.put(1880, "xaiomi");
    m1.put(2541, "Apple");
    m1.put(4772, "oppo");
    System.out.println(m1);
    m1.put(1880, "vivo");
    System.out.println("After adding duplicate key :" + m1);
    m1.put(2014, "Apple");
    System.out.println("After adding duplicate Value :" + m1);
    System.out.println("collection of keys :" + m1.keySet());
    System.out.println("Collection of Values :" + m1.values());
    System.out.println("For getting value assosciated with
                        key :" + m1.get(2014));
```

Scanned with CamScanner

output:-

[1880 - xaiomi, 4712 = Oppo, 7225 = Samsung, 1005 = oneplus, 2541 = Apple}

After adding duplicate key : [1880 = Vivo, 4712 = Oppo, 7225 = Samsung, 1005 = One plus, 2541 = Apple}

After adding duplicate value : [1880 = Vivo, 4712 - Oppo, 7275 - Samsung, 1005 = one plus, 2541 = Apple, 2014 = Apple}

Collection of Keys : [1880, 4712, 7225, 1005, 2541, 2014]

Collection of values : [Vivo, Oppo, Samsung, One plus, Apple, Apple]

for getting value associated with key : Apple

HashTable:-

* It is a datastructure (like an array) which stores the objects as key values pair as per object HashCode number
* But hashcode is a 32 bit number so storing object at that position is a lengthy process
* In order to overcome that we use Hashing technique

formulae:- position = key % size of hastable

* Default Size of hastable is 11
* one hastable reaches the 3/4 of default Size if size will be doubled ie 22

```
m1.put(1005, "one plus");
m1.put(7225, "samsung");
m1-put(1880, "xaiomi");
m1.put(2541, "Apple");
m1.put(4712, "oppo");
```

	Key	Value
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

HashTable

Scanned with CamScanner

Hashing technique:-

position = key % size of hastable

1005 % 11 = 4 (index)

7225 % 11 = 9 (index)

11) 1005 (91
99
15
11
(4)

java

unicode of j + unicode of a + unicode of v + unicode of a

LinkedHashMap:-

LinkedHashMap is a class which extends HashMap and implements Map interface

features:-

Introduced in 1.4 version

heterogeneous data allowed

Data structure is hastable + Linkedlist

duplicate keys are not allowed but value can be duplicate

if we add duplicate key it replace with original name

As per insertion order

only one null key is allowed and multiple null values are allowed.

```
import java.util.LinkedHashMap
public class HashMap
{
    public static void main(String args[])
    {
        LinkedHashMap<Integer, String> m1 = new LinkedHash
                                            map<>();

        m1.put(1005, "one plus");
        m1.put(7225, "Samsung");
        m1.put(1880, "Xiaomi");
        m1.put(2541, "Apple");
        m1.put(4712, "Oppo");

        System.out.println(m1);
    }
}
```

output:-

{1005 = One plus, 7225 = Samsung, 1880 = Xaiomi, 2541 = Apple, 4712 = oppo}

Scanned with CamScanner

TreeMap:-

TreeMap is a class which implements Map Interface

features:-

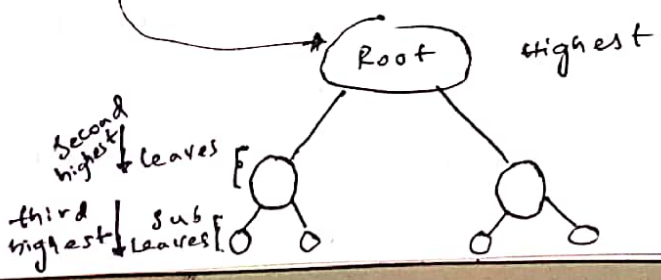
* Introduced in 1.2V
* only homogenous data allowed, if we add heterogenous objects will get class cast exception
* Data structure is tree
* duplicates keys are not allowed but value can be duplicate
* Sorting order of key's.
* Null keys are not allowed but null values are allowed

```
import java.util.TreeMap
public class TreeMap
{
  public static void main(String args[])
  {
    TreeMap<Integer,String> m1 = new TreeMap<>();
    m1.put(1005, "one plus");
    m1.put(7225, "Samsung");
    m1.put(1880, "xaiomi");
    m1.put(2541, "Apple");
    m1.put(4712, "oppo");
    System.out.println(m1);
    m1.put(null, "Nokia");
    System.out.println(m1);
  }
}
```

output:-

{1005 = one plus, 1880 = xaiomi, 2541 = Apple, 4712 = Oppo, 7225 = Samsung}

