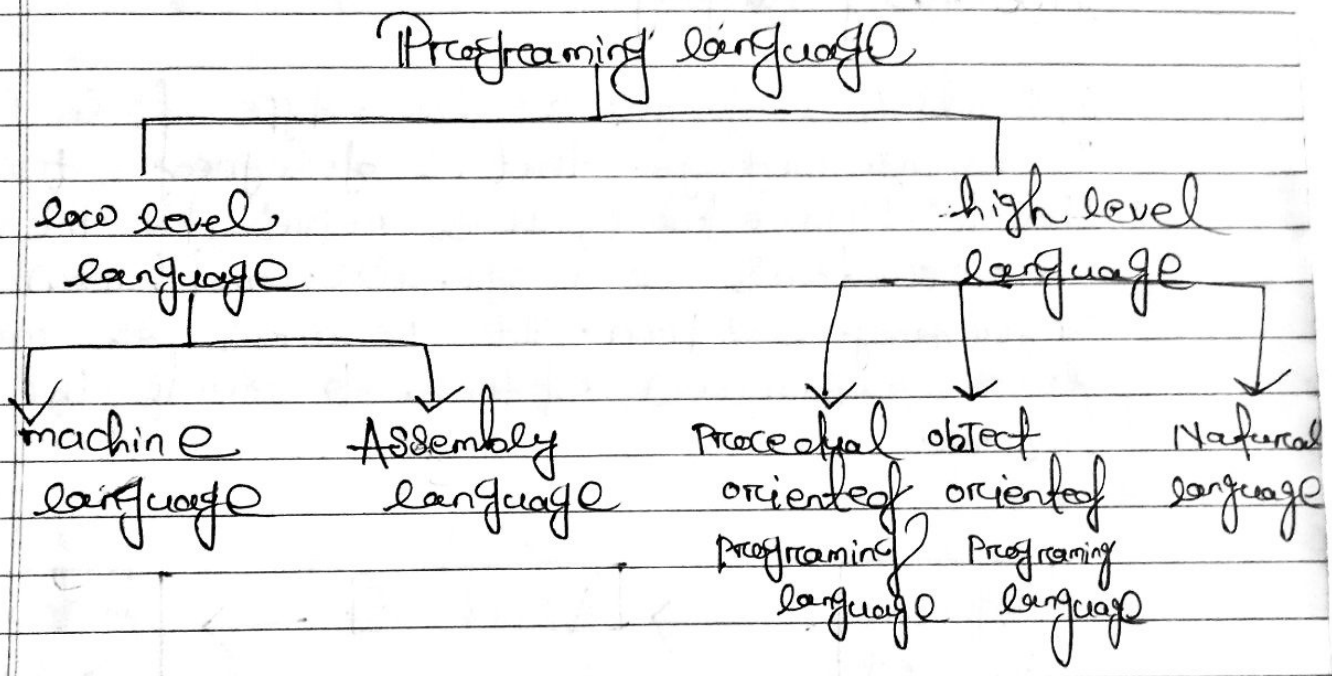
Programing Language :

* A Programing language is a computere language, ie used by Programmers to communicate with the computere.
* It is a set of instruction written any in a specific language (C, C++, Java, Python). to perform a specific task.



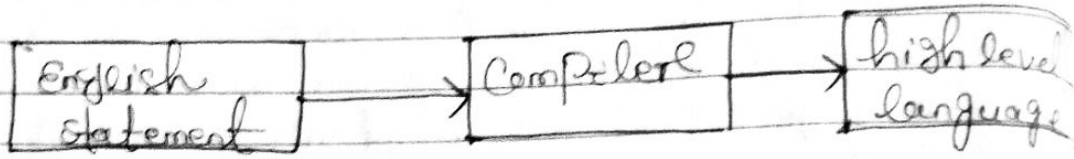
loco level language

loco level language is a machine depend as (and L) Programing language. The Processere reads loco level Programing language directly without any need of Compilere and/or interpeter.

machine level language

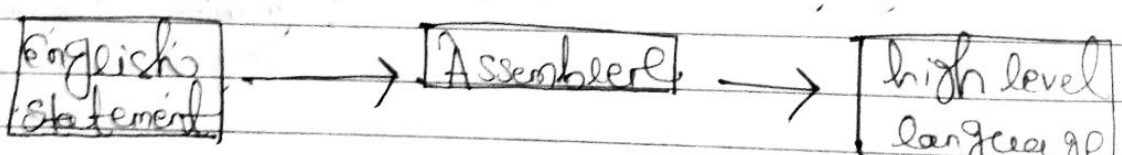
machine level language is a type of loco level Programing language and it is also called as machine Code or object Code. It is helps the Programmer to excute the Programmers faster

than the high level Programming language.



