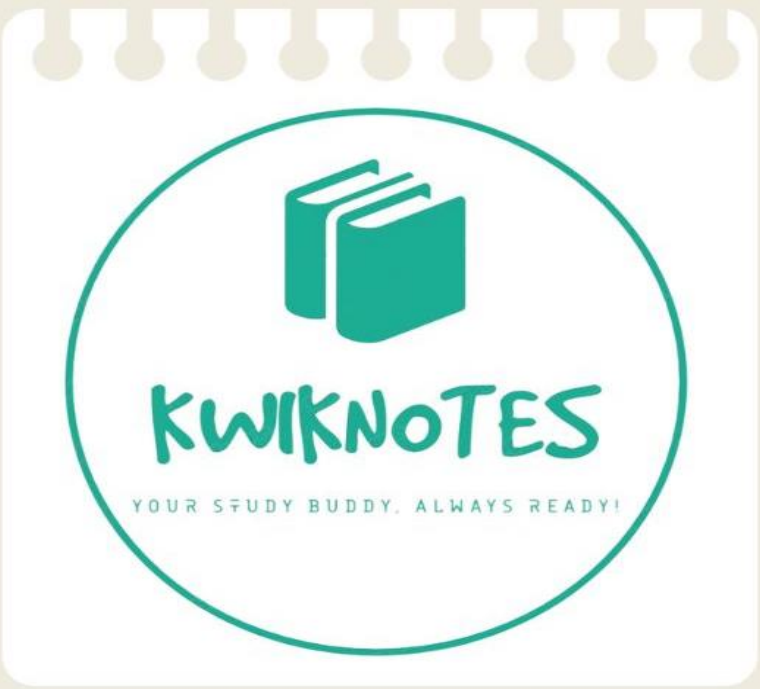


HANDWRITTEN NOTES

Download FREE Notes for Computer Science and related resources only at

Kwiknotes.in

Don't forget to check out our social media handles, do share with your friends.

JAVA

It is a widely used OOP language & software platform that run on billions of devices

It is a programming language and a platform

Java is a high level robust, OOP, secure language

Developed by Sun MicroSystems, 1995.

Father of Java - James Gosling

Why called a platform?

Any hardware or software environment in which a program is Run is called platform. Since Java has a Runtime environment (JRE) and API, so it is called platform

Rules & Syntax of Java are based on C and C++ language.

Application Not only used in S/W but also used in designing H/W controlling software components. 3 billion mobile phone run on Java.

History

Originally designed for interactive television but was too advanced for digital cable television at that time.
Java team members were known as Green team. initiated it to make digital devices such as settuphox television, etc.
However it best suited for internet prog
Later, Java was incorporated by Netscape.

Team of Sun engineers - James Gosling Mike Sheriden, Patrick Naughton.

1) First it was called Greentalk with .gt as extension
2) then Oak developed as a part of Green Project as Oak is national tree of US, France, etc.

In 1995, it was renamed Java because Oak was trademarked by Oak Technologies.

Features

1) Object Oriented
In Java everthing is object. Java can be easily extended since it is object based

2) Platform Independent
Unlike C, C++, when Java compiled it is not compiled into platform specific machine.

3) Simple
easy to learn, Simple syntax, pointer or operator loading nhi h i sme

4) Secure
It enable to develop virus free System

5) Portable
Java has clean portability boundary

6) Multithreaded
With multithreading it is possible to write programs that can perform many task simultaneously.
It allows to construct interactive apps that can run smoothly.

Distributed
Designed for distributed env of internet

Dynamic
More dynamic than C, C++

Robust
Java is Robust (strong)

C++ vs Java

	C++	Java
1)	Platform Dependent	Independent
2)	Used for system Programming	Used for app prog used in window based, web based
3)	Supports goto statement	Don't
4)	Support Multiple Inheritence	MI through class. MI achieved through Interfaces
5)	Operator overloading	Not supported
6)	Support call by value / Call by Refrence	only call by value
7)	Interactive with H/W	not so interactive H/W
8)	Has virtual Keyword	Not have

JDK Java Development Kit
It is a development Environment for building applications, applets and compo using Java Programming language.
JDK includes tools useful for developi and testing programs written in Java.
It physically exist.
It contains JRE + development tools.

JRE Java Runtime Environment
is a software that Java program requir to Run properly.
It is an underlying technology that communicates b/w Java and OS.

JDK for Development JRE for running

JVM Java Virtual Machine
is a virtual machine that enables computer to Run Java program as well as program written in other lang that are also compiled to Java bytecode

JDK Java Development Kit
It is a development Environment for building applications, applets and component using Java Programming language.
JDK includes tools useful for developing and testing programs written in Java.
It physically exist.
It contains JRE + development tools.

JRE Java Runtime Environment
is a software that Java program require to Run properly.
It is an underlying technology that communicates b/w Java and OS.

JDK for Development JRE for running

JVM Java Virtual Machine
is a virtual machine that enables Computer to Run Java program as well as program written in other lang that are also compiled to Java bytecode

Memory Management in Java

It is the process of allocation & deallocation of objects,

Java does MM automatically. It uses automatic memory management System called garbage collector

It is divided to two parts

- JVM Memory structure
- Working of garbage collector

Variables

Local

declared inside the body of a method/function

Can be only used with that method

Can be defined with static keyword

Instance

Declared inside class but outside body of method. Not declared as static. Value is instance specific

Static

Declared with static keyword

Single copy can be created & can be shared among instance

Access Modifiers

AM specifies accessibility or scope of a field, method, constructor or class. We can change access level of fields, constructors, methods and class by applying access modifier on them

1) Private - access within class. Cannot be accessed outside the class
Specified with private keyword.
Any other class of same package will not be able to access the members
Cannot be accesed within same package outside class

```
Class A {
private void display () {
        sout ("Hello");
    }
}
Class B {
  public static void main (String args[]) {
        A obj = new A();
        obj.display();
  }
}
```

→ Error display has private access in A

Protected : Specified using protected Keyword . Access Level is within package and outside package through child class. If you donot make child class , it cannot be accessed from outside package.

```
package 1;
public class A {
    protected void display(){
        Sout ("Protected fun");
    }
}

package 2;
import p1.*;  // all class from p1

Class B extends A {
    public static void main (String args[])
    {
        B obj = new B();
        obj.display;
    }
}
```

⇒ output

⇒ Protected fun.

