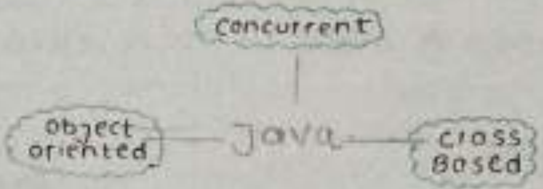
Java cowise By Gowrav

- Java is an object oriented programming longuage devloped by Sum Microsystems of USA in 1991

It was originally called Oak by James GosLin
↳ one of the inventer of "Java"

Java = Purely object oriented

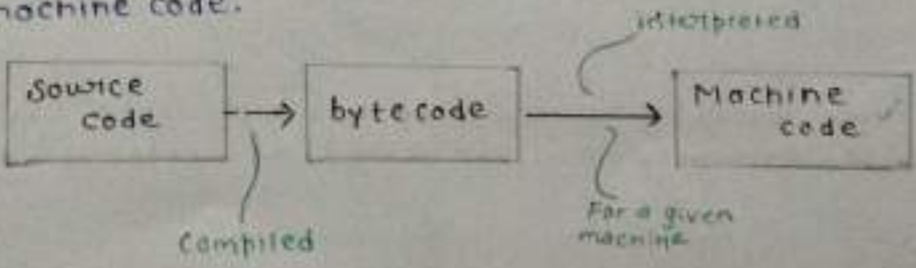
- Java is a class-based object-oriented simple programming language. However, we can't not considerd it to be fully-object-oriented as it supports primitive datatype. It is a genral-purpose. high-level programming language that help programmer and devlopers to write code once and run it anywhere.



→ We can call it a high-level programming language (which make program devlopment easy to much more user-friendly)
→ Java is class-based object-oriented programming lang that implement the principle of write once code anywhere
→ Java application can run on any JVM-support machine since they are compiled to byte-code.
→ Java code very similar to C/C++. which make eaiser to understand.

HOW JAVA WORKS?

Java is compiled into bytecode and then it is interpreted to machine code.



OOPs (object-oriented Programming)

Object oriented Programming or OOPs refer to language that use object in programming. Object-oriented programming aims to implement real-world entities.

→ (OOPs) is a methodology that simplifies software design by modeling real-world entities as object It emphasize the use of reusable components, making programs modular, maintainable, and scalable.

→ Java is an object-oriented programming language that implement OOP principle effecitivly.

OOPs Concepts :-

(or Benifit of OOPs)

1. Class :- A Class is user-defined data type. It consist of data member function. which can be accessed and used by creating an instance of that class. It represent the set of properties or method are common to all object of one type. A class like a blueprint for an object.

Object

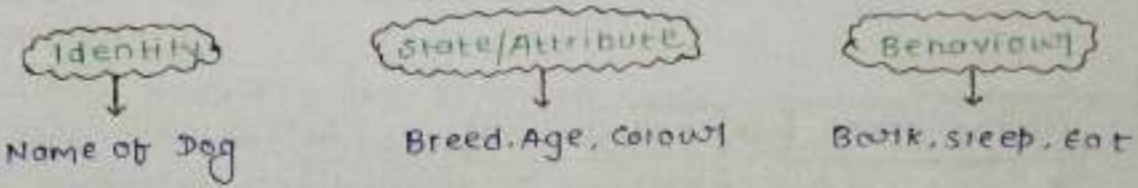
```
class Car {
  String color; // Attribute
  String Model;
  void drive() { // Behaviour
    System.out.println("The car is driving.");
  }
}
public class Main {
  public state void main(String[]args) {
    Car car = new Car(); // create an object
    Car.Color = "Red"; // Assign date
    car.drive(); // call behaviour
  }
}
```

Ex:- in the real world: A "car" object has attributes like color and model and behaviour like drive() and break()

2. Object :- It is a basic unit of object-oriented programming and represent the real-life entities. An object is an intance of a class. When a class is defined, no memory is allocated but when it is intanited (i.e object is created memory is allocated. An object has an identity, state and behaviour.

→ Each object contain data and code to manipulate the data. Object intract without having to know detail of each other's data or code.

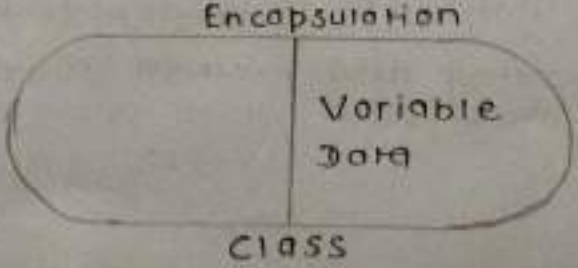
Ex:- "Dog" is a real-life object, which has some characteristics like colour, Breed, Bark, sleep, eat



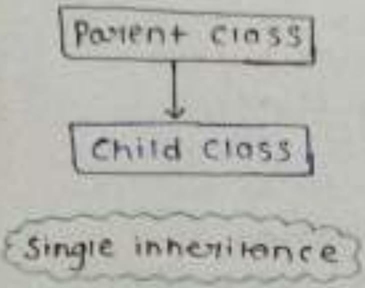
3. Data Abstraction :- Data Abstraction is one of the most essential and important feature of object-oriented programming. Data Abstraction refer to providing only essential information about the data to the outside world, hiding the background detail or implementation.

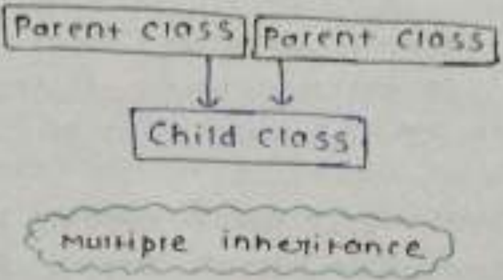
4. Encapsulation :- Encapsulation is defined as the warpping up to data under single unit. It is the mechanism that bind together code the data manipulate. In Encapsulation, the variable of data of a class are hidden from any other class and can be accessed only through any member function of their class in which they are decleared.

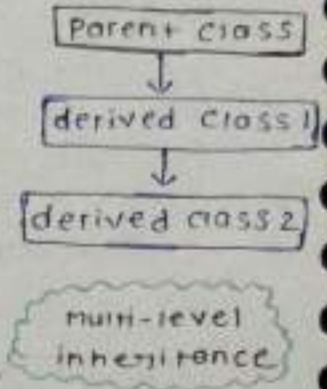
→ As in encapsulation, the data in the class is hidden from other classes, so it's know as data-hiding.