```
Exception in thread "main" java.lang.NullPointerException
at java.base/java.util.TreeMap.put(TreeMap.java:561)
at maps.HashMap.main(HashMap.java:16)
```



Scanned with CamScanner

String class

It is a pre-define class which is present in java.lang package

purpose:-

here will study about methods of string class and using them how to manipulate our string data

String object can be created in two ways:-

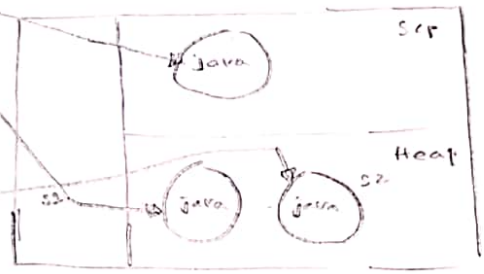
1. By using new keyword:-

ex: String s = new String("java");

* when we create a string object with new keyword two objects get created
* one is in Heap area and another one in string Constant pool Area
* string Constant pool Area is small part of Heap Area
* object created under heap area reference will be assign to it
* object created under scp is like a backup object where no refereace will be assign to it

ex:-1

String s1 = new String("java");
String s2 = new String("java");



If there is already an copy of same Content exisists jvm will not creates another object in scp area

2. without new keyword:-

ex:- String s = "java", s1 = "java";

1. Arrays
2. strings
3. Wrapper classes

when we create an String object without new keyword that first jvm will go to scp area and check whether there is an object present or not with given Content

Scanned with CamScanner

* If it is there JVM will assign reference to the existing object
* If it is not there JVM will create a new object in SCP and assign reference to it.

from the above example we observed that for one String object there are multiple reference so if we modify content of our String object it will effect to multiple references so therefore

String objects are declared as IMMUTABLE once we created we cannot modify it.

Methods of String class:-

1. length(): int

Syntax:- public int length()

It provides length of String (no of characters) length will be always calculated from 1

Ex:- String s = "java";
System.out.println(s.length());

output:- 4

```
public class First
{
    public static void main(String args[])
    {
        String s = new String("java developer");
        int len = s.length();
        System.out.println(len);
    }
}
```

output:-
14

2. charAt(index): char

Syntax:- public char charAt(int);

It provides character at given index

for ex:- String s = "java";
System.out.println(s.charAt(3));

output is a

Scanned with CamScanner

If we provides invalid index as args for charAt() we will get StringIndexOutOfBoundsException

```
public class First
{
public static void main(String args[])
{
    String s = new String("java developer");
    for(int i=0; i<=s.length(); i++)
    {
      char ch = s.charAt(i);
      System.out.println(ch);
    }
  }
}
```

output:-

```
j
a
v
a
d
e
v
e
l
o
p
e
r
```

Program:-1

write a program to calculate Count of e's character present in given string

String s = "development"

```
public class First
{
  public static void main(String args[])
  {
    String s = new String("development");
    int Count =0;
    Char ch;
    for(int i=0; i<=s.length-1; i++)
    {
      ch = s.charAt(i);
    if(ch=='e')    (e should be in Single quotes)
    {
      Count++
    }
  }
  System.out.println("count of e is :" +Count);
}
}
```

output:-

Count of e's is 3

Scanned with CamScanner

Prog:- 2 WAP to print all Vowel's from given String

String s = "development"

```
public class First
{
  public static void main(String args[])
  {
    String s = "development";
    char ch;
    for(int i=0; i<=s.length()-1; i++
    {
      ch = s.charAt(i);
      if(ch=='a'|| ch=='e'|| ch=='i'|| ch=='o'|| ch=='u')
      {
        System.out.println(ch);
      }
  }
}
```

output.

e
e
o
e

3. object1.equals(object2) : boolean :-

it checks whether two string are same or not

it Compares two string based on string data

1. String s1 = "java";
   String s2 = "java";
   System.out.println(s1.equals(s2)); //true

2. String s1 = "java";
   String s2 = "javadev";
   System.out.println(s1.equals(s2)); //false

3. String s1 = "java";
   String s2 = "JAVA";
   System.out.println(s1.equals(s2)); //false

because equals method Consider case Seasivitytivity.

Scanned with CamScanner

4. object1.equalsIgnoreCase(object2)

Compare two Strings based on String data without Considering case sensitivity

