CHAPTER - 1

Object Oriented Programming Concepts

What is OOP?

OOP stands for object-Oriented Programming. It is a programming paradigm based on concept of "objects", which can contain data and code: data in the form of fields, and code, in the form in procedures. A feature of objects is that an objects own procedures can access and often modify the data fields of itself.

What is POP?

POP stands for Progcedural Oriented Programming (POP). A program in a procedural language is a list of instruction where each statement tells the computer to do something. It focuses on procedure (function) & algorithm is needed to perform the derived computation.

Differences between OOP & POP?

OOP	POP
A programming paradigm based on the concepts of objects, which contains data in the form of fields known as attributes, and code in the form of procedures known as methods.	A programming paradigm that is based upon the concept of the procedure calls.
Stands for object oriented programming.	Stands for procedural oriented programming
Emphasis on Objects.	Emphasis on functions.
Divides the program into multiple objects.	Divides the program into multiple functions.
modification is easier as objects are independent.	modifications are difficult as they can effect the entire program.
Object communicate with each other by passing messages.	Functions communicate with each other by passing parameters
Each object controls it own data	Functions shared global variables.

OOP	POP
It is possible to hide data	There is no data hiding mechanism.
Has access specifiers	Do not have access specifiers
Supported by c++, Java and Python.	Supported by C, Pascal, FORTRAN and COBAL.

What are the features of oop in java?

Features of OOPs :-

1. Classes
2. objects
3. Data Abstraction
4. Encapsulation
5. Inheritance
6. Polymorphism.

What is class?

Class represents a real world entity which acts as a blueprint for all the objects. We can create as many objects as we need using class.

Example :-

We create a class for "student" entity as below.

Student.java.

```
class student {
String id;
int age;
String course;
Void enroll () {
System. out. println ("student enrolled ");
}
}
```

Above 3 fields id, age and course are classes and also it contains behavior or a method called "enroll".

What is an object?

A object oriented programming system (oops) is designed based on the concept of "objects". It contains both variables and methods (used for defining the behaviours)

We can create any number of objects using this class and all those objects will get the same fields and behavior.

Student s1 = new Student();

Now we have created 3 objects s1, s2, s3 for the same class "student".

We can create as many objects as required in the same way.

We can set the value for each field of an object as below.

s1.id = 123;

s2.age = 18;

s3.course = "computers";

What is Abstraction?

Abstraction is a process where you show only "relevant" data and "hide" unnecessary details of an object from the user.

We can achieve "abstraction" in java using 2 ways. 1*. Abstract class

2* Interface.

1. Abstract class:-

* Abstract class in java can be created using "abstract" keyword.

* If we make any class as abstract then it can't be instantiated which means we are not able to create the object of abstract class.

* Inside Abstract class, we can declare abstract methods as well as concrete methods.
* So using abstract class, we can achieve 0 to 100% abstraction.

2. Interface :-

* Interface is used to achieve pure or complete abstraction.
* we will have all the methods declared inside interface as abstract only.
* so, we call interface as 100% abstraction.

What is Encapsulation?

* Encapsulation is the process of binding object state (fields) and behaviors (methods) together in a single entity called "class".
* Since it wraps both fields and methods in a class, it will be secured from the outside access.
* We can restrict the access to the members of a class using access modifiers such as private, protected and public keywords.
* When we create a class in Java, it means we are doing encapsulation.

* Encapsulation helps us to achieve the re-usability of code without compromising the security.

What is inheritance?

A * one class inherits or acquires the properties of another class.

* Inheritance provides the idea of reusability of code and each subclass defines only those features that are unique to it, rest of the features can inherited from the parent class.

1. Inheritance is a process of defining a new class based on an existing class by extending it's common data members and methods.

2. It allows us to reuse of code, it improves reusability in your java application.

3. The parent class is called the base class or super class. The child class that extends the base class is called the derived class or subclass or Child classes.

To inherit a class we use extends keyword.

Here class A is child class and class B is parent class.

```
class A extends B
{
}
```

What are the various types in Inheritance?

A 1. Single Inheritance:-

Refers to a child and parent class relationship where a class extends the another class.

2. Multilevel Inheritance:- A child and parent class relationship where a class extends the child class. For example class A extends class B and class B extends class C.

3. Hierachical Inheritance:- Where more than one classes extends the same class. For example; class B extends class A and class C extends class A.

What is Polymorphism?

* It is the concept where an object behaves differently in different situations.
* Since the object takes multiple forms, it is called polymorphism.
* In java, we can achieve it using method overloading and method overriding.

* There are 2 types of polymorphism available in Java.

* Method Overloading

In this case, which method to call will be decided at the compile time itself based on number or type of the parameters. Static/compile Time polymorphism is an example for method overloading.

* Method Overriding:-

In this case, which method to call will be decided at the run time based on what object is actually pointed by the reference variable.

Compare the class and structure?

STRUCTURE	CLASS
If access specifier is (Basic) not declared, by default all member are "public".	If access specifier is (Basic) not declared, by default all members are "private".

Struct (Declaration) structure_name {
type struct_element 1;
type struct_element 2;
:
:
};
(Declaration)
	class class_name {
	data member;
	member function;
	};

Basic for comparision	STRUCTURE	CLASS
Instance	Instance of 'Struiture' is called "structure variable".	Instance of a "class" is called "object".
Polymorphism and Inheritance.	Not supported	Supports polymorphism and a class can also be inherited.
Nature	Value type	Reference type.
Memory is allocated on	Stack	Heap
Null values.	Not possible	Can have null values.
Requires Constructor and destructor	NO	YES.

CHAPTER - 2

JAVA Basics, Usage Of Classes And Objects

History of JAVA:-

* The history of java starts from green Team. Java team members (also known as green Team). Initiated a revolutionary task to develop a language for digital devices such as set-top boxes, televisions etc.

* James Gosling, Mike Sheridan and patrick - Naughton initiated the java language project in june 1991. The small team of sun engineeers called green team.

* Firstly, it was called "Green talk" by James Gosling and file extension was .gt

* After that, it was called Oak and was developed as part of the green project.

=> Why oak?

Name is inspired by an oak tree that stood outside of gosling's office. oak tree is a symbol of strength and chosen as a national tree of many countries like U.S.A,

france, Germany etc.

* But in 1995, team had decide to change the name of language, because Oak name was already a trademark by Oak technologies.