Public - specified using public Keyword
It has widest scope among all modifiers.
These are accessible from everywhere in program. There is no Restriction

```
package p1;
public class A {
    public void display() {
        System.out("Public");
    }
}

package 2;
import p1.*;
class B {
    public static void main(String args[]) {
        A obj = new A();
        obj.display();
    }
}
```

⇒ Public.

Default - access only within package.
Cannot be accessed outside package.
If you do not specify any access level, it will be default.

```
package p1;
class A {
    void display() {
        System.out.println("Default");
    }
}
```

```
package p2;
import p1.*;
class B {
    public static void main(String[] args) {
        A obj = new A();
        A.display();
    }
}
```

Control Statement-

1) Decision Making
   - if -else
   - Switch
2) Loop Statement
   - do while
   - while
   - for
   - for each
3) Jump Statement
   - break
   - continue

If

It evaluates a Boolean expression & enables the program to enter a block of code, if the condition is true.

```
if (cond) { - - - }
else { - - }
```

```
if ( ) { }
else if ( ) { }
else { }
```

Switch

```
switch (expression) {
   case value:
        - - -
   case value2:
        - - - -
   default:
        - - -.
}
```

Loop

```
for (start, end, step) { - }
```

For each

```
public static void calc {
    public static void main (String[] args) {
        String = {"Java", "C", "C++", "Python"};
        Sout("List is");
        for (String name : String) {
            Sout(names);
        }
    }
}
```

while loop

```
while (condition) {
    - -
}
```

```
do {
    - - -
} while (condition);
```

Arrays

Array is a group of values stored in a single variable instead of declaring separate variable for each value.

They are enclosed in between square brecket

Java array is an index based object which contains elements of similar datatype
Element stored in contiguous memory location

length() - to get length of array

```
class A{
    public static void main (String [] args){
        int a[] = new int [5];
        a [0] = 1;
        a [1] = 2;
        a [2] = 3;
        a [3] = 4;
        a [4] = 5;
        for (i = 0; i <= a.length ; i++) {
            Sout (a [i]);
        }}
// for each
    for (int i : a
        Sout (i);
```

Page No.
DATE: / /

Input Output Stream

Java I/O is used to process the input & produce output.

Java uses I/O to make the concept of Streamfast.

Java.io package contains all the classes required for input & output operations

file handeling can be performed by Java I/O API.

Stream

A stream is a seq of data. It is composed of bytes. Called stream becoz it is like stream of water which continues to flow

3 stream in Java

1) System.out — output
2) System.in — input stream
3) System.err — Standerd error stream

Classes

1) Object
An entity that has state and behavior
eg chair, bike, pen, etc. It can be physical or logical (banking system).

It is basic unit of OOPS & represent real time entities.
Characteristics.
1) State - represented by attribute of obj
   Reflects properties of object
Rpresent data of objet
2) Behavior - represented by method of obj
3) Identity - implemented via unique id

Class

is a set of objects which share common characteristics / behaviour
Class is not a real world entity. It is a template or blueprint or prototype from which objects are created.

A class contain
- data member - method - constructor
- nested class - interface
Everything in Java is associated with classes & objects.

Instance variable - created in class outside method. Inside method are local var

```
import java.util.Scanner;
class add {
int a,b;
void getdata() {
   Scanner sc = new Scanner(System.in);
   Sout("Enter a:");
   a = sc.nextInt();
   Sout("Enter b:");
   b = sc.nextInt();
}
void putdata(){
      Sout("Addition =" + (a+b));
}
public static void main(String args[]) {
   add obj = new add();
   obj.getdata();
   obj.putdata();
}
```

2) Reverse of a no. with oops.

```
import java.util.Scanner
class Revrse
{ int n;
  void getdata(int i) {
     n=i; }
  void put() {
    int revrzo;
    while(n>0) {
```

new allocate memory at runtime in heap memory area.

```
    n = n/10;
    Sout (rev);
  }
}
public static void main (String args[]) {
    int m;
    reverse obj = new reverse();
    Scanner SC = new Scanner(System.in);
    Sout ("Enter no:");
    m = SC.nextInt;
    obj.getdata(m);
    obj.put();
}
```

Advantage of Class

Code Reusability

Code optimization

Method

A method is a block of code which can run when it is called

we can pass data as parameters to method

Java Constructor types Parameterized, default

A constructor in Java is similar to a method that is invoked when an class is instantiated ie object is made.
These are member functions that get invoked on creating object
At the time of calling constructor, memory memory for the object is allocated.
Every time object is created using new() constructor is called. A default constructor is called

Called constructor because create value at object creation.
Java creates a default constructor if not any

Rule
Name same as class - no explicit return type
Cannot be abstract, Static, final, synchronized.

```
Class Student {
    int id;
    String name;
    Student(int i, String n) {
      id = i;
      name = n;
    }

    void display() {
        Sout(id + " " name);
    }

    public static void main(String args[]) {
        Student s1 = new Student(1, "Kashish");
        Student s2 = new Student(2, "abc");
        s1.display();
        s2.display();
    }
}
```

2) 1 Kashish
   2 abc

Java Constructor	Java Method
- Not have return type	have
- Must be same name as of class	may / may not be
- invoked implicitely used to initialize the state of object	explicitely used to expose behaviour of an object.