```
String s1 = "java";
String s2 = "JAVA";
System.out.println(s1.equalsIgnoreCase(s2)); // true
```

3 write a program to print reverse of a string

4. write a program to Check whether String is palindrome or not

```
public class Reverse
{
public static void main(String args[])
{
String s1 = "madam";
String rev = "";
for(int i = s1.length()-1; i >= 0; i--)
{
rev = rev + s1.charAt(i);
},
System.out.println(rev);
if(s1.Equals(rev)
{
System.out.println("palindrome String");
}
else
{
System.out.println("Non palindrome String");
}
}
}
```

output:-

madam

palindrome String

Scanned with CamScanner

```
public class Reverse
{
  public static void main(String args[])
  {
    String s1 = "Madam";
    String rev = "";
    for(int i = s1.length()-1; i>=0; i--)
    {
      rev = rev + s1.charAt(i);
    }
    System.out.println(rev);
**  if(s1.equalsIgnoreCase(rev))
    {
      System.out.println("palindrome string");
    }
    else
    {
      System.out.println("Not palindrome string");
    }
  }
}
```

output:-
madam
palindrome string.

5. trim(): String

it removes white spaces from starting and ending of string

trim() will not remove spaces from middle

program:5

write a program to Count total no of words present in string

s = " am an test engg

```
public class Counting
{
  public static void main(String args[])
  {
    String s = "  I am an  test engg  ";
```

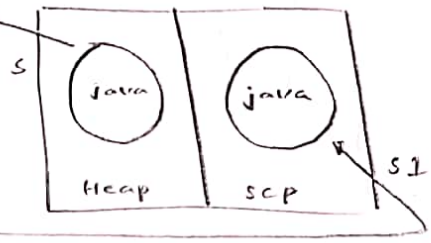
Scanned with CamScanner

```
String s1 = s.trim();
System.out.println(s1);
    int Count=1 // atleast one word is there
for(int i=0; i<s1.length(); i++)
{
    if(s1.charAt(i)=' ' && s1.charAt(i+1)!=' ') // current character is space and next Character is not a space
    {
        Count++
    }
}
System.out.println("Count is :"+Count);
}
}
```

output:-
I am an test sagg
Count is : 5

* Equals(): Compared based on Content
* == : Compared based on reference

Ex:-
```
String s = new String("java");
String s1 = "java";
```




```
Sop(s==s1); // Compare whether s and s1 pointing to same object → false
sop(s.equals(s1)); // Compare whether s and s1 have same data or not → true
```

6. substring(int from index) : string

* It provides subpart of a String

Ex:-
```
String s="java";
String s1 = substring(1); // it retrieves all characters from 1st index till Last
Sop(s1);
```

Substring(int from index, int to index) : String

substring(int from index(including), int to index(excluding)) : String

Ex:- String s="java";

Scanned with CamScanner

```
String s1 = substring(1, 3); //1- starting index and 3- ending index
                                                    (1, 3-1)
Sop(s1);

public class Counting
{
  public static void main(String args[])
  {
    String s = "I am an test sagg";
    String p1 = s.substring(4);
    String p2 = s.substring(11,17); //4, 7-1
    System.out.println(p1);
    System.out.println(p2);
  }
}
```

output:-

```
an test sagg
an t
```

Program :- Write a program and check the Count of happy in given string

```
public class Counting
{
  public static void main(String args[])
  {
    String s = "I want to be happy stay happy and make people
                happy";

    String p2[] = s.split(" ");
    int count = 0;
    for(int i=0; i<p2.length; i++)
    {
      if(word.equals(p2[i]))
      {
        Count++;
      }
    }
    System.out.println("Count of "+word+" is :" + count);
  }
}
```

output:-

```
Count of happy is : 3
```

Scanned with CamScanner

* write a program to find longest word from a string

```
" works in Deloitte

public class Counting
{
    public static void main(String args[])
    {
        String s = " works in Deloitte";
        String p2[] = s.split(" ");
        String max = p2[0];
        for(int i=0; i<p2.length; i++)
        {
            [redacted] = [redacted];
            if(p2[i].length() > max.length())
            {
                max = p2[i];
            }
        }
        System.out.println("Longest word is : " + max)
    }
}
```

* output:-
Longest word is : Deloitte

* toCharArray(): char[]

It Converts String into character array

Ex:- String s = "javadev";

char ch[] = s.toCharArray();

op:- ch[0] = 'j'
ch[1] = 'a' - - -

Scanned with CamScanner

program:-

write a program to calculate frequency of characters present in a string

ip : javadev

op : j-1 d-1
a-2 e-1
v-2