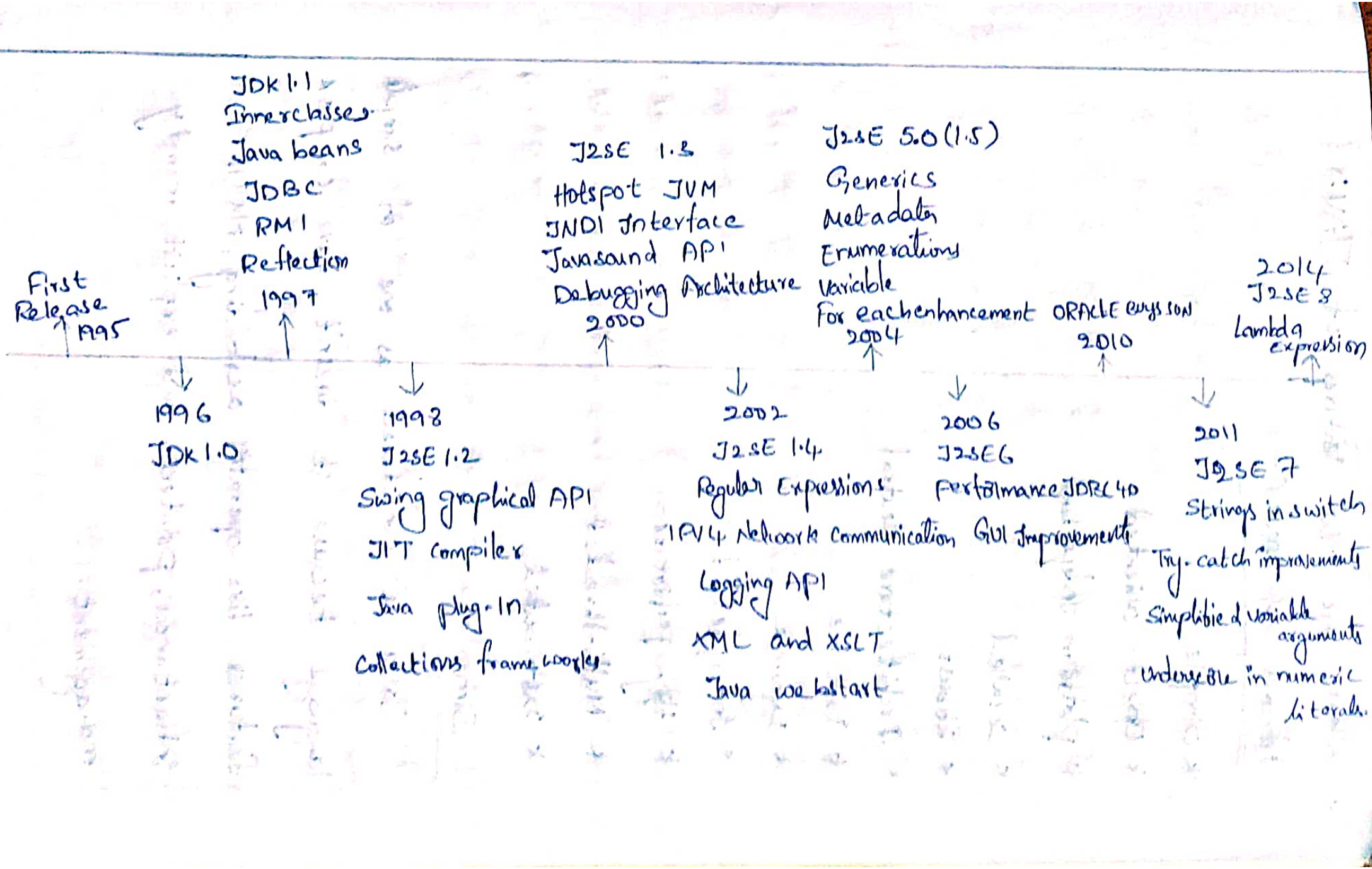
* New name was JAVA:-

The suggested words were "dynamic", "revolutionary", "silk", "jolt", "DNA" etc. According to james Gosling "java and silk" were top choices. Since java was so unique, most of the team members preferred java.

* Java:- Java is an Island of Indonesia where first coffee was produced (called java coffa).

* Java coffee consumed in large quantities by language creators.

* Java is just a name not an acronym.


First Release 1995
1996
JDK 1.0
JDK 1.1
Innerclasses
Java beans
JDBC
RMI
Reflection
1997
1998
J2SE 1.2
Swing graphical API
JIT compiler
Java plug-in
Collections framework
J2SE 1.3
Hotspot JVM
JNDI Interface
Javasound API
Debugging Architecture
2000
2002
J2SE 1.4
Regular Expressions
IPV4 Network communication
Logging API
XML and XSLT
Java web start
J2SE 5.0 (1.5)
Generics
Meta-data
Enumerations
Variable
For each enhancement
2004
2006
J2SE 6
Performance JDBC 4.0
GUI Improvements
ORACLE buys SUN
2010
2011
JSE 7
Strings in switch
Try-catch improvements
Simplified variable arguments
Underscore in numeric literals
2014
J2SE 8
Lambda expression

* What are the features of JAVA?

Features of JAVA :-

* Simple
* Object - oriented
* platform Independent
* secured
* Robust
* Architecture Neutral
* Portable
* Dynamic
* Interpreted
* High performance
* Multi-threaded
* Distributed.

1. Simple :-

Java is considered as one of simple language because it doesn't have complex features like operator overloading, Multiple inheritance, pointers and explicit memory management.

2. Object Oriented:-

Basic concepts of object oriented programming are
1. object
2. class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation.

3. Platform Independent:-

A platform is the hardware or software environment in which a program runs.

There are two types of platforms software-based and hardware-based. Java provides software-based platform.

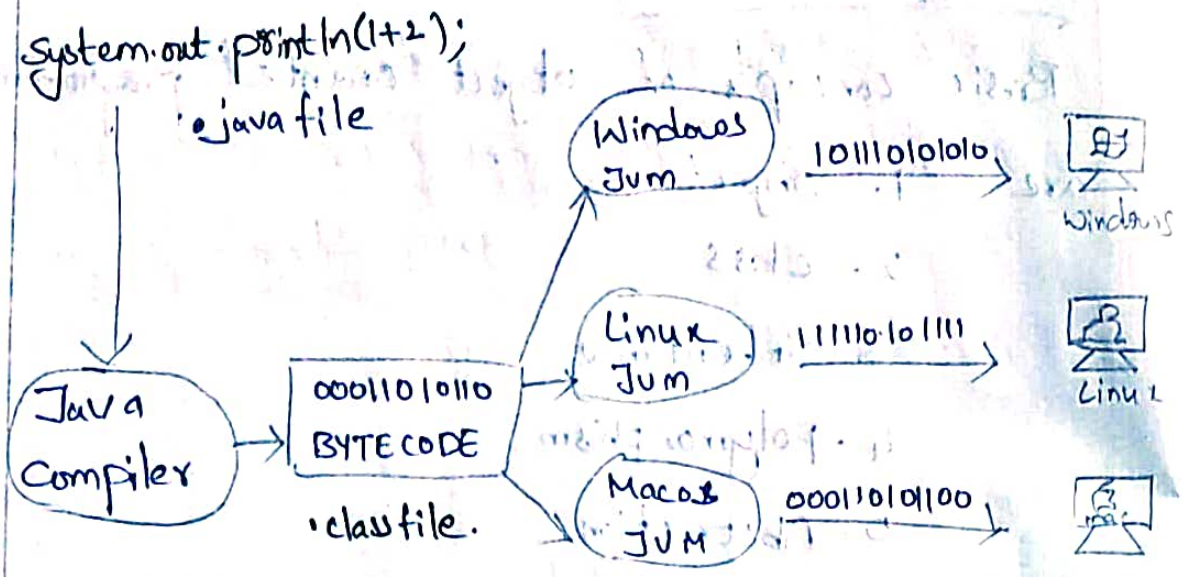
Java code can be run on multiple platforms

Ex:- Windows, Linux, sunsolaris, Mac/os etc.

Java code is compiled by the compiler and converted in to bytecode.

This bytecode is a platform - independent code because it can be run on multiple platforms.

i.e write once and run anywhere (WORA)



4 * Robust:- Two main problems that cause program failures are memory management mistakes and mishandled runtime errors. Java handles both of them efficiently.

* Memory management mistakes can be overcome by garbage collection. Garbage collection is automatic deallocation of objects which are no longer needed. Mishandled runtime errors are resolved by exception handling procedures.

5. Portable :-

When we compile the java program, the compiler of java generates .class file that contains the Byte codes of your java program.

* The generated bytecodes are secure and can run on any machine (portable) which has JVM.

6. Architecture - Neutral :-

In java there is no implementation dependent feature eg: size of primitive type is fixed.

In C programming, float data type occupies 4 bytes of memory for 32-bit architecture and 8 bytes memory for 64-bit architecture. But in java, it occupies 4 byte memory for both 32 and 64 bit architectures.

7. Multi threaded :-

Java supports multithreading. It enables a program to perform several tasks simultaneously.

8. Interpreted and High performance :-

Java interpreter converts the bytecode into processor readable binary code. For very high performance it uses a just-in-time compiler.

9. Distributed :-

RMI (Remote method Invocation) and EJB (Enterprise Java beans) are used for Creating distributed applications in java.

* In simple words :- The java programs can be distributed on more than one systems that are

Connected to each other using internet Connection objects on one JVM (Java Virtual Machine) can execute procedures on a remote JVM.

Write about Bytecode of JVM and Java?

A) 1 Byte Code of JAVA:-

Java byte code is the instruction set of the java Virtual Machine. Each bytecode is composed by one, or in some cases two bytes that represent the instruction (opcode), along with zero or more bytes for passing parameters.

Byte code is the compiled format for Java programs. Once a java program has been converted to bytecode, it can been converted to bytecode, it can be transferred across a network and executed by java Virtual machine (JVM)

Byte code files generally have a .class extension.