Inheritence

is a concept that acquire the properties from one class to other class

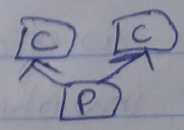
It is a mechanism in which one object acquire all properties & behavior of a parent object. It is a part of OOP's

Inheritence Represent IS-A relationship. also parent child relationship

Code Reusability - Implement polymorphism

```
class A {
    int a = 100;
}
class B extends A {
    int b = 200;
    public static void main (String args[]) {
        B obj = new B();
        sout (p.a + p.b);
    }
}
```

⇒ 300



Types - singl, mutilevel Heirarchical

Q Why Multiple Inheritence not supported in Java?
To reduce complexity and simplify language multiple inh is not supported.

Consider A, B and C are classes. C inherit A & B.
If A & B have same method & you call it from child object ie C is object, there will be ambiguity.
It render compile error in multiple inh.

Component

Class — Superclass / Parent class / Base Class — Subclass / Derived / Extended / Child Class

Method overriding - same function in parent & child class.

Final Keyword

Final is Keyword is a non access modifier used for classes, attributes & methods which makes them non changable (impossible to inherit or override)
It is a modifier which provide Restriction for 1) variable 2) Method 3) Classes

1) final variable → once declared as final, a variable cannot be reassigned

final int A = 10;

↳ final Keyword in caps.

```
class Final {

public static void main (String args[]) {
        final A = 10;
            sout (A);          => 10
        A = 20;
            sout (A);          => Error as it can be reassigned
    }
}
```

2) final method – when a method is declared as final, it cannot be overridden in child / extended class

final void funname () { --- }

Ex –

```
class final {
    final void pin () {
        sout ("123");
    }
}
class access extends final {
    void pin () {
        sout ("456");          // overriden but give error
    }
}
```

```
class Main {
    public static void main(String args[]){
        access obj = new access();
        obj.fin();
    }
}
```

⇒ error as, it cannot override final method

3) final class - when a class is made final it cannot be extended or inherited to a Subclass

Syntax: final class A { - - - }

Ex -

```
final class A {
    int a=10;
}
class B extends A {
    void print {
        Sout (super.a); }
}
class Main {
    public static void main(String args[])
    {
        B obj = new B();
        obj.print();
    }
}
```

⇒ error : print in access cannot inherit final class

Page No.
DATE: / /

Package — It is a container / mechanism

A Package in Java is a group of similar types of classes, interfaces and sub-packages.

A package arrange no. of classes, interfaces and subpackage of same type into a particular group.

ex- folder in windows.

It is used for -
- Prevent Naming conflict
- Make Searching / locating easier
- For data encapsulation (data hiding)

Types

Predefined :
java.util
java.lang (by default)
java.io
java.applet
java.SQL

User defined package P1
package add
package mypack

User defined :
package package_name

Java is hybrid lang. both compiles & Interpreted.

Abstract Classes, is a
A class which usses abstract Keyword
It can have abstract or non abstract methods.

Abstraction is the process of hiding implementation details & showing only functionality to user.

It is used to provide a base for subclass to extend & implement abstract class methods

⇒ We cannot create object of abstract class
abstract cass A{ ---}

* it can have abstract or non abstract methods.
Abstract class number & variable can be accessed by subclass in which it is inherited.

Note: Is is mandatory to make a class abstract if it conteins abstract method.

* If Object not created?
Ans.

Purpose of abstract class
to provide a common definition for a base class that multiple derived class can share.

It cannot be instantiated.

It can have constructor & static methods

```
abstract class Bike {
    abstract void run();
}
class Honda extends Bike {
    void run() {
        Sout("Honda Run");
    }
public static void main(String args[]) {
    Bike obj = new Honda();
    obj.run();
}
}
```

Superclass can store any sub object

⇒ Honda Run.

```
abstract class Bank {
    abstract int intrest ();
}
class SBI extends Bank {
    int intrest () { return 7; }.
}
class PNB extends Bank {
    int intrest () { return 8; }
}
class Main {
    public static void main (String args[]) {
        Bank b;
        b = new SBI;
        Sout ("Intrest =" + b.intrest() );.
        b = new PNB;
        Sout ( "Intrest=" + b.intrest() ; } );
    }
}
```

Interface

An interface is a blueprint of class
It contains static constant & abstract classes.
To achieve interface java provides a Keyword called implements.

Interface methods are by default public & abstract

Objects cannot be made

```
interface Bicycle {
    void brake(int decrement);
    void speedup(int increment);
}

class AvonCycle implement Bicycle {
    int speed=7;
    void brake(int decrement) {
        speed = speed - decrement;
    }
    void speedup(int increment) {
        speed = speed + increment;
    }
}

public class main {
    p s v main(String args[]) {
        Bicycle obj = new AvonCycle();
        obj.brake(5); } }
```

Purpose -

To achieve abstraction

By inheritence, we can implement

Multipl Inheritence => class A implement (, B

Used to do Loose Coupling

Polymorphism

It is a OOPs concept by which we can perform single action in different ways. It refers to the ability of a class to provide different implementations of a method.

2 types
→ Compile time (Static polymorphism)
→ Runtime (dynamic polymorphism)

Polymorphism allows you to define one interface and have multiple implementation.
Poly - many, morphism - forms.

Any Java object that can pass more than one IS A object can be polymorphic.