```
public class Frequency
{
 public static void main(String args[])
 {
    String s1 = "javadev";
    s1 = s1.toUpperCase();
    char s[] = s1.toCharArray();
    for(char ch = 'A'; ch <= 'Z'; ch++)
    {
      int Count = 0;
      for(int i=0; i<s.length; i++)
      {
         if(ch == s[i])
         {
           Count++
         }
      }
      if(count>0) // want to print only characters which
                     occurs atleast 1 time
      {
        System.out.println("count of "+ch+" is:"+Count);
      }
    }
 }
}
```

* output:-

Count of A is : 2
Count of D is : 1
Count of E is : 1
Count of J is : 1
Count of V is : 2

Scanned with CamScanner

8. toUpperCase(): String ---→ Converts string into uppercase

9. toLowerCase(): String ---→ Converts String into lowercase

10. startsWith(String): boolean ---→ Check whether string is starting with given string or not

11. endsWith(String): boolean ---→ checks whether String is ending with given string or not

12. Contains(String): boolean ---→ checks whether string is present in given String or not

13. IsEmpty(): boolean : ---→ checks whether string is empty or not

14. object.Concat(String) ---→ Concatinates string in end of called string

Note:-

startswith(), endswith() and Contains() check with case sensitivity

```
class frequency
{
public static void main(String args[])
{
    String s = "java Developement with JDK 16";
    String s1 = s.toUpperCase();
    System.out.println(s1);
    String s2 = s.toLowerCase();
    System.out.println(s2);
    boolean b1 = s.startsWith("java");
    System.out.println(b1);
    boolean b2 = s.endsWith("java");
    System.out.println(b2);
    boolean b3 = s.Contains("with");
    System.out.println(b3);
    String s3 = s.Concat("or JSE 16");
    System.out.println(s3);
}
}
```

Scanned with CamScanner

output:-
JAVA DEVELOPMENT WITH JDK16
java development with JDK16
false
false
true
Java Development with JDK16 or JEE16

indexOf()

indexOf(char): int

returns index for given character

indexOf(String): int

returns index of starting character of a string

indexOf(String, fromindex): int

It provides index value for given string by start searching from given index not from zero

Note :- indexOf(): return -1 if character is not present in String

```
class Frequency
{
    public static void main(String args[])
    {
        String s = "java developement";
        int i = s.indexOf('a'); //when we have Repeated characters
                                  it gives index of first occurence
        int j = s.indexOf("eve");
        int k = s.indexOf("a", 2);
        int m = s.indexOf('z');
        System.out.println(i);
        System.out.println(j);
        System.out.println(k);
        System.out.println(m);
    }
}
```

output
1
6
3
-1

Scanned with CamScanner

* write a program to remove duplicate characters from a string

String s = "testinggen"

output :- tesing

```
public class Frequency
{
  public static void main(String args[])
  {
      String s = "testingen";
      String dup = "";
      for(int i=0; i<s.length(); i++)
      {
         char ch = s.charAt(i);
         if(dup.indexOf(ch) == -1)
         {
            dup = dup + ch;
         }
      }
      System.out.println(dup);
  }
}
```

* output:-

tesing

Scanned with CamScanner

StringBuffer and StringBuilder

* These are classes which are present in java.lang.package
* String buffer and String builder objects can be created only by using new keyword

ex:- StringBuffer b1 = new StringBuffer("java");
StringBuilder b2 = new StringBuilder("java");
StringBuffer b2 = "java"; // invalid
StringBuilder b4 = "java" // invalid

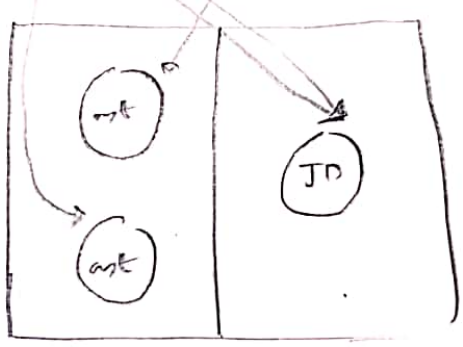
- StringBuffer and Builder objects are mutable once we create we can modify it
- StringBuffer and String Builder objects will get created in heap area only

→ append() is used for modifying Stringbuffer and String builder objects

```
public class Demo
{
  public static void main(String args[])
  {
    StringBuffer s1 = new StringBuffer("Manual testing");
    StringBuilder s2 = new StringBuilder("Manual testing");