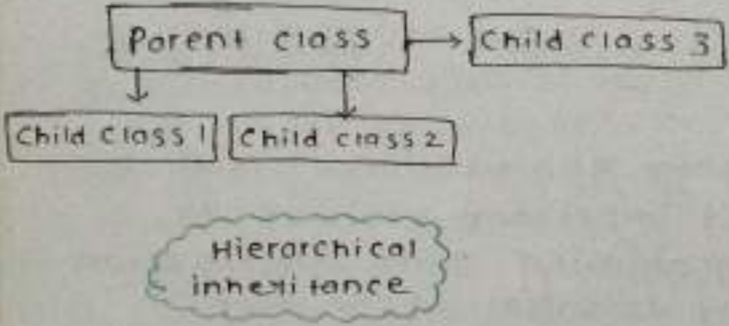


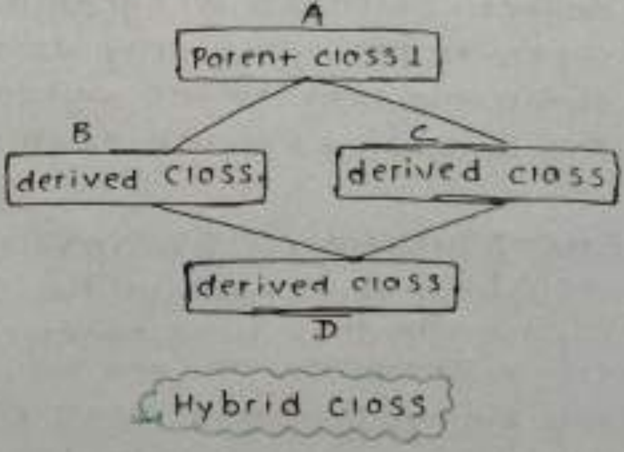
5. Inheritance :- Inheritance is an important pillar of OOP(object-oriented-Programming). The capability of class to derive properities and characteristics from another class called Inheritance.



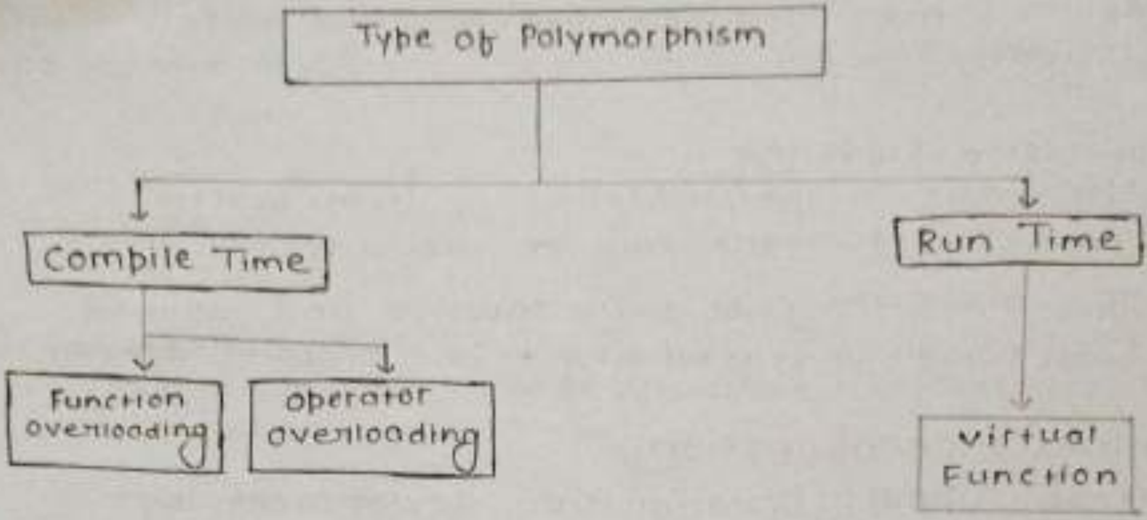








6. Polymorphism :- Polymorphism mean "many forms" It allow object to perform diffrent behaviour based on their context. It is achived through method overloading and method overriding.



Benefits of OOPs in Java

1. Modularity
   - → OOP promote the division of a software program into distinct module or classes. each representation a specific component or functionality.
   - → This modularity make the code eaiser to manage maintain. and understand.

2. Reusability
   - → OOP encourage code reuse through inheritance and Composition
   - → By creating new class from existing once. devloper can leverage pre-existing code, reducing redudancy and efforts.

3. Improved Maintainability
   - → OOP's modularity and encapusulation make updating. modifying. and debugging code eaiser.
   - → change in one part of the systum have minimal impact on other parts. enhancing maintainbility.

4. Ease Of Troubleshooting
   - → OOP simplifies the process of debugging and troubleshooting by isolating funcationality into seprate classes.

→ Problum can be localized and resolved more efficiently.

5. Real-World Modeling
→ OOP helps in the modeling of real-world entities and relationship in software
→ This make the code more intutive and aligned with human understanding of the problum domain.

6. Enhance Colloboration
→ OOP support collaborative devlopment by enabling multiple devlopers to works on diffrient classes or module simultaneously.
→ Well-defined interface and encapsulation improve teamwork and integration.

7. Extensibility
→ OOP'S principle make it eaiser to extend and scale software system.
→ New Feature and functionilites can be added with minimal disruption to existing code.

Application OOP'S

→ Object-oriented Programming (OOP) is widly used in software devlopment due to it's modular and resuble design approch.

1. Graphical user interface (GUI)
→ used to build graphical interface like Java swing and JavaFx

2. Web Devlopment
→ Framework like spring (Java) and Django (Python) rely OOP.

3. Gaming
→ OOP helps design game element like character and logic (eg. unity, Unreal Engine)

4. Mobile Apps :-
→ Android devlopment uses OOP's longuage like java and kotlin.

5. Enterprise software
→ large-scale system like ERP and CRM use OOP for scalbality.

6. Embedded system
→ used in IOT device and automotive system for modularity.

Java History

1. Devloper
→ created by James Gosling and team at sum Microsystem in 1991
→ initally name Oak, later renamed Java (inspired of java

2. Purpose
→ Designed for interactive television system but found to be too advonce
→ Aimed to create a platform-independent simple, and secure programming longuage.