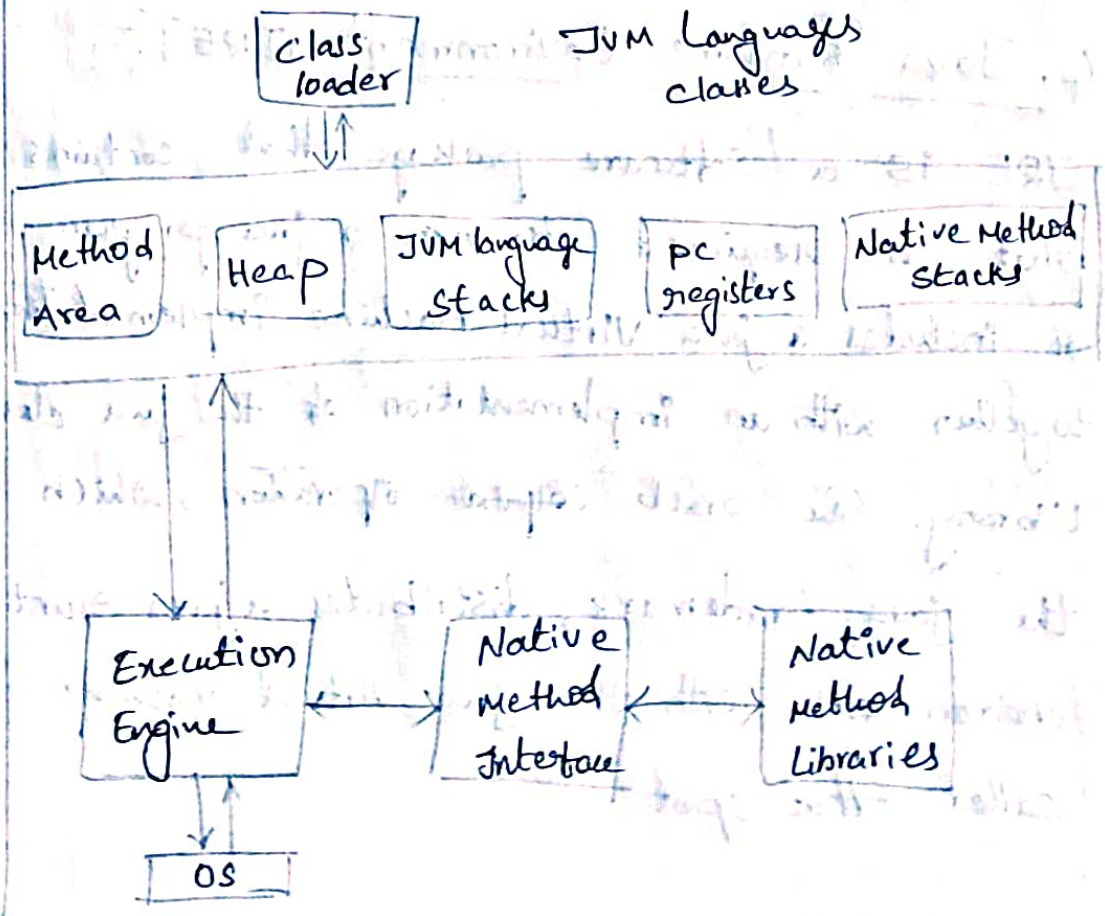
2 JVM:-

A java virtual Machine (JVM) is an abstract computing machine that enables a computer to run a java program.

There are three notions of the JVM:-

* Specifications
* Implementation and
* Instance.

The specification is a document that formally describes what is required of a JVM implementation. Having a single specification ensures all implementations are interoperable. A JVM implementation is a computer program that meets the requirements of the JVM specification. An instance of a JVM is an implementation running in a process that executes a computer program compiled in to java bytecode.

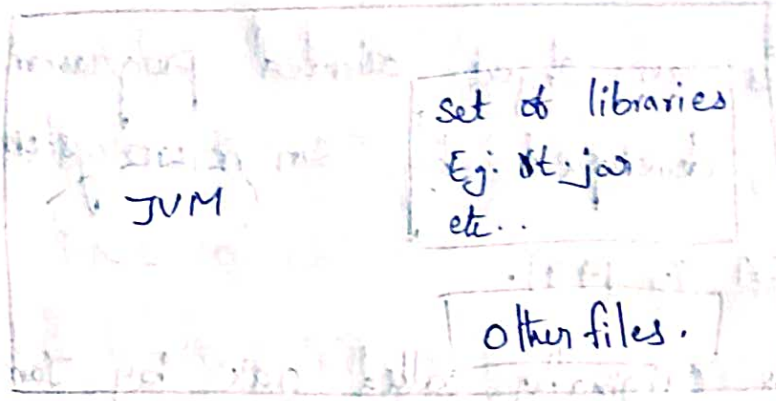


3. Virtual Machine (VM):-

It is a software based, imaginary computer. Originally developed by Sun Microsystems. Virtual machines based on specifications of a hypothetical computer or emulate the computer architecture and functions of a real world computer. The Virtual Machine runs programs in a byte code interpreter (Computing). This code is most often generated by java language compilers; the availability of JVMs on many types of hardware and software platforms enables java to function both as middleware and a platform in its own right. Hence the expression "Write once, run anywhere".

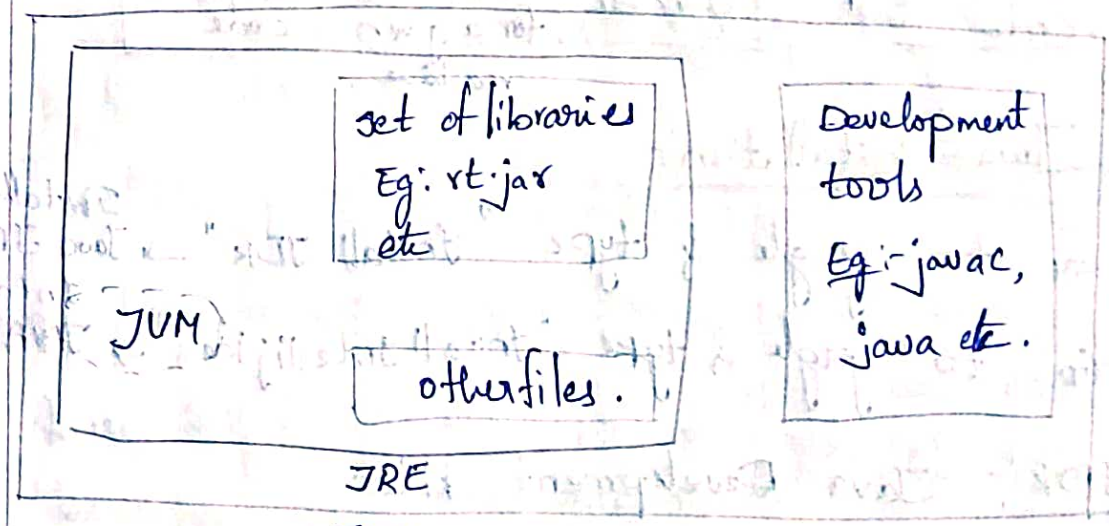
4. Java Runtime Environment [JRE]:-

JRE is a software package that contains what is required to run a java program. It includes a java Virtual Machine implementation together with an implementation of the java class Library. The oracle corporation, which owns the java trademark, distributes a java runtime environment with their java virtual machine called Hot spot.



5. Java Development kit [JDK] :-

JDK is a superset of a JRE and contains tools for java programmers. Eg:- A Javac compiler. Java Development kit is provided free of charge either by Oracle corporation directly, or by the open JDK open source project, which is governed by Oracle.



JAVA

Java is an object oriented programming language developed by sun microsystems of USA in 1991.

It was originally called oak by James Gosling one of the inventor of java.

Java = purely object oriented.

How java work?

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

Java Installation:-

Go to google & type "Install JDK" → Installs Java JDK

Go to google & type "Install Intellij Idea" → Installs JAVA IDE

JDK:- Java Development Kit.

Collection of tools used for developing and running java programs.

JRE:- JAVA Runtime Environment

Helps in executing programs developed in java.

Basic structure of a java program?

```
Package com.company;  → Groups classes!

public class main {  → Entry point into the application
    public static void main (String [] args) {
        system.out.println ("Helloworld");
    }
}
```

Naming Conventions:-

=> For classes, we ~~have~~ use pascal convention. First and subsequent characters from a word are capital letters (upper case).

Ex:-
Main, Myscam, Myemployee, Code With Harry.

=> For functions and variables, we use Camel case Conventions. Here first character is lowercase and the subsequent characters are uppercase like below:-

main, myScam, myMarks, codeWithHarry.

Variables and Datatypes.

Just like we have some rules that we follow to speak english (the grammer), we have some rules to follow the while writing a java program. The set of these rules is called syntax. → Vocabulary & Grammer of java.

Variables:-

A variable is a container that stores a value. This value can be changed during the execution of the program.