Compile Time PolyMorphism

→ Static → Early Binding

This type of polymorphism is resolved during compilation process.
Overloading of method is called through refrence variable of a class.

This can be achieved through method overloading & operator overloading

Interfa
public

Objects

ex- interf

class

public
p s

Purpo
- To ac
- By in
Mul
- overrid

Function / Method Overloading

It occurs when Multiple functions with same name but different parameters then these functions are said to be overloaded by changes in no. of arguments and / or change in the type of argument It can have different datatype or no. of parameters.

```
public class overload {
    //int
    void Sum(int a, int b) {
        Sout (a+b); }
    void Sum(double a, double b) {
        Sout (a+b);
    }
    public static void main (String args[])
    overload obj = new overload();
        obj.sum (5,6);
        Obj. sum (7.1 + 0.2);
```

2) Operator Overloading

3) templates

Runtime Polymorphism

→ Dynamic Method Dispatch

It is a process in which a function call to to the overridden method is Resolved at Runtime.

Achieved By Method Overridding

Method overriding occurs when a derived class has a definition for one of the member function of base class. Body of function is overridden / changes

```
Class Shape {
    void draw() {
        Sout("Shape class");
    }
}
Class Square extends Shape {
    void draw() {
        Sout("Square");
    }
}
Class Main {
    public static void main(String args[]){
        SquareObj = new Square();
        // Shape Obj = new Square();
        obj.draw();
    }
}
```

⇒ Square

Constructor Overloading

In Java, we can overload constructor like method. It can be defined as a concept of having more than one constructor with different parameter so that every constructor can perform different tasks

```
ublic class Student {
    int id;
    String name;
    Student() {
        sout("Default constructor"); }
    Student (int i, String n) {
        id = i;
        name = n; }
ublic static void main (String [] args) {
    Student& S1 = new Student();
    Sout ("Default cons");
    Sout ("id" + S1.id + "name=" + S1.name);
    Sout ("parameterized");
    Student S2 = new Student (1, "Kash");
    Sout ("id =" + S2.id + "name =" + S2.name);
}
```

this is default constructor
Default Cons
id = 0
name = null
Parameterized
id = 1
name = Kash.

Exception Handelling

It is a mechanism in Java to handle runtime error so that regular flow of applications can be preserved
Runtime errors such as - ClassNotFound Exception, IO Exception, etc

Exception it is an unwanted event that disrupts the normal flow of the program. It is an object thrown at runtime.
When an exception occurs within a method, it creates an object. This is called exception object. It contains info about exception, such as name & description of exception.
Reasons - Invalid input, Code error, opening unavailable file, Device failure

Error

It represent irrecoverable condition such as JVM - Java virtual Machine out of memory, memory leaks, infinite loops, recursions, etc

Exception Handeling

It is a mechanism to handle runtime error such as Exceptions.

The advantage of Exception Handelling is to maintain normal flow.

Exceptions

Checked → checked at compile time

Unchecked → checked at Runtime

→ Error

Terminologies

try - can't be used alone. Must be followed by catch. Try block contains exception code block.

catch - used to handle exception. Must be preceded by a try block. Can be followed by finally block later.

try { -- }
catch(Exception e) { -- }

finally - used to execute necessary code of program. whether exception is handeled or not. used to close connection.

throw - used to throw an exception.

throws - use to declare exceptions. It specifies that there may occur an exception in method.

```
class Test {
    public static void main(String args[]) {
        try {
            int data = 25/0;
            Sout(data);
        }
        catch (ZeroDivisionError e) {
            Sout(e);
        }
        finally {
            Sout("Rest Alway Executed");
        }
```

throw	throws
→ used to throw exception explicitely	to declare an exception which might be thrown by function while execution
→ followed by instance of exception	followed by classname of exception
→ used within function	with func declaration

Throw

```
public class test{
    public static void num(int n){
        if(n<1){
            throw new ArithmeticException
            ("n is negative");
        }
        else{
            Sout(n**2);
        }
    }
```

Throws

```
public class test{
    public static void num(int n,)throws{
        throws ArithmeticException{
            int div = m/n;
            return div;
        }
    }
    publi static void main(String args[]){
        test obj = new test();
        try{
            Sout(obj.num(5,0));
        }
        catch(ArithmeticException e){
            Sout("Error");
        }
    }
```

Page No.
DATE: / /

Exception Class

It is the base class from which exceptions inherit.
The class Exception & its subclasses are a form of throwable that indicates conditions that a reasonable application might want to catch.
This class uses throw ~~throw~~ keyword

Checked Exception

Checked at compile time. If some code within a method throws a checked exception, the method must either handle exception or must specify exception using throws.

Fully checked → Partially checked

Unchecked

Not checked at compile time.
In Java, Exception under Error & RuntimeException classes are unchecked, everything else under throwable is checked.

User defined Exception

Also called custom Exceptions
We can create user defined Exception that are derived classes of Exception class.
Created as per our case & throw using throw keyword

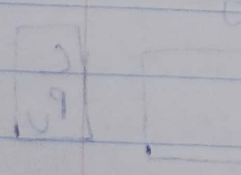
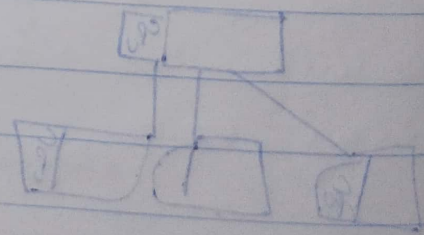
```
class InvalidAgeException Extends Exception
{
    InvalidAgeException (String msg) {
        sout (msg);
    }
}
class test {
    public static void main (String[] args)
    {
        try {
            vote (12);
        }
        catch (Exception e) {
            sout (e);
        }
    public static void vote (int age) throws
        InvalidAgeException {
            if (age < 18) {
                throw new InvalidAgeException
                ("Not Eligible");
            }
            else {
                sout ("Eligible");
            }
        }
    }
```

Multithreading in Java

Process of executing multiple threads at same time without dependency on other threads.
Increase performance
Used to do multitasking within a process
Multithreading allows concurrent execution of two or more parts of a program for maximum utilization of CPU.
Each part of such program is called thread.