    String s3 = "java development";
    String s4 = "java development";
    System.out.println(s1.equals(s2));
    System.out.println(s3.equals(s4));
  }
}
```

output:-
false
true



Scanned with CamScanner

String	String Buffer	String Builder
1. String objects are immutable	1. String buffer objects are mutable	1. String builder objects are mutable
2. objects will be created in 2 ways 1. new 2. literal (without new)	2. objects can be created only using new keyword	2. objects can be created only using new keyword.
3. objects will created in SCP or heap	3. object will created in heap	3. object will created in heap
4. if Context is fixed we will go to string	4. if Context is varying we will go for String buffer	4. if Context is varying we will go for String builder
5. equals() – compares two string by seeing Content	5. equals() – Compares two string by seeing reference	5. equals() – Compares two String by Seeing reference
6. performance is faster	moderate	6. faster
7. for Concatination we have Concat()	7. for Concatination we have append()	7. for Concatination we have append()
8. Thread safe	8. Thread safe – multiple threads not allowed	8. Not thread safe

Scanned with CamScanner

Access Specifier or Access Modifiers

Provides Accessibility permission for various field like class, method, Variable and interface

- public
- private
- protected
- default

⇒ Public :-

public fields are accessible

(a) within same class

~~public field are accessible~~

```
package demo1;
public class Sample1
{
    public static int i=100;
    public static void main(String args[])
    {
        System.out.println(i);
    }
}
```

(b) within sub class of same package

```
package demo1
public class Sample2 extends Sample1
{
    public static int i=100;
    public static void main(String args[])
    {
        System.out.println(Sample1.i);
    }
}
```

Scanned with CamScanner

C. within non subclass of samepackage

```
package demo1
public class Sample2
{
  public static int i = 100;
  public static void main(String args[])
  {
    System.out.println(Sample1.i)
  }
}
```

d. within class of different package but only after import statement

```
package demo2
import demo1.Sample1;
public class Pp
{
  public static void main(String args[])
  {
    System.out.println(Sample1.i);
  }
}
```

default:-

a. within same class
b. within sub class of same package
c. within non subclass of same package

— default field not accessible

d. default fields cannot be accessible different package

Conclusion:- default fields are only available within package

Scanned with CamScanner

Protected :-

a. within same class
b. within subclass of same package
c. within non subclass of same package
d. within class of different package but only after inheritance and adding import statement

```
package demo2;
import demo1.Sample1
public class Pp extends Sample1
{
  public static void main(String args[])
  {
    System.out.println(Sample1.i);
  }
}
```

*Note :- we cannot apply ~~[illegible]~~ protected access modifier for class or interface

Private :-

Private field are accessible
a. only with in class

Scanned with CamScanner

	private	Default	protected	public
Same class	Yes	Yes	Yes	Yes
Same package Subclass	No	Yes	Yes	Yes
Same package Non-Subclass	No	Yes	Yes	Yes
Different package Subclass	No	No	Yes	Yes
Different package non-Subclass	No	No	No	Yes

⇒ final vs finally vs finalize

final	finally	finalize
final is a keyword	finally is a block	finalize() method is a protected method of java.lang.Object class It is inherited to every class you create in java
final is used to apply restriction on class method and variable	finally is a block which is used for Exception handling along with try and Catch blocks	This method is Called by garbage Collector thread before an object is removed from memory
* If the final Keyword is attached to a variable then the variable becomes Constant its value Cannot be changed in program	* finally is always executed whether exception is raiser or not and raised Exception is handled or not most of time this block is used to close the resources like database Connection, I/O resources .. etc	* finalize() method is used to perform some clean up operations on an object before it is removed from memory

Scanned with CamScanner

If a method is marked as final then the method cannot be overridden by any other method

finalize is useful for more than just exception handling: it allows the programmer to avoid having cleanup code accidentally bypassed by a return continue or break. putting a cleanup code in a finally block is always good pratice even when no exceptions are anticipated

for each loop

```
for(Arraytype var: arrayname)
{
    Sop(var);
}
```

ex:- int a[] = {10, 20, 30, 40}

```
for(int data : a)
{
    Sop(data);
}
```

* output:-
10
20
30
40

(or)

```
for(Collectiontype var: Collectionname)
{
    Sop(var);
}
```

```
ArrayList<Object> a1 = new ArrayList<>();
a1.add(100); a1.add("java"); a1.add(true);
for(Object f1 : a1)
{
    Sop(f1)
}
```

Scanned with CamScanner