Ex:-

```
int number = 8;
```

- `int` → datatype
- `number` → variable name
- `8` → value it stores

Rules for declaring a variable name.

We can choose a name while declaring a java variable if the following are followed:-

1. Must not begin with a digit → `int 1array;` is invalid!
2. Name is case sensitive → harry and Harry are different.
3. Should not be "0" keyword → (like void)
4. White space not allowed. → `int cake with harry;` is invalid!
5. It can contains alphabets, $ characters, _ characters and digits if the other conditions are met.

Data types :-

Data types in java fall under the following categories :-

1. primitive Data types (Intrinsic)
2. Non-primitive Data types (Derived)

Primitive Data types

Java is ~~sati~~ Statically typed → Variables must be declared before use !

There are 8 primitive data types supported by java :-

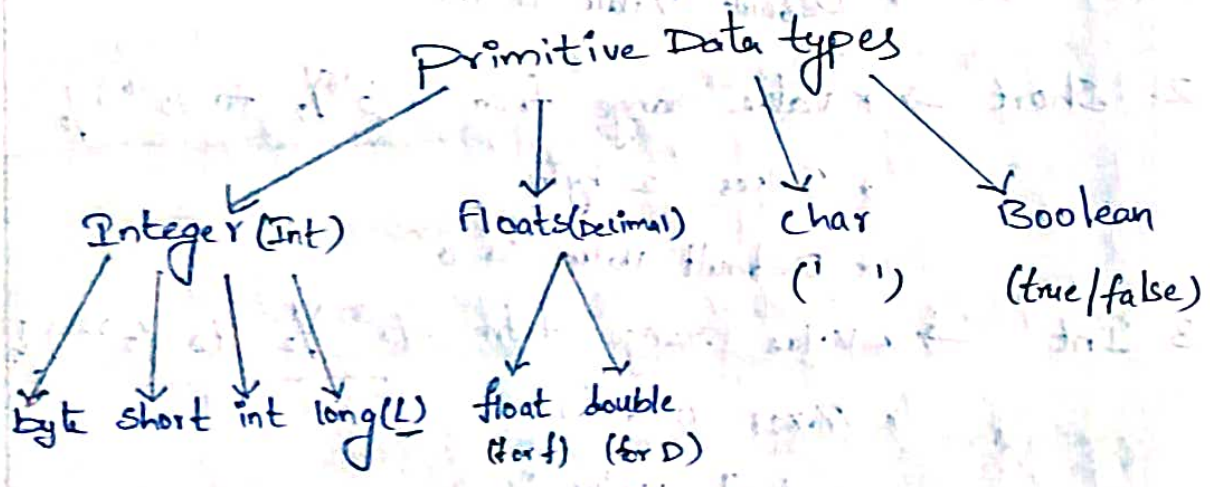
1. byte → * Value range from -128 to 127
   * Takes 1 byte
   * Default value is 0.
2. Short → * value range from $-(2^{16})/2$ to $(2^{16})/2 - 1$
   * Takes 2 bytes
   * Default value is 0.
3. Int → * value range from $-(2^{32})/2$ to $(2^{32})/2 - 1$
   * Takes 4 bytes
   * Default value is 0.
4. Float → * value range from (see docs)
   * Take 4 bytes
   * Default value is 0. of
5. Long → * value range from $-(2^{64})/2$ to $(2^{64})/2 - 1$
   * Takes 8 bytes
   * Default value is 0.

6. Double → * Value range from (See docs)
* Takes 8 byte
* Default value is 0.0d

7. Char → * Value range from 0 to 65535 ($2^{16}-1$)
* Takes 2 byte → because it supports unicode
* Default value is '\u0000'.

8. Boolean → * value can be true or false
* size depends on JVM
* Default value is false.

How to choose data types for our variables.



In order to choose the data type we first need to find the type of data we want to store. After that we need to analyze the min & max value we might use.

Literals

A constant value which can be assigned to the variable is called as a literal.

101 → Integer literal

10.1f → Float literal

10.1 → Double literal (default type for Decimal)

'A' → Character literal

true → Boolean literal

"Hari" → String literal

Key Words :-

Words which are reserved and used by the Java compiler. They cannot be used as an identifier.

⇓

Go to docs.oracle.com for a comprehensive list!

Reading Data from the keyboard.

In order to read data from the keyboard, Java has a scanner class.

Scanner class has a lot of methods to read the data from the keyboard.

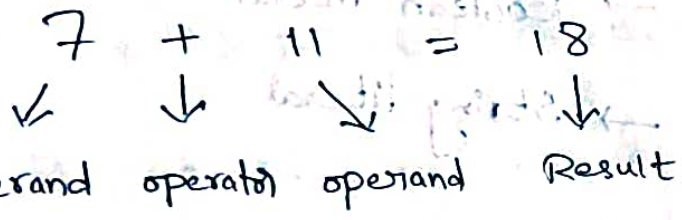
```
Scanner s = new Scanner(System.in);
```

↳ Read from the keyboard.

```
int a = S.nextInt();
```
↳ Method to read from the keyboard (Integer in this case)

Operators and Expressions

Operators are used to perform operations on variables and values.



Types of operators :-

* Arithmetic operators → +, -, *, /, %, ++, --
* Assignment operators → =, +=
* Comparision operators → ==, >=, <=
* Logical operators → &&, ||, !
* Bitwise operators → &, | (operates bitwise)

Arithmetic operators cannot work with booleans

% operator can work on floats & doubles.

Precedence of operators :

The operators are applied and evaluated based on precedence. For example (+, -) has

less precedence compared to (*, /).
Hence * & / are evaluated first.
In case we like to change this order, we use paranthesis.

Associativity:-

Associativity tells the direction of execution of operators. It can either be L to Right or Right to left.

* /	→	L to R
+ -	→	L to R
++ , =	→	R to L

Resulting data type after arithmetic operation.

following table summarizes the resulting data types after arithmetic operation on them.

R = b + s → int	b → byte
R = s + i → int	s → short
R = l + f → float	i → integer
R = i + f → float	l → long
R = c + i → int	f → float
R = c + s → int	d → double
R = l + d → double	c → character
R = f + d → double	R → Result

Increment and Decrement operators.

a++, ++a → Increment operator → Data type

a--, --a → Decrement operator → remains same

These will operate on all data types except booleans.

Strings

A string is a sequence of characters.

A string is instantiated as follows:-

```
String name;
name = new String ("Harry");
```

String is a class but can be used like a data type; [strings are immutable and cannot be changed]

```
String name = "Harry";
```

reference — object

Different ways to print in Java.

1. System.out.print() → No newline at the end!
2. System.out.println() → Prints a new line at the end.
3. System.out.printf()
4. System.out.format()

System.out.printf ("%.", ch)

↳ %d for int
%f for float
%c for char
%s for String

String Methods

String methods operate on java strings. They can be used to find length of the string. convert to lowercase, etc.

Some of the commonly used string methods are :—

String name = "01234 Harry";

1. Name.length() → Returns length of String Name. (5 in this case)
2. Name.tolowercase() → Returns a new string which has all the lowercase characters from the string name.
3. Name.touppercase() → Returns a new string which has all the uppercase characters from the string name.
4. Name.trim() → Return a new string after removing all the leading and trailing spaces from the original string.
5. Name.substring(int start) → Returns a substring from start to the end substring(3) returns "ry"

[Note that index starts from 0]

6. Name.substring(int start, int end) →
Returns a substring from start index to the end index. Start index is included and end is excluded.

7. Name.replace('r', 'P') →
(char, char)
Returns a new string after replacing r with P. Happy is returned in this case.

8. name.startswith("Ha") → returns true if name starts with string "Ha". true in this case!
(String)

9. name.endswith("ry") → returns true if name ends with string "ry". true in this case.
(String)

10. Name.charAt(2) → returns character at a given index position. r in this case!
(int)

11. Name.indexOf(s) → returns the index of the given string
(String)
Forex: name.index of("ar" returns 1 which is the first occurance of ar in string "Harry", -1 otherwise.

12. Name.indexOf("s", 3) → returns the index of the given string starting from the index 3 (int). -1 is returned in this case!

13 Name.lastIndexOf("r") →
returns the last index of the given string. 3 in this case!

14 Name.lastIndexOf("r",2) →
returns the last index of the given string before index 2.

15. name.equals("Harry") →
returns true if the given string is equal to "Harry" false otherwise [case sensitive]

16. Name.equalsIgnoreCase("harry") →
returns true if two strings are equal ignoring the case of characters.

Escape Sequence characters

Sequence of character after backslash "\"
= Escape (Sequence character)

Escape Sequence characters consist of more than one characters but represents one character when used within the strings.