Thread - are lightweight processes within a process. Allows to Run things simultaenously. Used to perform task in background without affecting main program.

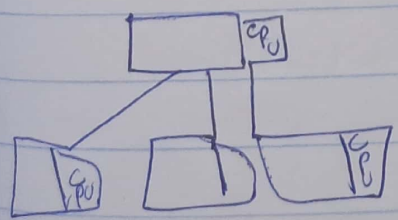
Threads can be created by 2 mechanism:
1) Extending thread class
2) Implementing the Runnable Interface.

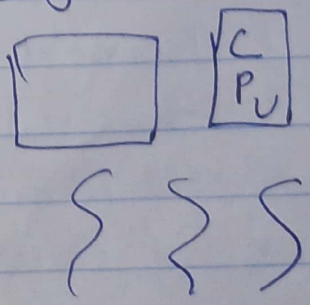



```
public class Main Extends Thread {
    public static void main(String[] args) {
        Main thread = new main();
        thread.start();
            sout("This is outside thread");
    }
    public void run() {
        Sout("This is inside thread");
    }
}
```

Multiprocessing	Multithreading
CPU are added for increasing computing power	Many threads are created of a single process for increasing computing power
Many processes executed simultaneously	→ Many threads of a process are executed simultaneously
Every process owns a separate address space	→ Common address space shared by all thread

Multiprocessing can be Symmetric & ASymmetric → No categories





Thread can be created in two ways
1) Extending thread class
2) Implementing Runnable interface

1) Thread class
Java.lang.Thread is a thread of execution in a program. JVM allows an app to have multiple threads of execution running concurrently.
Thread class provide constructor & methods to create & perform operations on a thread. Thread class extend obj class & implement Runnable interface

We can override Thread predefined functions by inheriting it in base class
JVM bydefault calls main() thread.
Commonly used Thread method:
- public void run() - perform action for thread
- public void start()
- public void sleep()
- public void join()
- public void getName()
- public void getPriority()

(ms) → Milliseconds

1 sec = 1000 ms.

A thread is started with a method (
known as start() method.
run() - executes the program body
sleep() - to delay output

ode -

```
class A extends Thread {
    public void func() {
        Sout("This is thread class");
    }
class B {
    public static void main(String args[]) {
        A obj = new A();
        obj.func();
    } Sout("This is main method");
}}
```

Agar method run, K naam se
hota to hum Thread class me
predefined Run method Ko override
krte. We can call Run method
by obj.start() with Run method
case. Lekin Khali func nya func
bnaya h isliye need to use start.

main() Ka Kaam h class Ka obj bna K
baaki body execute Kraa.

Note: However we dont know which sout will be executed first becoz they are Running concurrently.
To run sequentially use ~~obj~~
Thread.sleep(1000);

---

2) Implementing Runnable Interface
⇒ in this we need to make Thread class object to run start().

Java runnable is an interface used to execute code on parallel/concurrent threads. It is implemented by any class

we can override run() method of Runnable interface ~~in~~ in class which implements Run.

We need to pass object of class to thread class object for Refrence.
Then run run() method using start method.

Runnable interface contains single method called run().

```
class A implements Runnable {
    public void run() {
        Sout("This is Child Thread");
    }
}
class B {
    public static void main(String args[]) {
        A obj = new A();
        Thread t = new Thread(obj);
        t.start();
        (Sout(" Main class");
    }
}
```

Life Cycle of Thread

A thread in java is a direction or path that taken to execute a program.
It is an independent from within same adress space.

life cycle of thread is basically the state transitions of a threads that starts from its creation to end of thread.
It involves 5 stages:

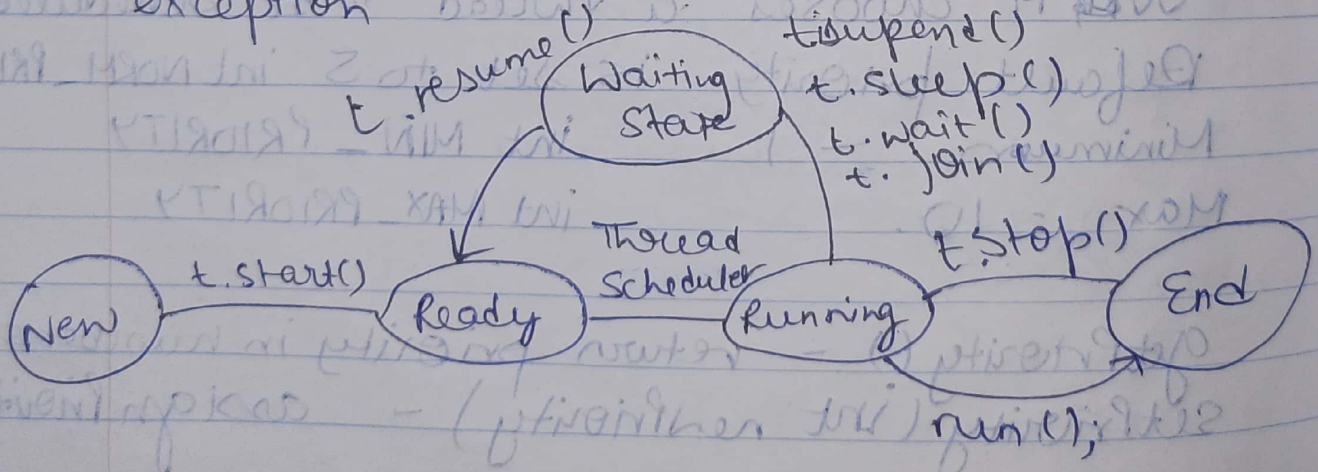
New state (Born) - when a thread is created, it is in new state. after start() is goes to Ready stage.