3. ~~kile~~ key Milestone
→ 1995 : officialy release of Java 1.0 by sun Microsystem
→ 1997 : Java acquire by oracal corporation
→ 1998 : Java 2 introduced with edition like J2SE. J2EE. J2ME
→ 2004 : Java 5.0 introdused feature like genric annotations, and autoboxing.
→ Recent update : Java 8 (stream, lambdas) and java 17 (long-term-support)

Java Feature

1. Simple :- Java is very easy to learn, and it's syntax is simple, clean and easy to understand According to Sun Microsystem

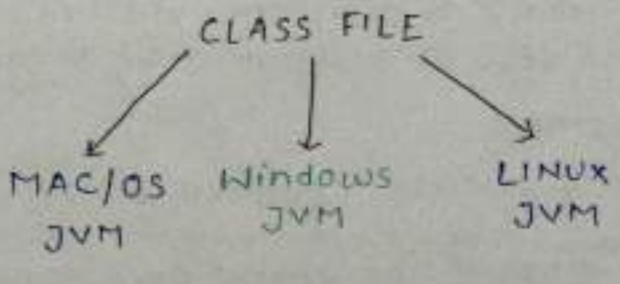
   → Java syntax is based on C++ (so eaiser

   → Java has removed many complicated and rarely-used feature, for example explict pointer, operator overloading.

   → There is no need to remove unreferenced object because their is an Automatic Garbage.

2. Platform independent :- Java is platform independent because it diffrient from other language like C, C++ which compiled into platform specific machine while Java is write once, run anywhere language.

   → There are two type of platform software-based and hardware-based Java provided a software based platform.



3. Secured :- Java is best knows for it's security with Java, we can devlop Virus-free systums. Java is secured because.

   - No explict pointer
   - Java program run inside a virtual machile Sandbox.

4. Object-oriented :- Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different type of object that incorporate both data and behaviour.

5. Portable :- Java program can be easily moved from one system to another
→ No dependency on platform-specific feature.

6. Distributed :- Support distributed computing using technologies like ~~RAM~~ RMI (Remote method invocation) and ~~EB~~ EJB (Enterprise Java Beans)

7. Robust :- The English mining of Robust is strong Java is robust because:
→ It uses strong memory managment
→ There is lack of pointer that avoids security problum.

8. Multi-threaded :- A thread is like a seprate program, executing concurrently. We can write Java program that deal with many task at once by defining multiple threads.

9. Ease to devlopment :- Provide a rich API and a vast ecosystem of libraries.
→ Tools like IDE's (Eclipse, Intellij), simply devlopment.

Java vs C

Java (HIGH LEVEL)	C (LOW LEVEL)
→ Object-oriented language for cross-platform application	→ Procedurial language for system-level programming
→ Object-oriented programming focuses on class/object	→ Procedural programming focuses on function.
→ Platform-independent; run on JVM ("write once run anywhere)	→ Platform dependent; depend to machine code.
→ Automatic memory managment via garbage collection.	→ Manual memory managment using malloc and free
→ Slower due to JVM overhead and abstraction.	→ Faster execute as it compile to machine code.
→ Does not support pointer directly for safety	→ Support pointer for memory acess and manipu
→ Used for web, mobile app and enterprise software	→ Used for OS devlopment embedded system, and drive.
→ Rich standard libraries for various functionalitie	→ Fewer built in libraries relies on external libraries

C++

- → C++ is platform dependent
- → C++ is mainly used for system programming
- → C++ was designed for system and application programming. It was extension of the C lang
- → C++ supports multiple inheritance
- → C++ supports operator overloading
- → C++ supports pointer. You can write a pointer program in C++
- → C++ uses compile only C++ compiled and run using the compiler which convert source code into machine code so, C++ is platform dependent.
- → C++ supports structure and union.
- → C++ always create a new inheritance tree.

Java Enviorment

↳ consist of various tool, libraries, component required for deviop. debugging. and execute

JDK → JAVA Deviopment kit = collection of tool used for deviloping and running java program

JRE → Java Runtime Enviorment = Help in executing program deviloped in java

JAVA Deviopment kit

↳ (JDK) is a cross-platformed software deviopment enviorment that offer's collection of tool and libraries neccesary for devioping Java-based application and applets.

→ It is core package of java

components :-

a. Java compiler :- Convert java code into bytecode
b. Java Runtime Enviorment :- Includes JVM, libraries and other component for running java application.
c. Java Debugger (Jbd) :- Helps in debugging java program
d. Java documentation Tool (Javadoc) :- Genrates documentation from comment in the code.
e. Additional tool :- Jar, Javap. etc.

Java Deviopment tool :-

a. Javac : Java compiler for compilling .Java files to .class file
b. Java : Java interpeter for executing bytecode (.class file)
c. Java doc : Tool ~~of~~ for genrating API documentation.
d. Jbd : Debugging tool for java program
e. Jar : Tool for packaging Java classes into .Jar class
f. Javap : Diassembler tool for inspecting.
g. Javaxpackage : Tool for packageing JavaFx application.

Application Programming Interface (API)

→ The (API) is a collection of pre-written package classes and interfaces provided by Java simply programming These are grouped into sevral package based on their functionalities. (Applet package use for ~~sum~~ Run small program on ~~[illegible]~~ browser.

- Language support package
  - Provide classes and interfaces for basic language feature.
  - contain fundmental classes requrie for Java program
  - key classes
    - Java.lang : Automatically imported in every Java program
    - classes : object, Math, string. Thread, etc

- Utilities Package
  - Provide classes for data structure. utility operations and collections framework.
  - used for task like sorting. searching and managing data.
  - key classes → ArrayList, HashMap, Data, collection
    - Java.io
    - classes : File. BufferRedar. Printwriter. etc

- Input/output Package
  - supports input and output (I/O). operations in Java
  - provided classes for file handling, reading. and writing data
  - key classes
    - Java.awt
    - classes : Buttom, Label, Frame, Panel, etc

- AWT (Abstract window Toolkit) Package
  - Provide classes for creating graphical user interface (GUIs)
  - Include component like buttom, window and menus
  - key classes — Java.awt

Applet package
- Java applet
- classes Applet. Applet. Context, etc

- Networking package
  - Provide classes for Network programming (eg. connecting to server, sending, reciving)
  - Enable handling protocals lice TCP and UPD
  - key classes } URL. socket. serversocket, etc.