Ex:- \n, \t, \', \\ etc.
↓ newline, ↓ Tab, ↓ Singlequote, ↓ backslash.

Conditional in Java

All these are decisions which depends on a certain condition being met.

In java, we can execute instructions on a condition being met.

Decision making instructions in java

⇒ If-else statements

→ Switch statements

If-else statement

The syntax of an if-else statement in c looks like that of c++ and java script.

Java has a similar syntax too. It looks like:-

```
if (condition-to-be-checked)
{
  Statements-if-condition-true;
}
else
{
 Statements-if-condition-false;
}
```

Code example :-

```
int a = 29;
if (a > 18)
{
 system.out.println("You can drive");
}
```

Note that the else block is optional.

Else-if clause :-

Instead of using multiple if statements. we can also use else if along with if thus forming an if-else ; if-else ladder.

Using such kind of logic reduces indents. last else is executed only if all the conditional fail.

Syntax :-

```
if (conditional ) {
        // statements;
}
elseif {
    // statements;
}
    else {
        // statements;
        }
```

Switch Case control instruction:-

Switch case is used when we have to make a choice between number of alternatives for a given variable.

Syntax :-

```
Switch (var)
{
 case c1:
        // code;
        break;
  Case c2:
        //code;
        break;
   case c3:
        //code;
        break;
    Default :
        //code;
}
```

Var can be an integer, character or string in java.

A switch can occur within another but in practice this is rarely done.

Relational operators in java :-

Relational operators are used to evaluate conditions (true or false) inside the if statements. Some examples of relational operators are:-

`==` , `>=` , `<` , `>` , `<=` , `!=`

↓ equals ; ↓ greater than or equal to ; ↓ less than ; ↓ greater than ; ↓ Lesser than ; ↓ Not equal to

Note:- "=" is used for assignment where as "==" is used for equality check.

The condition can be either true or false.

Logical operator:-

&&, ||, and ! are most commonly used logical operators in java.

There are read as :-

`&&` → AND
`||` → OR
`!` → NOT

} → used to provide logic to our JAVA programs.

AND operator

Evaluates to true if both the conditions are true.

Y && Y = Y
Y && N = N
N && N = N
N && Y = N

Y = true
N = false

OR operator

Evaluates to true when at least level one of the conditions is true.

Y || Y = Y

Y || N = Y

N || Y = Y

N || N = N

Y = true

N = false.

NOT operator

Negates the given logic (true becomes false and false becomes true)

!Y = N

!N = Y

Y = true

N = false.

Loop Control Instruction

Sometimes we want our programs to executes a few set of instructions over and over again. For example - print 1 to 1000, print multiplication table of 7 etc...

Loops make it easy for us to tell the computer that a given set of instructions need to be executed repeatedly.

Types of loops :-

primarily there are three types of loops in java:-

1. While loop
2. Do- while loop
3. for loop.

we will look in to these one by one.

While loops

```
while (boolean condition)
{
    // Statements;
}
```

=> This keeps executing as long as the condition is true.

If the condition never becomes false, the while loop keeps getting executed. such a loop is known as an infinite loop.

Do - while Loop :-

This loop is similar to a while loop except the fact that it is guaranteed to execute at least once.

```
do {
    // code
} while (condition);
```

→ Note this semicolon.

while → Checks the condition & executes the code.

do-while → Executes the code & then checks the condition

For loop:-

The syntax of a for loop looks like this:-

```
for (initialize ; check-body-expression; update)
{
   // code;
}
```

A for loop is usually used to execute a piece of code for specific number of times.

Decrementing for loop:-

```
for ( i=7 ; i=0; i--)
{
  System.out.println (i);
}
```

This for loop keeps running until "i" becomes "0".

Break Statement:-

The break statement is used to exit the loop irrespective of whether the condition is true or false.

Whenever a "break" is encountered inside the loop, the contrd is sent outside the loop.

Continue Statement

The continue statement is used to immideately move to the next iteration of the loop.
The control is taken to the next iteration.
Thus skipping everything below "continue" inside the loop for that iteration.

In a Nut Shell ----

1. Break Statement completely exits the loop.
2. Continue Statement skips the particular iteration of the loop.

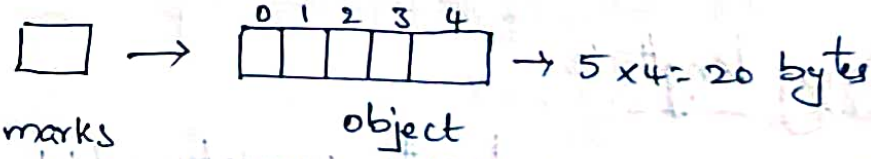
Arrays

Array is a collection of similar types of data

Use case: storing marks of 5 students.

int [] marks = new int [5] ⇒ [datatype Arrname;]

reference ↓ object.



Accessing Array Elements :—

Array elements can be Accessed as follows :-

marks [0] = 100

marks [1] = 70 ⇒ Note that index starts from 0

⋮

marks [4] = 98

so in a nut shell, this is how array works :-

1. int [] marks; → Declaration !
   marks = new int [5]; → Memory allocation !

2. int [] marks = new int [5]; → Declaration + memory Allocation

3. int [] marks = { 100, 70, 80, 71, 98 }
   ↓ Declaration + initialize !

Array indices start from 0 and goes till (n-1) where n is the size of the array.

Array length :-

Arrays have a length property which gives the length of the array.

marks.length → gives 5 if marks is a reference to array with 5 elements.

Displaying an array :-

An array can be displayed using a for loop:

```
for (int i=0 ; i< marks.length; i++)
{
  sout (marks[i]);  => Array Traversal.
}
```

For-each loop in java

Array elements can also be traversed as follows:

```
for (int element : Arr)
{
  sout (element);  => prints all the elements
}
```

Multidimensional Arrays :-

Multi dimensional Arrays are arrays of arrays each element of a M-D array is an array itself makes in the previous example was a 1-D array.

Multidimensional 2-D Array

A 2-D array can be created as follows :-

```
int [][] flats = new int [2][3]
```

↳ A 2-D array of 2 Rows + 3 columns.

we can add elements to this array as follows:-

```
flats[0][0] = 100
flats[0][1] = 101
flats[0][2] = 102
 :
&so on!
```

This 2-D array can be visualised as follows:-

		[0] Col 1	[1] Col 2	[2] Col 3
[0]	Row 1	(0,0)	(0,1)	(0,2)
[1]	Row 2	(1,0)	(1,1)	(1,2)

Similarly a 3-D array can be created as follows:-

```
String[][][] arr = new string[2][3][4];
```

Methods In Java

Some times our program grows in size and we want to seperate the logic of main method to other methods.

For instance — If we are calculating average of a number pair 5 times, we can use methods to avoid repeating the logic.

↳ DRY → Don't Repeat Yourself

Syntax of a Method

A method is a function written inside a class. Since java is an object oriented language, we need to write the method inside some class.

```
data type name () {
    // method body
}
```

following method returns sum of two numbers.

```
int mysum (int a, int b) {
    int c = a+b;
    return c;
}
```

↑ return type

→ Return value.

Calling a Method

A method can be called by creating an object of the class in which the method exists followed by the method call :-

```
Calc obj = new Calc ();   → object Creation.
obj.mysum(a,b);           → Method call upon
                            an object.
```

The values from the method call (a and b) are copied to the a and b of the function mySum. Thus even if we modify the values a and b inside the method, the values in the main method will not change.

Void return type:-

when we don't want our method to return anything, we use void as the return type.

Static keyword:-

Static keyword is used to associate a method of a given class with the class rather than the object. Static method in a class is shared by all the objects.

Process of method invocation in Java:-

Consider the method Sum:

```
int sum (int a, int b)
{
  return a+b;
}
```

The method is called like this:

```
Calc obj = new Calc();
c = obj.Sum(2,3)
```

The values 2 and 3 are copied to a and b and then a+b = 2+3 = 5 is returned in c which is an integer.

Note:- In case of Arrays, the reference is passed. Same is the case for object passing to methods.

Method Overloading

Two or more methods can have same name but different parameters. Such methods are called overloaded methods.

```
void foo()
void foo(int a)          => Overloaded function foo
void foo(int a, int b)
```

Method overloading cannot be performed by changing the return type of methods.