Ready Stage - A thread that is ready to run, is stored in this stage also called active state.
When thread scheduler selects thread & move it to Running state.

Running state - when thread gets CPU(), it is in Running state. Run() is executed

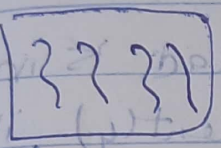
Block / Waiting - When thread is inactive for a span of (not permanently), it is in waiting stage. If it is send to waiting using t.suspend, it can be brought back only by t.resume. Baki function ne apne app resume hojata h.

Dead State - when it has finished its job or Abnormal termination due to exception



Single Threaded Process → Single connection
It contain execution of instruction in a single sequence. In other words, one command is processed at a time.

Multi threaded Process → Multiple connection
This allows executing of multiple parts of program at same time.



Thread Priorities

Each thread has a priority represented by a number between 1 and 10.
Mostly, thread scheduler schedules a thread based on priority but sometimes JVM chooses a thread to be scheduled
Default priority is set to 5 int NORM_PRIORITY.
Minimum - 1 int MIN_PRIORITY
Max - 10. int MAX_PRIORITY

getPriority() - return priority in numbers
setPriority(int newPriority) - assign Priority

```
public class T extends Thread {
    public void main (String args[]) {
        T t1 = new T();
        T t2 = new T();
①       sout ("Priority t1 =" + t1.getPriority());
②       sout ("Priority t2 =" + t2.getPriority());
        t1.setPriority(3);
③       Sout ("Priority changed t1" +
                t1.getPriority());
④       Sout ("Priority thread Running =" +
            Thread.currentThread().getName());
    }
}
```

⇒ Priority t1 = 5
Priority t2 = 5
Priority changed t1 = 3
Priority thread Running = main

Synchronization

It is the capability to control the access of Multiple threads to any shared Resource

Purpose - to overcome problem of Multithreading when Multiple threads access same resource at same time, it may provide Wrong result, so synchronization is used.

Type → Process Sync
→ Thread sync

Synchronized block in Java marked with synchronized Keyword.

two categories
Method Level synchronization
Block level synchronization (best)

1) In this if t1 thread has acquired a resource say it is running run(), then t2 & t3 ... tn will wait & get resource if left. Prevents wrong output problem
ex- if bank has 500, in multithreading 2 ek sath aae to sabka 500 milja toh, lekin it is wrong kyuki h to 500 hi, islie sync. m ek ek krke aaenge, or agr t1 n 500 mese kuch choda to t2, t3 ko milega

2) ~~Block level~~ ~~Effect~~

ex-

```
public synchronized void run(){
    // same resource
}
t1.start();
t2.start();
```
] enters sequentially

2) Block

public void hotel1() {
synchronized (this) {
// Same resorce
}
}

In this thread enter to krjava hotel() m liken isme method kar der aake wait krenge resource ka

Synchronization within a method

JAVA Libraries

It is a collection of classes that have been written & stored already
we can easily import them and use

Java Standerd Libraries

→ Java.util → Java.lang → Java.math
→ Java.io → Java.lang

→ Mockito → Junit → Apache
→ Google Guava → Jackson
→ HTTP libraries

String Handelling

String are sequence of character enclosed within quotes

String Str1 = "Hello";
datatype var = value

String S1 = new String ("hellow");
or
char [] Strarray = {'h', 'e', 'l', 'l', 'o'};
String S1 = new string (Strarray);

Functions | String Str1 = "Hello";

1) CharAt() - Return char at given part
Sout (str1.charAt (1)); ⇒ e

2) Concat() - combine
Sout (s1.concat(s2));

3) copyValueOf() - copy string into other
Str2 = str2.copyValueOf (Str1, 0, 4);
⇒ Hell

4) Replace str.replace('e', 'E');

5) toLowerCase str.toLowerCase();
Str.toUpperCase();

Applet in Java

- Small java prog used in Internet Computing
- Can be transported from one comp to another
- Run using applet viewer or any web browser that support java.
- It can perfor calculation, display graphic, play sound, accept input, etc.

It has enabled interactive Multimedia webdocuments
Significant impact on WWW
Designed to execute within webbrowser

→ An applet is a Java program that can be embeded in a webpage. Runs inside web browser & works at client side
It is embeded in HTML page using APPLET or object tag
Used to make website dynamic & entertaining.
Used to create interactive applet

- All applets are part of java.applet.Applet class.
- These are not stand-alone prog
- Execution of apple does not begin at main() method.
- Output is not performed by Sout()
Rather by AWT method such as drawString

Types of Applet

Local Applet

Developed locally & stored in a local system. When a webpage trying to find a local applet, it does not need internet as stored in local system. It simply searches the directories locally & loads the applet.