Simple java program

1. Class declaration
   - Java is oop. and all code must be insite ~~code~~ class.

Syntax:- class ClassName{ }

```
Public Class My Class {
  // code
}
```

2. opening & closing Braces
   - curly braces {} are use to define the boundries of classes. method. and block of code.
   - Every opening braces { must have a corrosponding closing braces }.

```
Public Class My Class {
 Public static void main(string[] args) {
   System.out.print("Hello world");
 }
}
```

3. Main Line
   - Syntax:- public static void main(string[] args){...}
     >public : Make it accesible to the JVM
     >static : Allow the method to run without creating a job
     >void : Indicate no return value
     >string[] args : Array to accebt Command line-argument

```
Public static void main(string[] args) {
  // code
}
```

4. Output line
   - To display output. java uses the system.out.printn () method
   - System : A built-in class

out : Represent the standerd output stream

println : Print a line to text and move to the next-line.

```
system.out.println("Hello world");
```

5. creating an object

- object are instance of the class, created using the new keyboard.
- Syntax :- ClassName = new Class Name ();

```
My class obj = New Myclass();
```

NOTE :- object are use to access non-static method and variable.

1.12 use Math function in java

- Math pow (a, b); calculate a^b.
- Math. sqrt (x) : Return the square root of x;
- Math.abs(x) : Return the absolute value of x;
- Math.max(a, b) : Return maximum

```
Public class Mathexample {
    Public static void main (String[] args) {
        int a=5, b=3;

        system.out.println("Power:" + Math.pow(a,b)) ; //5^3
    =125.0
        system.out.println("square root :" + Math.sqrt(25));
    √25 = 5.0
```

comment in java

- Single Line Comment : Start with // and extended to the end of the line.

```
// This is single line comment.
```

- Multi-line Comment

```
/*
content
*/
```

1.13 Java program structure

1. Document section :- Contain comment or documentation about the program.

```
/**
 --
**/
```

2. Package statement :-
   - Define the package to which the class belong
   - A package organize relate class and interface into directory structure.
   - syntax : package packageName;

3. Import statement
   - Allow the use of pre-defined classes from other package or user-defined package.
   - Syntax : import packageName.class Name; or import package Name.*;

```
import Java.util Scanner;
import java.util.*
```

4. Interface Statement
(collection of abstract method
syntax:- interface InterfaceName {...}

```
interface My Interface {
    void display Message();
}
```

5. class
- Define the structure and behaviour
- Can incude field, methods, constructors, and nested class

syntax;

```
class classname {
    //Field
    //Method
}
```

6. Main Method Class ⌣ primary class of program
- The main method serve as the entry point of the program
- syntax:

```
Public static void main(String[] args) {
        //program executed starts here
}
```

Java Token

1. Reserved keyboard
→ Predefined word in java that have specific meaning and cannot be used as identifiers

Ex:- variable name, method name, etc

→ This keyboard are case-sensitive and must be used in correct syntax.

Ex:- class, public, static, void, int, if, else, for while, return, etc

2. Identifiers

→ (are name given to variable, method, classes, interface and other program element.

· Rule of Identification

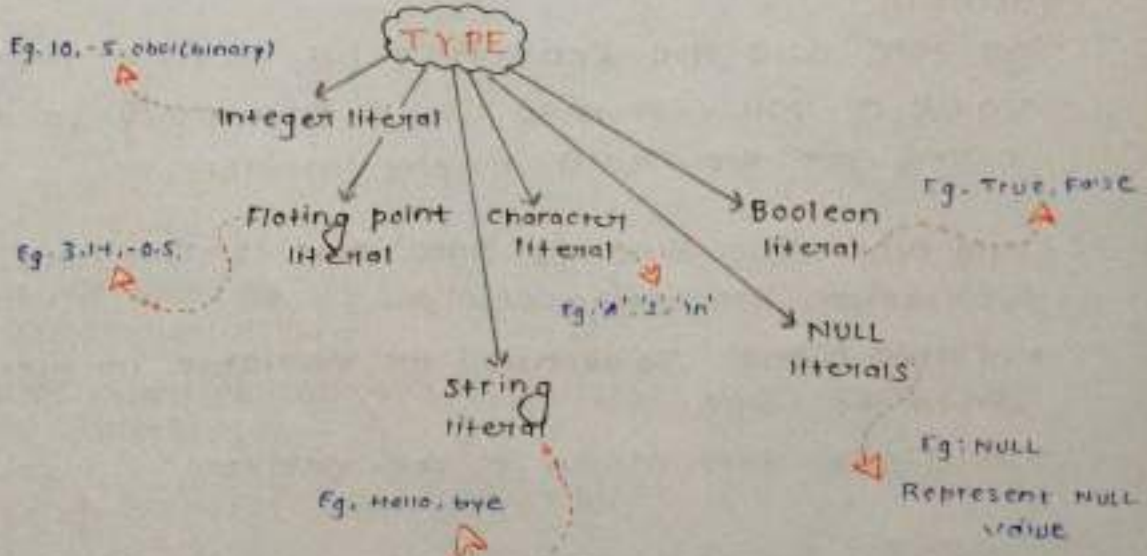
1. Must start with letter (A-Z, a-z), underscore (_) or dollar sign ($)
2. Can be followed by letter (A-Z, a-z), digit (0-9)
3. Cannot be reserved keyboard
4. Are case-sensitive.

· Example

· Valid : myVariable, _count, $ salary, MAX_VALUE

3. Literals

(are constant value assigned to variable or used directly in the code.



4. Operator - are symbols that perform operations on variable and values.

1. Arthmatic : +, -, *, /, % (module)
2. Relationl : ==, !=, >, <, >=, <=
3. Logical : $$ (AND), || (OR), ! (NOT)
4. Assigment operator : =, +=, -=, *=, /=.
5. Bitwise : &, |, ^, <<, >>, >>>
6. Tenary : ?: (conditional)

1.19 : Constant $ Variable

Constant :

- A constant is a variable whose value cannot be changed once it is assigned.
- In java, constant are decleart using the final keyboard.

• Syntax

```
final data-type CONSTANT_NAME = VALUE
```

• Example

```
final double PI = 3.14159;
   final int MAX_VALUE = 100;
```

Variable

- Variable are the container for storing the data value or you can also call it a memory location name for the Data. Every variable as.