Variable Arguments ~~(varag~~ (Varargs)

A function with vararg can be created in Java using the following syntax:-

```
Public Static Void foo (int ... arr)
{
  // arr is available here as int [] arr
}
```

foo can be called with zero or more arguments like this:

foo(7). foo(7,8,9) foo(1,2,7,8,9)

we can also create a function bar like this.

```
Public static void bar (int a, int arr)
{                          ↳ Atleast one integer
  //code                      is required now.
}
```

bar can be called as: bar(1), bar(1,2),

bar(1,7,9,11) etc---

Recursion

A function in java can call itself such calling of function by itself is called recursion.

Ex:- Factorial of a number

$$factorial(n) = n * factorial(n-1) \quad \forall\, n \geq 1$$

Access Modifiers & Constructors

Access Modifiers

Specifier where a property / method is accessible. There are four types of access modifiers in java:
1. private
2. Default
3. protected
4. public.

Getters and Setters

Getter → Returns the value [accessors]
Setter → sets / updates the value [mutators]

Example:-

```
public class Employee {
    private int id;
    private string name;
    public string getName() {
        return name;
    }
    public void setName() {
        this.name = "your-name";
    }
    public void setname(string n) {
        this.name = n;
    }
}
```

Constructors in Java :-

A member function used to initialize an object while creating it.

```
Employee harry = new employee();
harray setname ("Harry bhai");
```

In order to write our own constructor, we define a method with name same as class name.

```
public Employee()
{
name = ("YourName");
}
```

Constructor Overloading in Java :-

Constructor can be overloaded just like other methods in java. We can overload the Employee constructor like below:

```
public Employee (string n)
{
  name = n;
}
```

Note :- ① Constructors can take parameters without being overloaded.

② There can be more than two overloaded constructors.

Inheritance

Inheritance is used to borrow properties & methods from an existing class.

Phone → Smart phone

Superclass → Subclass Subclass extends Superclass

Declaring Inheritance in Java :-

Inheritance in java is declared using extends keyword.

SuperClass
↓
Subclass

⇒ subclass extends the superclass.

More Examples :-

Vehicle → car

Animal → Dog

Animal → cat

Vehicle → Truck.

when a class inherits from a superclass, it inherits parts of superclass methods and fields.
Java doesn't support multiple inheritance
i.e. two classes cannot be superclasses for a subclass.

Code Example :-

Inheritance in java is declared using extends keyword.

```
public class Dog extends Animal
{
//code
}
```

Inheriting Dog from Animal class !!

Constructors in Inheritance :-

When a Derived class is extended from the Base class, the constructor of the Base class is executed first followed by the constructor of the derived class.
for the following inheritance hierarchy the constructors are executed in the order.

① → ② → ③

C_1 → parent
↓
C_2 → child
↓
C_3 → Grandchild

↓ Constructor executes in top to bottom order.

Constructors during constructor overloading:—

when there are multiple constructors in the parent class, the constructor without any parameters is called from the child class. If we want to call the constructor with parameters from the parent class, we can use super keyword.

Super(a,b); → calls the constructor from the parent class which takes 2 variables.

this keyword:—

This is a way for us to reference an object of the class which is being created/referenced.

this.area = 2 → this is a reference to current object.

Super keyword:—

A reference variable used to refer immediate parent class object.

→ Can be used to refer immediate parent class object.

→ can be used to invoke parent class methods

→ can be used to invoke parent class constructors.

Method overriding:-

In this child class implements the same method present in the parent class again, it is known as method overriding.

Abstract classes & Interfaces

What does Abstract (class) mean?

Abstract in english means → existing in thought or as an idea without concrete existance.

Abstract Method.

A method that is declared without an implementation.

```
abstract void moveTo(double x, double y)
```

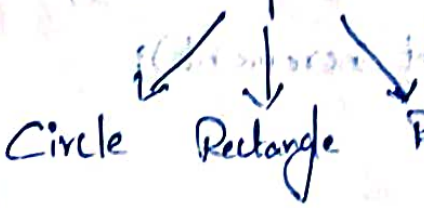
Abstract class:-

If a class includes abstract methods, then the class itself must be declared abstract, as in:

```
Public abstract class phoneModel {
    abstract void switchoff();
    // more code
}
```

When an abstract class is subclassed, the subclass usually provides implementations for all of the methods in parent class. If it doesn't, it must be declared abstract.

An Example:-



Note:-
It is possible to create reference of an abstract class.
It is not possible to create an object of an abstract class.
We can also assign reference of an abstract class to the object of a concrete subclass.

Interfaces in Java

Interface in english is point where two systems meet with empty interact.

In java interface is a group of related methods with empty bodies.

An Example:-

```
interface Bicycle {
    void applyBrake (int decrement);
    void speedUp (int increment);
}

class AvonCycle implements Bicycle {
    int speed = 7;
    void applyBrake (int decrement) {
        speed = speed - decrement;
    }
    void speed up (int increment) {
        speed = speed + increment;
    }
```

Abstract class V/S Interfaces

We can't extend multiple abstract classes but we can implement multiple interfaces at a time.

Interfaces are meant for dynamic method dispatch and run time polymorphism.

Is multiple inheritance allowed in Java?

Multiple inheritance face problems when there exist methods with same signature in both the super classes.

Due to such problems, Java does not support multiple inheritance directly but the similar concept can be acheived using interfaces.

A class can implement multiple interfaces and extend a class at the same time.

Note :-

① Interfaces in java is a bit like the class but with a significant differences.

② An interfaces can only have method signatures, constant fields and default methods.

③ The class implementing an interface needs to only declare the methods (not fields)

④ You can create a reference of interfaces but not the object.

⑤ interface methods are public by default.

Default Methods

An interface can have static and default methods. Default methods enable us to add new functionality to existing interface.

This feature was introduced in java 8 to ensure backward compatibility while updating an interface.

Classes implementing the interface need not implement the default methods.

Interfaces can also include private methods for default methods to use.

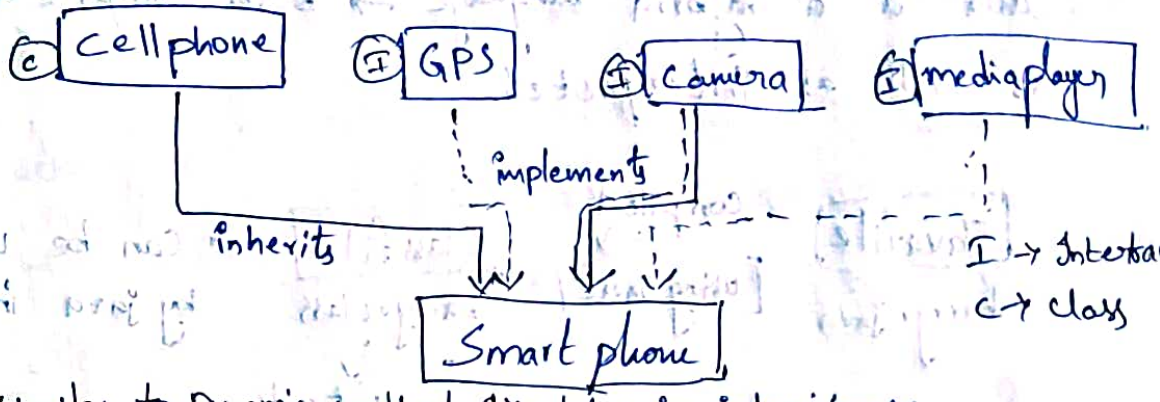
Inheritance in Interfaces

Interfaces can extend another interfaces.

```
public interface Interface1 {
    void meth1();
}
public interface interface2 extends interface1 {
    void meth2();
}
```

Remember that interface ~~extends~~ ~~int~~ cannot implement another interface, only classes can do that!

Polymorphism using Interfaces:-



Similar to Dynamic method dispatches in inheritance

GPS g = new Smartphone();

Can only use GPS methods.

Smartphones = new Smartphone();

Can only use Smartphone method

Implementing an interface forces method implementation.