Remote Applet

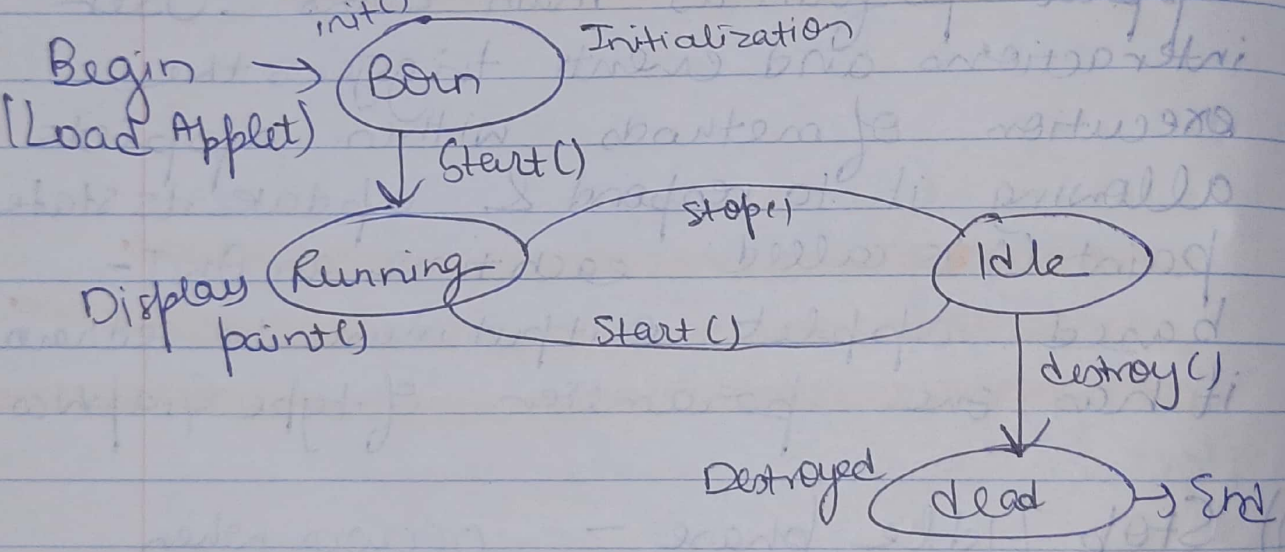
Developed by someone else & stored on a remote comp connected to internet. If system is connected to internet we can download applet.
To load a remote applet we need its URL.

App v/s ~~main method~~ Applet.
Applet don't have main method, dont run indepently, need a webpage
It cannot read/write to a file in local system
It cannot communicate with other servers
cannot use libraries from other lang such as [c/c++]
cannot Run a prog from local comp

Lifecycle

The life cycle of an applet refers to various stages that an applet goes through from initialization to termination.

1) Initialization state
2) Start
3) Running
4) Stop
5) Destroy



1) Initialization Stage : During this Stage, applet is loaded by browser or applet container. Resources are allocated & necessary initialization code is executed. init() is used to perform initialization.

This method is only called once during runtime of applet.

Start: Once applet is initialized, it enters start stage, the applet is visible and start executing its main functionality.
start() - used to begin execution of applet called each time an applet's HTML doc is shown on screen.

Running - This is active stage where applet perform its main tasks. User interactions and events triggers the execution of methods within applets allowing it to respond & updak its state paint() is called each time an AWT-based applet output must be redrawn it has one parameter of type Graphics.

Stop / Idle phase - occurs when applet loses focus or is explicitely stoped by user. Applet is no longer visible or actively running.
stop() is called to pabsed or halt for - ex - going to another page.

5) Termination/ Destroy - it marks end of lifecycle. occurs when applet is not needed. destroy() is called to release resources held by applet.

Java plug-in software is responsible to maintain lifecycle

Q How to run an applet?
1) By html file
2) By appletViewer (for testing purpose)

Simple Code

```
import java.applet.Applet;
import java.awt.Graphics;
public class first extends Applet {
    public void paint(Graphics g) {
        g.drawString("welcome", 150, 150);
    }
}
```

In myapplet.html

```
<html> <body>
<applet code="First.class" width="300" height="300">
</applet>
</body>
</html>
```

or

In cmd appletviewer Run.html

earliest framework of java

AWT in Java

Abstract Window toolKit is an Application Programming Interface for creating Graphical User Interface. (GUI) API
Allows java programmers to build develop Window based applications

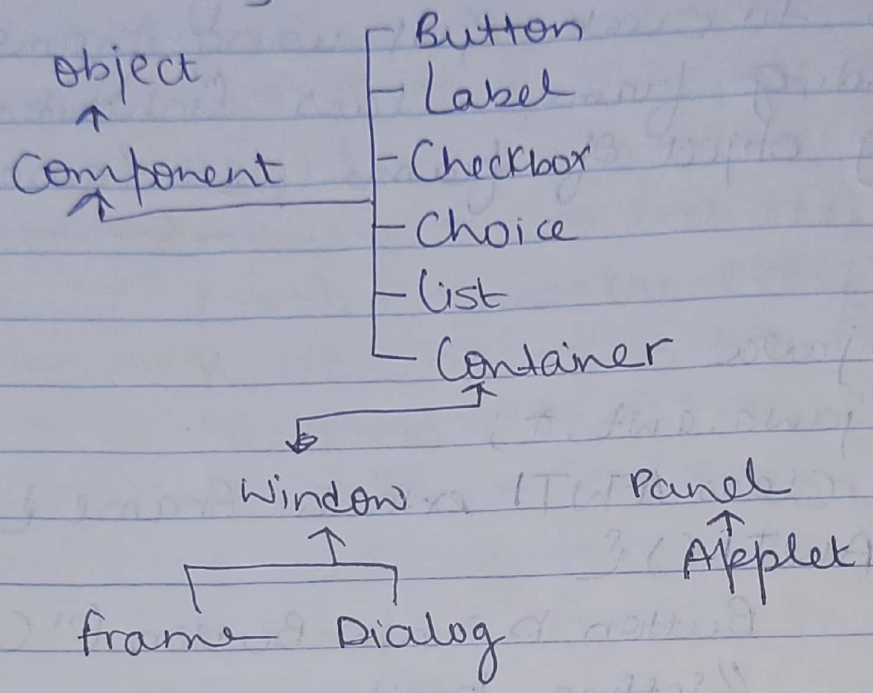
It provides components like button, label, checkbox, etc used as objects inside Java program. placed in container

Java AWT comp are platform dependent and are heavy neight i.e components are using resources of underlying (OS). partialy

java.awt package provides classes for AWT API such as TextField, TextArea, Choice, List, etc.