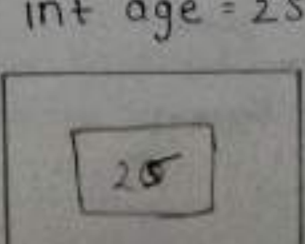
- Data type :- The kind of Data that it can hold. For example, int. string, Flot, char
- Variable Name :- To identify the variable uniquely within the scope
- Value :- The data assign to the variable

Syntax

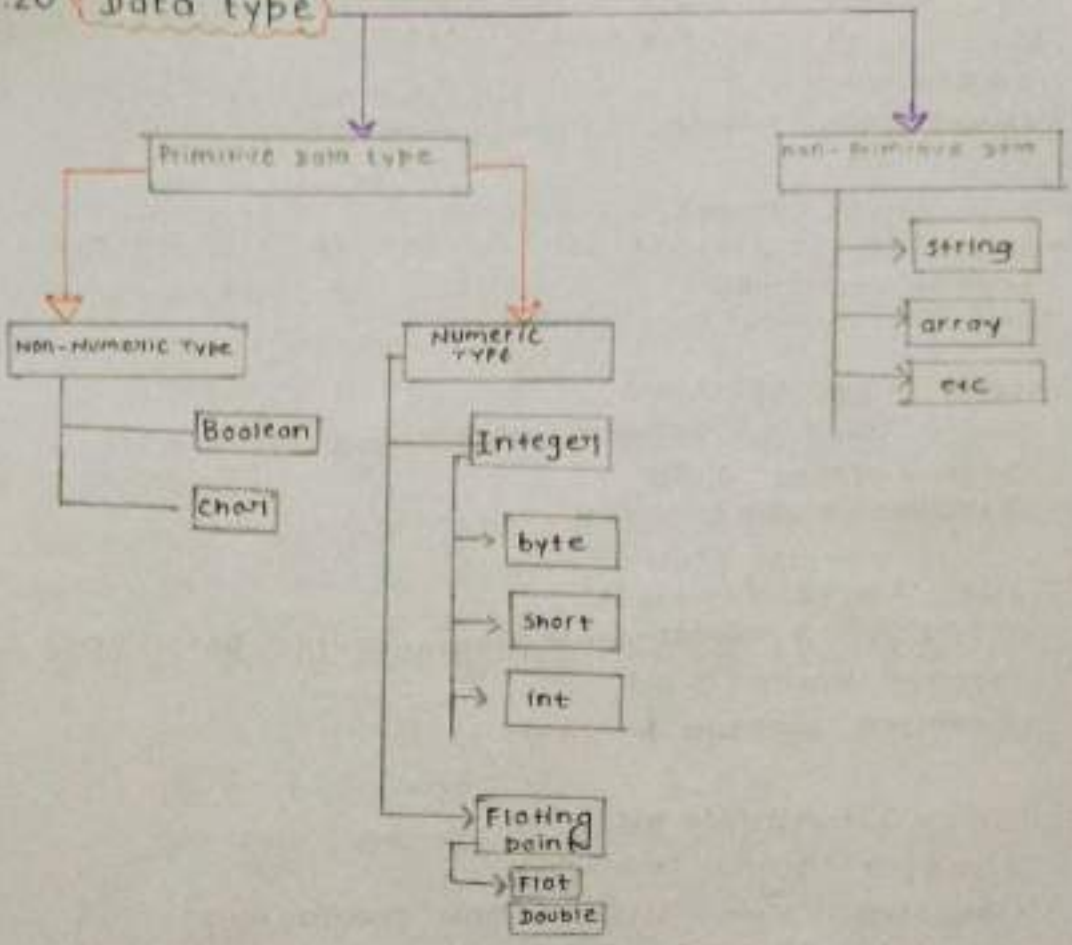
Ex:- data_type variable-Name = value;

Example

int age = 25;   int age = 25



1.20 Data type



1. Primitive Data Types:

Primitive data type are the most basis type of data that are prededefined in java

These types represent simple values such as number and characters.

List of Primitive Data Types:

1. byte: · size 1 byte (8-bits)
   · Range: -128 to 127
   · Default values: 0
   · Example: byte a-100;

2. Short: · size : 4 byte (32-bits)
   · Range: -2,147,483,648. to 2,147,487,648
   · Default values: 0
   · Example short b=32000;

3. INT: • Size : 4 byte (32 bits)
• Range : -2,147,483,648 to 2,147,487,648
• Default value :- 0
• Example : int c = 1000;

4. long: • Size : 8byte (64bit)
• Range : - 9,223,372,036,854,775,808. to 9,223,372,036,854
• Default value :- 0L
• Example :- long d = 1234567890L;

5. Float: • Size : 4 byts (32-bits)
• Range : used for Decimal value (single precisions)
• Default value : 0.0f
• Example : Float e = 3.14F

6. Double : • Size : 4 bytes (64 bits)
• Used for : Decimal value (~~signal~~ double precisions)
• Default value : 0.0d
• Example : double F = 3.14159;

7. Char: • Size : 2 bytes (16 bits)
• Used for : signal character
• Default value : '\u000' (null charactor)
• Example : char g = 'A';

8. boolean: • Size : 1 bit
• Used for : True/False values
• Default values : false
• Example : boolean h = Itue;

2. Non-Primitive Data Types:

Non-Primitive data Type, also know as refrence type, are used to store more complex structures. These are object created from class.

2.1 Strings:

• Strings is a sequence of charaacter, and it is represented as an object in java.

• Strings are immutable, meaning that their content be changed once created.

Example :

```
String message = "Hello, Java!";
```

String operation can include :

- Concatenation : "Hello" + " " + "World"
- Length : message.length()
- Substring : message.substring(0,5)

2.2 Arrays:

- An array is collection of variable of the same type, store in contiguos memory locations.
- Arrays have fixed size after they are created.

Example

```
int [] numbers = {1,2,3,4,5};
String [] name = {"Alice", "Bob", "Charlie"};
```

2.3 Variable & Constant

↳ Explained in previous page

1.21 OPERATORS

1.23 Decision Statement

a. If-statement:

The if statement runs a block of code only if a condition is true.

Example:

```
if (a >b) {
    printf("a is greater than b");
}
```

It is used when you want to perform an action only if a condition is met.

b. If-else Statement

The If-else statement let you choose between two blocks of code : one if the condition is true, and one if it's false.

Example

```
if (a > b) {
    printf("a is greater than b");
} else {
    printf("b is greater than a");
}
```

It is used when there are two parallel action to take based on condition.

c. Nested if-else statement

nested if-else statement is an if-else statement inside another if-else statement. It is used when you need to cheak multiple condition. in a hierichal mannar. Each if or else block can contain another if-else statement, allowing for more complex decision making.