Packages

Interpreter vs compiler

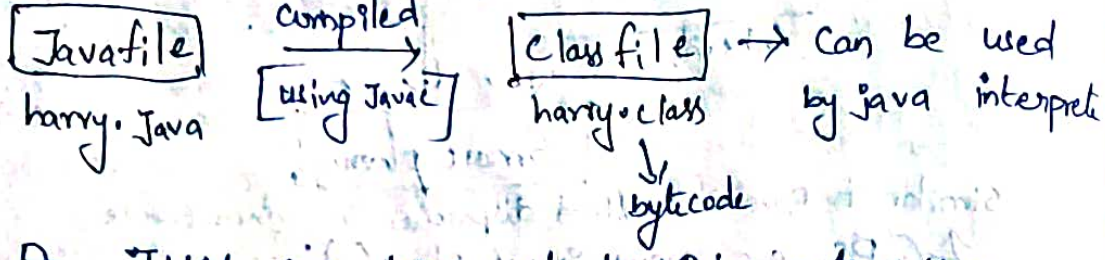
Interpreter translates one statement at a time in to machine code.

Compiler scans the entire program and translates whole of it into machine code.

Interpreter	compiler
* one statement at a time	* Entire program at a time
* Intepreter is needed every time.	* once compiled it is not needed.
* partial execution if error	* No execution if an error occurs.
* Easy for programmers.	* usually not as easy as interpreted ones.

Is java compiled or interpreted?

Java is a hybrid language → both compiled as well as interpreted.



→ A JVM can be used to interpret this bytecode.

⇒ This bytecode can be taken to any platform (win/mac/Linux) for execution.

⇒ Hence java is platform independent (write once run everywhere).

Executing a java program:-

Javac Harry.java → compiled

Java Harry.class → Interpreted

So fare the execution of our program was being managed by ~~intellij~~ intelly Idea.

We can download a source code editor like VScode to compile & execute our java programs.

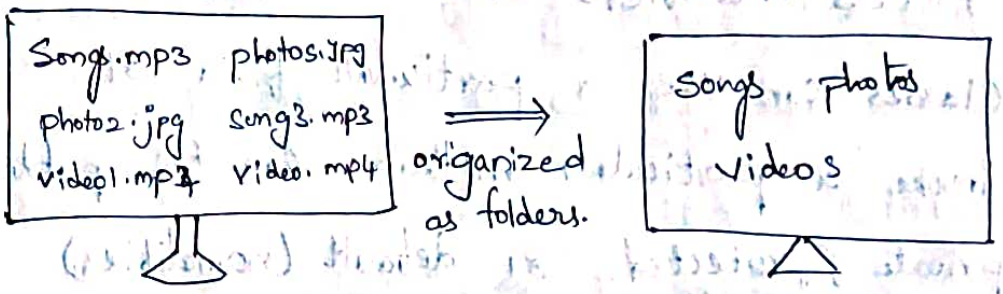
Packages in Java :-

A package is used to group related classes.

Packages help in avoiding name conflicts.

There are two types of packages :-

* Built in packages → Java API
* user defined packages → Custom packages.



1. class this java my.mp3 ⟹ organised as
Song.java harray.java packages.

Using a java package :-

import java.lang.* → import everything from java.lang

import java.lang.string → import string front java.lang.

S = new java.long.string ("Harry") → use without importing.

Creating a package

Javac Harry.java → creates Harry.class

Javac -d . Hary.java → creates a package folder
↳ we can keep adding classes to a packages like this.

We can also create packages by adding "package.inner" as package name.
↓ folder, ↓ subfolder.
These packages once created can be used by other classes.

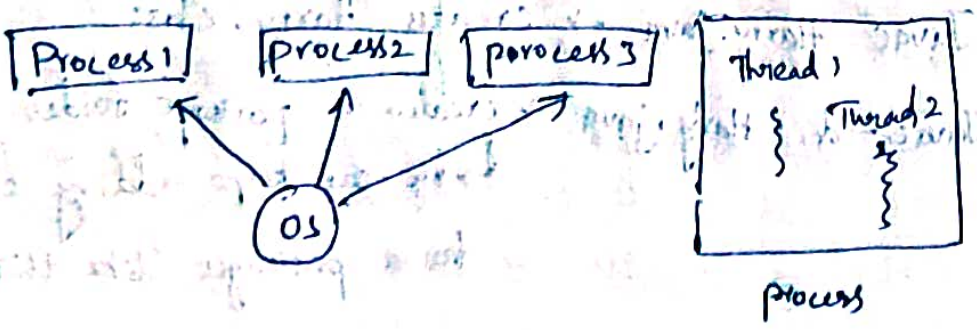
Access Modifiers in java.

Access modifiers determine whether other classes can use a particular field or invoke a particular method. can be public, private, protected or default (nomodifier)

Modifier	class	package	subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
default (No)	Y	Y	Y	N
private	Y	N	N	N

Multithreading

Multiprocessing and multithreading both are used to acheive multitasking.



In a nut shell.....

→ Threads use shared memory area.

→ Threads => Faster content switching

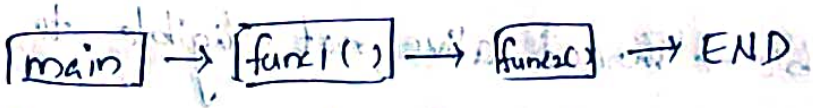
→ A thread is light-weight whereas a process is heavy weight.

For Example :-

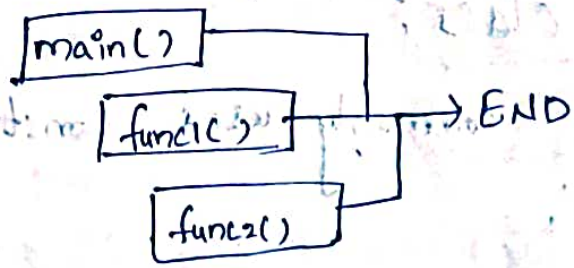
A word processor can have one thread running in foreground as an editor and another in the background auto saving the document.

Flow of control in Java :-

1. Without threading :



2. With threading

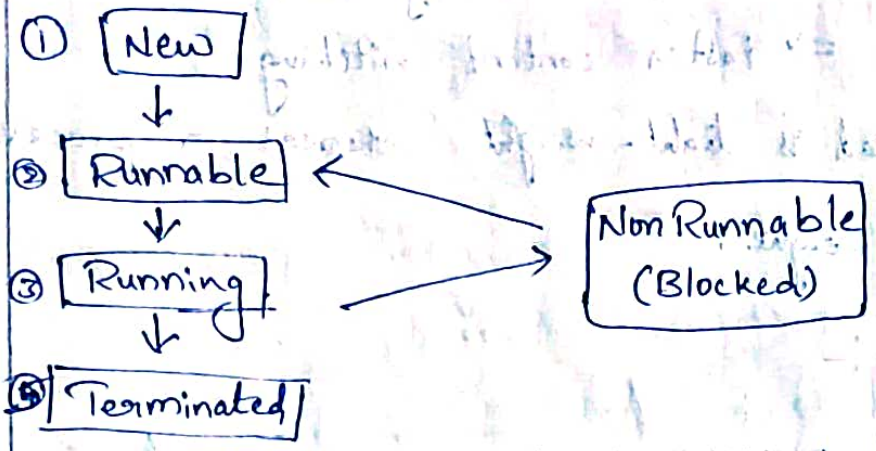


Creating a thread :-

There are two ways to create a thread in java.

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

Life cycle of a Thread.



1. New → Instance of thread created which is not yet started by invoking start().
2. Runnable → After invocation of start() & before it is selected to be run by the scheduler.
3. Running → After thread scheduler has selected it.
4. Non Runnable → Thread alive, not eligible to run.
5. Terminated → run() method has exitted.

The Thread class:

Below are the commonly used constructors of Thread class:

① Thread()
② Thread (String name)
③ Thread (Runnable)
④ Thread (Runnable, String name)

Method of Thread class :-

Thread class offers a lot of methods such as run(), start(), join(), getPriority(), setPriority() etc. More can be found on visiting java docs.

Errors & Exceptions

No matter how smart we are, errors are our constant companions with practice, we keep getting better at finding & correcting them.

There are three types of errors in Java

1. Syntax errors
2. Logical errors
3. Runtime errors → Also called exceptions!

Syntax Errors :-

When compiler finds something wrong with our program, it throws a syntax error.

```
int a = 9  → No semicolon, syntax error!
a = a + 3;

d = 4  → variable not declared, syntax error!
```

Logical errors

A logical error or a bug occurs when a program compiles and runs but does ~~not~~ the wrong thing.

→ Message delivered wrongly.

→ Wrong time of chats being displayed.

→ incorrect redirects!

Runtime errors :-

Java may sometimes encounter an error while the program is running. These are also called exceptions!

These are encountered due to circumstances like bad input and (or) resource constraints.