Why Platform Independent)
Java AWT calls the native platform (os) subroutine for creating API component like buttons, etc.
AWT GUI looks different in different OS because diff platforms have different view for their native components & AWT directly calls the native subroutines that create those components.

Hierarchy



- Component → all elements like buttons
- In order to place every component in a particular position on screen we need to add them in container

- Container → is a component in AWT that contain other components
- The class that extends container class are known as container such as Frame, Dialog & Panel.

Types - Window, Panel, Frame, Dialog

Creating AWT

To create AWT, we need frame
2 ways to create GUI using frame
extending frame class (inheritence)
creating object of frame class (association)

AWT1.java

```
import java.awt.*;
public class AWT1 extends Frame {
    AWT1(){
        Button b = new Button("ClickMe");
        //Setting position
        b.setBounds(30,100,80,30);
        add(b);
        setSize(300,300);
        setTitle("Example of AWT");
        setLayout(null);
        setVisible(true);
    }
//main method
public static void main(String args[]){
    AWT1 f = new AWT1();
}
}
```

2) AWT2.java

```
import java.awt.*;
class AWT2 {
    AWT2 () {
        Frame f = new Frame();
        Label l = new Label("Empid:");
        Button b = new Button("Submit");
        TextField t = new TextField();
        l.setBounds (20, 80, 80, 30);
        t.setBounds (20, 100, 80, 30);
        b.setBounds (100, 100, 80, 30);
        f.add (b);
        f.add (l);
        f.add (t);
        f.setSize (400, 300);
        f.setTitle ("Info");
        f.setLayout (null);
        f.setVisible (true);
    }
public static void main (String args[]){
    AWT2 a= new AWT2();
  }
}
```

See example of List, Checkbox, etc.

```
List l1 = new List (5);
l1.add ("item1");
```

```
Checkbox c1 = new Check
Checkbox c2 = new Checkbox
    ( ) true);
```

Layout Manager

It is an object that controls the size and position (layout) of components inside a container object

It is used to arrange components in a particular manner

It facilitates to control positioning & size of components in GUI forms

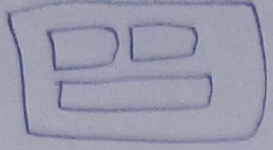
It is an interface that is implemented by all the classes of layout Managers

java.awt.BorderLayout - to arrange in five region (north, west, south, east, center)
It is default layout of frame or window

FlowLayout - to arrange comp in one line. It is defaut layout of applet/panel
(left, Right, Center, Leading, trailing)

Box Layout - arrange comp vertically or horizontally
(x_axis, y_axis, line_axis, page_axis)

Card layout - manage component such that one is visible at a time
It treat each component as a card.

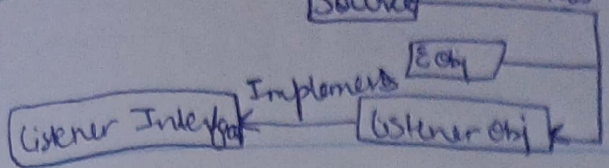


GridBagLayout
to align components vertically, horizontally or along their baseline.

GroupLayout
group its component & places them in a Container hierarchically.
Grouping is done by instances of Group class

Spring Layout
arranges the children of its associated container acc to a set of constraints
Constraint are nothing but horizontal & vertical distance b/w two component edgs.

ScrollPane Layout
Layout manager is used by JScrollPane
JScrollPaneLayout is responsible for nine components:
a viewport, two scrollbars, a row header, a column header, & four corner components.



Event Handelling in Java

A mechanism that controls the events and decides what should happen if event occurs

Event - Changing the state of an object can be done by performing actions such as a button click, cursor movement Keypress, page Scrolling.

Classification of Events

Foreground - events that require user interaction to generate (in GUI)

Background Events - that don't require user interaction to generate

Ex - OS failiure, Operation Completion, etc

To handle events Java follows Delegation Event Model.

is defined to handle events in GUI programming languages.

Java support event processing since Java 1.0. Providing support for AWT

DEM

Working
In this model, a source generates an event & forword it to one or more listeners. The lister wait till it hears an event. Once received, it is processed & returns it.
UI elements are able to delegate the processing of an event to separate functi

Ad → Application logic is completely seperated from interface logic.

An event Model is based on
- Events
- Events Sources
- Events Listeners

Events objects that define State change in source. Generated as a reaction of user while interacting with GUI

Event Sources object that causes & generate an event it generate event when internal state of object is Changed.
Sources are allowed to generate diff events.
Source must register a listener to receive notification of a event

DEM

public void addTypeListener (TypeListener e1)

Event Listeners

object that is invoked when an event triggers. It require two things, first, must be registered with a source; however can be registered with several resources to receive notification about event.

Syntax –
addTypeListener()

// See code

```
public static void main (String args[]) {
    new AEvent();
}
}
AEvent() A = new AEvent();
```

to get input
String str = e.getActionCommand();
tf.setText(str);