Syntax:

```
if(condition1){
  // Code to execute if condition1 is true
  if(condition2){
    // Code to execute if condition2 is true
  }else{
    // code to execute if condition2 is false
  }
}else{
   // code to execute if condition 1 is false
}
```

d. If-else Ladder

The if-else-if ladder cheaks multiple conditions one by one unit it find a true conditions or reached the end.

e. Switch statement

The switch statement is a cleaner way to handle multiple possible condition by cheaking one variable against many option.

```
switch(day){
    case1: printf("Monday"); break;
    case2: printf("Tuesday"); break;
    case3: printf("invalid day");
}
```

It is used when you have many diffrent option to cheak for. like menu choise.

1.24 Loop statement

→ Loop are used to repeat a set of action multiple time., like running a task over and over until a condition changes.

a. While loop:

→ A while loop repeat an action an long as a condition is true. you might not know how many time it will repeat.

Example

```
int i=0;
While (i<5){
  print("%d", i); //Print 01234
   i++
 }
```

It's usefull when you ~~kn~~ don't know how many repetitions you need. but just want to keep going until something change.

b. do-while loop:

→ A do-while loop is similar to a while loop but it always run at least once before checking the condition.

```
int i=0;
 do {
   printf("%d". i);
   i++
 } While (i<5);
```

It's used when you want to ensure the code runs at least once like showing a menu before checking the conditions.

c. for statement

- The for loop is used for executing block of code repeatedly for a fixed number of iterations. It consist of three parts: initialization, condition, and statement.

Example:

```
public class ForLoopExample {
    public static void main(string[] args){
        for(int i=1; i<=5; i++){
            system.out.println("Iteration:" +i);
        }
    }
}
```

d. For-each Statement

- The for-each loop is used to iterate over arrays or collections without using an index. It provide a simpler way to traverse element.

Example:

```
public class ForEachExample {
    public static void main(string[] args){
        int[] number = {10,20,30,40};
        for(int num : number){
            system.out.println(num);
        }
    }
}
```

1.25 Control statement

a. break

→ The break statement stop the loop completly and move on the Next part of the program.

Example:

```
for (int i=0; i<10; i++){
    If (i==5) break; //stop the loop when i is 5
}
```

b. continue

→ The continue statement skips the current iterations of a loop and move on the next one.

Example

```
For (int i=0; i<10; i++){
   if (i==5) continue; //skip printing 5
   printf("%d", i);
           //Print 0,1,2,3,4,5,6,7,8,9
   }
```

use continue when you want to skip certain steps in a loops but keeps the loop going.

c. Return statement

→ The return statement is used inside a loop to exit both the loop and method immediately. once executed, no further iteraction occurs and control goes back to the calling function.

Example

```
Public class Return Example{
    public static void main(string[] args){
       system.out.println(FindFirst Even());
    }

    Public static int FindFirstEven(){
       int[] number = {3,5,8,11};
       For(int num : number){
         return num; //Exits the method immediatly
       }
     }
       return-1; // if no even number is found
    }
  }
```

Constructors

→ Constructors in java are special method used to initilize object. They are called when an instance of class is created. constructors have the same name as the class and do not have a return type.

Type of Constructors

1. Default constructors
   - A constructor with no paramete.
   - If no constructor is defined in class, java provide a default constructor automatically.
   - Example

```
class student {
      student() {
         systum.out.println("Default constructor");
      }
   }
```

2. Parameterized constructor:
   - A constructor that take one or more parameter
   - used to initilize object with specific values.
   - Example

```
class student {
      string name;
      int age;
      student(string n, int a) {
          name = n;
          age = a;
      }
   }
```

3. Non-parameter Constructor
   - A constructor with no parameter but explicity define by the programmer
   - unlike the default constructor, it can include custom initilize logic.

{this keyboard}

→ The this keyboard in java is refference to the current ~~gob~~ object it is used to differentiate between instance variable and parameter (or local variable) when they have the same name.

2.4 {Visibility control}

1. Public
   - Accessible from any other class
   - Example : public int count;
2. Private
   - Accessible only within the same class
   - used for encapsulation.
   - Example : privet string secret;
3. Proctected
   - Accessible within the same package and subclass
   - Example : protected void display(){}
4. Default (Package-private)
   - No explict modifier. Accessible only within the same package
   - Example : void calculate(){}

2.5 {Arrays}

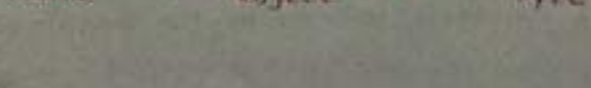
→ An array is a collection of element of the same data type store in contiguos memory location. Array allow easy access and manipulation of data using an index.

use case : storing marks of 5 student.

int [] mark = new int [5] => [datatype ArrayName]

Refrence

object



Type of Arrays

(a) One-Dimensional Array (1D Array)

- A linear collection of element accessed used a single index.

Example:

```
int[] arr = {1,2,3,4,5};
```

(b) Multi-Dimensional Array

i) Two-Dimensional Array (2D array)

- Repeat matrix (row and coleum)
- Accessed using two indices [row][coloum].

Example:

```
int[][] matrix = {{1,2,3},{4,5,6},{7,8,9}};
```

(ii) Three-Dimensional Array 3-D array

- A collection of 2D array
- Accessed using three indicies [layer, row, coloum]

Example:

```
int[][][] arr = new int[2][3][4];
// 2 layer, 3 row, 4 colum
```

3. Declaration of Arrays

→ Before using an Array, it must be declared by specific the data type and size.

Syntax:

```
data_type[] array_name; //Prefered syntax
//OR
data_type array_name[]; //Valid but less common.
```

Example:

```
int[] number; // Decleariing an inte
float mark[];
```

4. Creating an array

→ After declaration, array must be created using the new keyboard.

Syntax:

```
arrayName = new data_type[size];
```

Example:

```
int[] arr = new int[5]; //create an integer array of size 5
```

4. Initialization of Arrays

→ Array can be initilized while declaring them.