Ex:-

User supplies 's' + 8 to a program which adds 2 numbers.

Syntax errors and logical errors are encountered by the programmer where as Runtime errors are encountered by the users.

Exceptions in Java :-

An exception is an event that occurs when a program is executed disrupting the normal flow of instructions.

There are mainly two types of exceptions in java :-

1. Checked Exception → compile the exception (Handled by compiler)
2. Unchecked Exception → Runtime Exceptions.

Commonly Occuring Exceptions :-

Following are few commonly occuring exceptions in java :-

1. Null pointer Exception
2. Arithmetic Exception
3. Array Index outof Round Exception
4. illegal Argument Exception
5. Number format Exception.

try - catch block in Java :-

In java, exceptions are managed using try-catch blocks.

Syntax :-

```
try {
    // code to try }
Catch (exception e) {
    // code if exception
}
```

Handling specific Exceptions :-

In java, we can handle specific exceptions by typing multiple catch blocks.

```
try {
    // code
    }
catch (IOException e) {  → Handles all Exceptions of
    // code                type IO exception.
    }
catch (ArithmeticException e) {  → Handles all exception of
    // code                        type Arithmetic exception.
    }
catch (Exception e) {  → Handles all other exception.
    // code
    }
```

Nested try-catch.

We can nest multiple try-catch blocks as follows :-

```
try {
    try {
        }
    catch (Ex.e) {
        }
}
Catch (Ex. e) {
    }
```

=> Nested try-catch blocks.

Similarly, we can further nest try catch blocks inside the nested try catch blocks.

Exception class in Java:-

We can write our custom Exceptions using Exception class in java.

```
public class MyException extends Exception {
    //overridden Methods
}
```

The exception class has following important methods:-

1. String to string() → Executed when sout(e) is ran
2. void printStackTrace() → prints stack trace.
3. String getMessage() → prints the Exception Message

The Throw keyword:-

The throw keyword is used to throw an exception explially by the programmer.

```
if (b==0) {
    throw new ArithmeticException ("Div by 0");
}
else {
    return a/b;
}
```

In a similar manner, we can throw user defined exception:

```
throw new MyException ("Exception thrown");
```

The throws Exception:-

The java throws keyword is used to declare an exception. This gives an information to the programmer that there might be an exception so its better to be prepared with a try catch block!

```
public void calculate (int a, int b) throw IO
Exception {
        //code
}
```

Java finally block :-

Finally block contains the code which is always executed whether the exception is handled or not.

It is used to execute code containing instructions to release the system resources, close a connection etc...

Advanced Java - 1

Collections Framework :-

A collection represent a group of object. Java Collection provide classes and interfaces for us to able to write code quickly and efficiently.

Why do we need Collections :-

We need collections for efficient storage and better manipulation of data in Java.

Ex:- we use arrays to store integers but what if we want to.

=> Resize this array?

=> Insert an element in between?

=> Delete an element in Array?

=> Apply certain operations to change this array?

How are collections available?

Collection in java are available as classes and interfaces. following are few commonly used Collection in Java :-

* Arraylist → For variable size collection.
* Set → For distinct collection.
* Stack → A LIFO data structure.
* HashMap → For storing key-value pairs.

Collection class is available in java.util package Collection class also provides static methods for sorting, searching etc...

Date & Time in Java :-

Java.time → package for Date & time in Java.
↳ from Java 8 onwards.

Before java 8, java.util package used to hold the date and time classes. Now these classes are deprecated.

How java stores a Date?

Date in java is stored in the form of a long number. This long number holds the number of milliseconds passed since 1 jan 1970.

Java assumes that 1900 is the start year which means it calculates year passed since 1900 whenever we ask it for years passed.

The Date class in Java.

```
Date d = new Date();
System.out.println(d);
```

we can also use constructors provided by the Data class.

Java Date class has few methods which can be used for ex:- getdate(), getDay() etc...

All these methods are deprecated.

Calander Class Methods:-

1. get method is used to get year, date, min, second.

```
a.get(Calender.SECOND)
a.get(Calender.MINUTE)
a.get(Calender.DATE)
a.get(Calender.YEAR)
```

2. get time method returns a Date object.
3. Other methods can be looked up from the java docs!

Gregorian Calender class

This class is used to create an instance of gregorian calender.
We can change the year month & date using set method!

Time Zone.

Time zone class is used to create time zones in java. Some of the important methods of Timezone class are:

getAvailableIDS() → get all the available IDs supported
getDefault() → get the default timezone
getID() → get the ID of a Timezone.

Java.time package.

→ Available from java 8 onwards
→ capable of string even nano seconds.

Following as some of the most commonly used classes from java.time package.

LocalDate → Represents a Date

LocalTime → Represents a Time

LocalDateTime → Represents a Date + time

DateTimeFormatter → Formatter for displaying & parsing date-time objects.

Creating our own Java Documentation

Java documentation is great!

It helps us get info about which method/class/entity to use when.

We can create our own packag is Documentation in java.

Javadoc tool :-

Java doc command allows us to create documentations in HTML format for our own package.

Java provides tags for clss in package to assist with the java doc generation.

Tags for class or a package:

1. @author : Adds the author name.
2. @version : Adds the version.
3. @since: To add when was this version written.
4. @see : Adds a see also heading with a link.

Tags for Methods:

Javadoc provides following tags for methods:

1. @param → for describing parameter of a method.
2. @return → for describing about the return value.
3. @throws → For describing exceptions thrown.
4. @deprecated → for describing deprecation status.

Description can be added at the start of javadoc comment.

Annotations in Java

used to provide extra information about a program Annotations provides metadata to class/methods.

Following are some common annotations built into java.

1. @Override → used to mark overridden elements in the child classes.
2. @SuppressWarnings → used to suppress the generated warnings by the compiler
3. @deprecated → used to mark deprecated methods
4. @FunctionalInterface → used to ensure an interface is a functional interface.

Lambda Expressions:-

Added in Java 8

Lambda expressions let us express instances of Single method classes more compactly.

Anonymous classes are used to implement a base class without giving it a name.

For classes with a single method even anonymous classes get slightly excessive & cumbersome.

Java Generics

Introduced from JDK 5.0 onwards

very similar to c++ templates (but not the same)

If we write

```
Arraylist a = new Arraylist ();
a.add (75)
int anum = a.get(0) → we can't do this.
int anum = (int) a.get(0) → we will have to do this.
```

Hence generics aim to reduce bugs & enhance type safety.

Note: Type parameter in java generics cannot be a primitive data type.

File Handling In Java

Reading from & writing to files is an important aspect of any programming language.

We can use the file class in java to create a file object.

↳ Create NewFile() method → creates a file object.

↳ For reading files we can use the same Scanner class and supply it a file object.

↳ To delete a file in java we can use file object delete() method.

Introduction to oops

object oriented programming tries to map code instructions with real world making the code short and easier to understand.

What is object oriented Programming?

Solving a problem by creating objects is one of the most popular approaches in programming. This is called object oriented programming.

What is DRY?

DRY stands for → Do not repeat yourself
↳ focuses on code reusability

Class :-

A class is a blue print for creating objects.

JEE application form => filled by an student => Application for that student

class => object Instantiation => Object

Contains info. to create a valid object.

Object.

An object is an instantation of a class. When a class is defined a template (info) is defined. Memory is allocated only after object instantiation.

How to model a problem in oops.

we identify the following:

Name. → Class → Employee

Adjective → Attributes → name, age, salary

Verb → Methods → getSalary(), increment()

OOPs terminology:—

1. Abstraction:- Hiding the internal details [show only essential info!]

⇒ Use this phone without bothering about how it was made.

2. Encapsulation:- The act of putting various components together (in a capsule)

⇒ Laptop is a single entity with wifi + speaker + storage in a single box!

In java, encapsulation simply means that the sensitive data can be hidden from the users.

3. Inheritence :- The act of deriving new things from existing things.

Rickshaw => E-Rickshaw

phone => Smart phone.

Implements DRY !

4. Polymorphism :- one entity many forms.

Smartphone → phone Smartphone → calculator

Writing a custom class.

We can write a custom class as follows :-

```
public class Employee {
    int id;      → Attribute 1
    String name; → Attribute 2
}
```

Any real world object = properties + Behaviour

object in oops = Attributes + Methods.

A class with methods :-

We can add methods to our class Employee as follows :

```
public class Employee
{
  public int id;
  public String name;

  public int getsalary()
  {
    // code
  }
  public void getdetails()
  {
    // code
  }
};
```