Example:

```
int[] arr = {10,20,30,40,50}; //Explicit initilization.
```

2.6 String

1. String classes

→ String are object that represent sequence of character. The two primary classes for handling string are

- String (immutable)
- StringBuffer (Mutable & Thread safe)

String class (immutable)

- once created, a string object cannot be modified
- Any modification result in a new string object.
- stored in the string constant Pool (for memory efficiency)

Example:

```
String s1 = "Hello";
String s2 = new string("World");
S1 = S1.Constant(S2); // Create a new object "Helloworld"
```

2. String Buffer (Mutable & Thread safe)

- Mutable → can be modified after creation
- Thread & safe → synchronized method (safe or multi-threading)
- Preferred for heavy string manipulations

Java

UNIT-III Inheritance, Interface and Package

Concept of Inheritance

→ Inheritance is mechanism in java that allow one class to inherit properties (fields) and behaviour (method) from another class. It promotes code resobility, improve maintability and establish a relationship between class.

KEY TERM

- Superclass - (Parent class) - The class whose properties and method are inherited.
- Subclass - (child class) - The class that inheritance relationship between two class
- Overriding - When a subclass provide a specific implementation of a method already ~~finished~~ defined in the superclass.
- Super keyword - Used to reffer to the parent class constructor, method, or fields.

TYPE OF INHERITANCE

a. Single inheritance

b. Multiple

c. Multi-level inheritance.

d. Hierichical inheritance.

a. Single inheritance

In java refer to the inheritance relationship where a subclass extends only one superclass. Here demonstrating.

```
//superclass
class Animal {
      void eat() {
           systum.out.print("Animal is eating");
  }
}
//subclass (single inheritance)
class Dog extend Animal {
      void bark() {
           systum.out.println("Dog is barking");
  }
}
```

```
OUTPUT
•Animal is eating
•Dog is barking
```

B. Multi-level inheritance

in java refrence to a senario where a class inheritance properties and behaviour from another class, which is twin inherits from another class. This create hirechical structure of class.

```
//Parent class
class Animal {
    void eat() {
        System.out.println("Animal is eating");
    }
}
//child class inheritance from Animal
class Dog extend Animal {
    void bark() {
        System.out.println("Dog is barking")
    }
}
//sub child class inheriting from Dog
class Labrador extend ~~Animal~~ Dog {
    void display() {
        System.out.println("Labrador is a type of Dog");
    }
}
```

OUTPUT

```
1. Animal is eating
2. Dog is barking
3. Labrador is a type of Dog.
```

C. Hierarchical Inheritance

in java reference to a scenario where multiple classess inherit property and behaviour from a single parent class in this inheritance structure, there is one parent class and multiple child classes that inherit from it.

OUTPUT

```
1. Animal is eating
2. Dog is barking
3. Animal is eating
4. Cat is meowing.
```

D. Hybrid Inheritance

in java refrence to a combination of multiple inheritance and hierical inheritance. In hybrid inheritance. a class is derived from two or more class. and these derived class are further have their own subclass. java does't support multiple inheritance directly due to dimond problum.

Interface

→ An interface in Java is a refrence type that define a collection of abstract method that a class must implement. It serves as a blueprint for other class and help in achiving abstraction and multiple Inheritance.

KEY FEATURE

- Interface can contain abstract method (without body) and default method (with implementation)
- They cannot have instance variable but can have static and final Constant.
- Interface are implemented by classes using the implement keyword.

Syntax of Interface

```
Interface vechical {
    void start(); // Abstract method
}
```

1. Extending Interface

→ An interface can extend another interface using the extend keywords. This allow an interface to inherit method from another interface.

Example

```
interface vechical {
    void start();
}
interface car extends vechical {
    void speedup();
}
```

2. Implementing Interface

→ A class implement and interface using the implement keyword. It must provide implementation for all the method decleared in the Interface.

```
Ex  class Bike implements vechicals {
       public void start() {
           system.out.println("Bick is starting");
       }
    }
```

Method overloading and overriding

overloading

→ Method overloading is when a class has multiple method with the same name, but different type of number of parameter. The compiler decide which method to run based on the method signature (parameter), so this happen at compile time.

Example

```
Class Print {
    void Show (string text) {
      system.out.print(text);
    }

    void Show(int number) {
       system.out.println(number);
     }

    void Show (string text, int number) {
        system.out.print(text + "" +number);
     }
  }
```

overriding

→ Method overriding is when a child class define a method that is already present in it's parent class, using the same name and parameter. This method in the child class replace and override the behaviour of the parent method This decision happen during runtime.

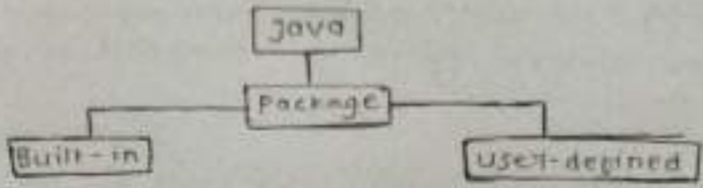
Example

```
class Animal {
   void speak () {
     system.out.print("Animal speak");
   }
 }

 class Dog extends Animal {
   @override
   void speak() {
      system.out.println("Dog barks");
    }
 }
```

3.5 Package

→ A package in Java is a collection of related classes, interface, and sub-package. It help organize code, avoiding name conflict, and controlling access.



- These are provided in java
- Ex:- java.util, java.io

- These are created by programmer to organize their own classes
- Ex:- A package named myproject.

- Naming and creating Package

- Usally written in lowercase
- if needed, follow the reserve domain naming convention
  Ex:- com.gaurav.project

Creating a Package:

- Use the package keyboard at the top of your Java file.

```
Package mypackage;
public class Hello{
    public void display(){
        system.out.print("Hello from package");
    }
}
```

- Accessing a package

- To use class from another package, we have to import them in our current program.

- Using a package

- If you created a package named mypackage with class Hello, and now you want use it.

```
import my.package.Hello;
Public class Test{
    public static void main(String[] args){
        Hello obj = new Hello();
        obj.display();
    }
}
```

UNIT-IV Errors & Exception Handling Multithreadings:

Errors

→ When writing a computer program. we something make mistake. This mistake are called errors. Errors stop the program from running properly just like in match. if you make mistake. your answer will be wrong.

Type of Errors

1. Compile - Time Errors

→ These are the mistake that happen when we try to run (compile) the program. The computer cheaks the code first before running it. if it find probium like a spelling mistake in the code or missing symbol.

- Example : Forgetting a put a semicolon(;) at the end of line.
- The program won't me even start until you fix these error.

2. Run - Time Errors

→ These error happen after the program starts running. The code look fine, so it's starts. but something goes wrong during the process.

- Example : Trying to divide a number by zero (like 10 ÷ 0)
- The progrim suddenly stop working or crashes.

Exception

→ A exception is an unwanted or unexpected event that disrupts the normal flow of program. Java provide mechanisms to handel exception using.

1. Try and catch statement

→ The try block contain code that might throw an exception. While the catch block handel the exception.

~~Syntax~~

```
try {
    //code that may throw an exception
} catch (Exception Type e) {
    //Exception handling code
}
```

2. Nested try statement

→ A try block inside another try block is called a nested try-catch.

• useful when a block of code inside inside a try can throw diffrent exception.

Syntax:

```
try {
  //outer try block
  try {
    //inner try block
  } catch (Exception Type1 e) {
    //inner catch block
  }
} catch (Exception Type2 e) {
    //outer catch block
}
```

Throws keyboard

→ The throw keyboard is used to declare an exception that a method migth throw, forcing the caller to handel it.

Syntax:

```
returnType methodName() throws Exception Type {
        // method code
   }
```

- Used for cheaked Exceptions
- The caller must handel the exception using try-catch or declare it again using throws.

Finally statement

→ The finally block execute wheother an exception occure not found. It is used for cleanup (eg- closing file, database connection)

Syntax:

```
try {
  // Risky code
} catch (Exception e) {
  // Exception handling
} finally {
  // Always execute
}
```

- try-catch → Handel exception gracefully
- Nested try → Handel exception at diffrent level
- Throws → Delegate exception handling to carrer
- Finally → Ensure critical code run regardless of Exception.

Built-in Exception

→ Built-in Exception are the Exception that are available java libraries. These exception are suitable to explain certain error suitable.

Example of Built-in Exception

1. Arithmatic exception
2. Array index out Bound
3. ClassNotfoundException
4. File Not foundException
5. IoxException
6. Interrupted Exception.

4.5 Multithreaded Programming

→ Multithreading means running two or more parts of a program at the same time.

Thread

→ A thread is a small part of program that runs on it's own, like mini-program inside your main program.

1. By Extending the Thread class

→ You create a new class that extends (inherit form) the built-in thread class and override it's run() method. Then you start using .start.

```
class My Thread extend Thread {
        public void run() {
            system.out.println(" Thread is running using thread
                                                      class");
        }
    }
    public class Main {
        public static void main(string[] args) {
            my thread t1 = new My thread();
            t1.start(); //start the thread
        }
    }
```

2. By implementing the Runnable interface

→ You create a class that implements the Runnable interface and write the code in the run() method. Then you pass it to a Thread object start it.

Ex:-

```
Class MyRunnable implements Runnable {
    public void run() {
        system.out.println("Thread running using Runnable
                                        interface");
    }
}
public class Main {
    public static void main(string[] args) {
        my Runnable obj = new MyRunnable
        Thread t1 = new Thread(obj);
        t1.start(); //start the thread
    }
}
```

4.6 Life cycle of Thread

1. New state

→ when a thread object is created using the Thread class (or via Runnable). It is the New state,

~~2~~ R   Thread t1 = new Thread(); // New state

2. Runnable State

→ When you call start() on the thread, it enter the Runnable state.

- The thread is ready to run, and it's waiting for the CPU to schedule it.
- It does'nt run imeddiatly, only when CPU allow.

3. Running state

→ When the CPU assign time to the thread, it m
from Runnable to Running.

→ The thread's run() method is actively executi

4. Blocked state

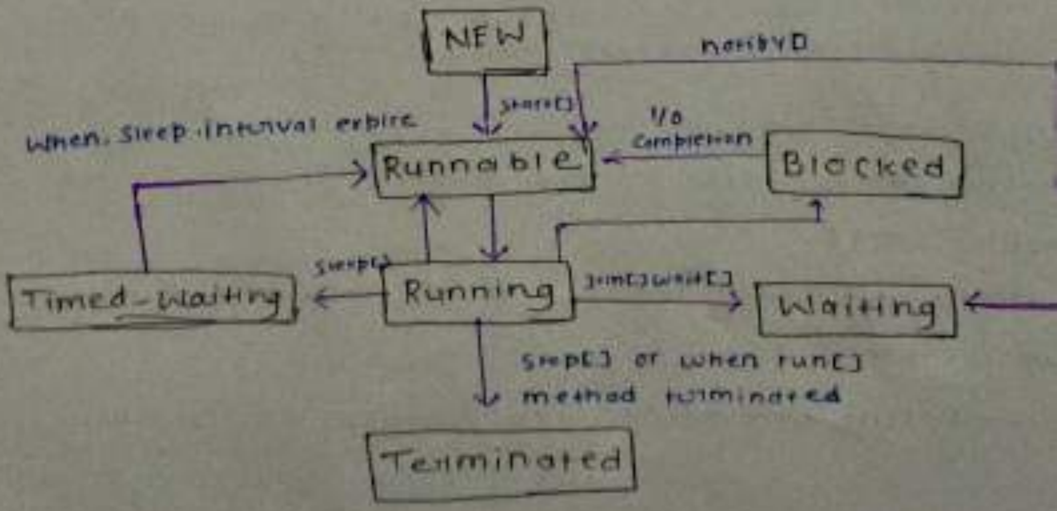
→ A thread goes to Blocked (or waiting)
- It is waiting for a resowrce (file, memory)
- It is seleeping using sleep() method.
- It is waiting for another thread to finis

Thread.sleep(1000); // Thread now is Blocked/Wa
state

5. Dead state

→ A thread reach the Dead state when
- The run() method complete.

* once dead, a thread cannot be restart.



Thread Method in Java

1. wait()
   - Used for : making a thread wait (pause) until another thread notify it.
   - Belong to : object class (not Thread class).
   - Must be called inside synchronized block.

2. sleep()
   - used for : Pausing the current thread for a specific time
   - Does not release the lock if the thread is holding one.
   - Come from Thread class and is static.

3. notify()
   - used for : Walking up one waiting thread that is waiting on same object.
   - Also must be used inside synchronized block.

4. resume() (Despirate)
   - used for : Resuming a thread that was suspands using suspands.

5. suspend() (Deprecated)
   - Temporarily pausing thread
   - Not recommand modern Java.

6. stop() (Deprecated)
   - Forcibly stopping a Thread
   - used interrupt mechanisms instead