Assembly language

Assembly language is also a type of low level Programming language that is designed for specific procedures, It is represents the sets of instructions in a symbolic and human understandable form. It required less memory less execution of time to excute the program.

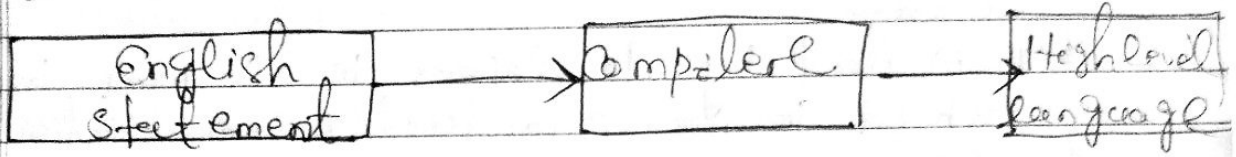


Procedual oriented Procedual oriented Programing language:

Procedual oriented Programing language is derived from structure Program based when the procedual all Concept. It is divides a program into small Procedual called

High level language :-

* High level language is a designed tore developing user friendly software programs and websites. The programing language required a compiler and an interpreter to translate the program into machine language (0 and 1).
ex :- C++, C, Java, Python.



Proceafual oriented Programing language (POP)

* It is derived from structure programing and based upon the procedure or concept, It devides a program into small procedures called Rotines and functions. It helps the programmers to easily talk the program flow and code can be reused in different parts of the program.

OBJECT ORIENTED PROGRAMMING LANGUAGE (OOP)

object oriented Programing language is based upon the object in this Programing language. Programs are divided into small parts called object. It is used to implement the real world entities like, Inheritance, Polymorphism abstraction, encapsulation etc.

→ OOP method is commonly used to develop software packages.

→ The main advantages of oops is faster, easier to execute, maintain, modify etc. as usere as debug (c++, Java, Phython).

Natural language

* Natural language a part of human language such as english, Russia. It is used by machine understand, manipulate and interpret human's language
* It is used by developers to perform task such as transaction, automotic summarization etc.
* main advantages of Natural language is that it helps user to ask question in any subject and directly respond with in second.

Middle level language

* Middle level language lies between the low-level programming language and high level language.
* It is also known as intermediate programming language and pseudo language
  e.g :- c, ~~Python~~, c++

object oriented programming concept :-
object oriented programming that four basic concepts :-

1. encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

(1) Encapsulation :-

-> Binding both code and data member all together into a single unit is known as encapsulation. A Java code is the example of encapsulation and Java bean is the freely encapsulated class because all the data members are private here.

(2) (II) Data abstraction :-

* It is the process of hiding the background details and show only the functionality.
* It is achived by abstract data types.
* It is achived by using Abstract class.
* 0-100% data abstraction is normally achieved by data abstraction.

e.g -> suppose stack is an abstract data types and it contains some methods like push pop peek etc and how the operation is going on this is unknown to the end user.

(3) Inheritance :-

* It is the process of creating a new type from an existing type.
* when one object acquires all the properties and behaviour of present object is called as inheritance.

e.g; — plane and fighter plane are two diffrent classes the instance of fighter plane class will be a new

instance of parent type, it has been inheritance

(4) Poly morphism :

→ one task is going to be performed in different ways.
→ Achived two different ways compile time and runtime.
→ solved using function overloading and function overriding.

CLASS :

* class is a blue print of an object or logical entity.
* It is the keyword to creat users defined type.
* It is a class name present in side Java or package.
* All totally if we will say a class is normal an encapsulated area of properties, value and method.

OBJECT :

* It is an real life enitity which an existance.
* from language point of view it is a simple variable.
* from programmers point view it is a simple memory location.
* object is the supper class of Java object present on side Java ons package

Reusability:

* In heritance is the real feature to data reusability if you need a same functionality is multiple class then you can write a common class for the same functionality and inherit those class to subclasses.

Cost of maintenance:

* It is easy to maintain and modify existing code as new object can be created with small differences to existing ones.

Security:

using data hiding and abstraction only necessary should be provided thus maintain the security of data.

Difference of OOPS

Object oriented system:

* object oriented world software system provide the extensibility, creating object oriented real world, enhanced, expression.

Object oriented Data base:

* they are also called object data base management system (oDBMS) these data base store objects instead of data such as

real, numbers, bcd real integers.

Real time system design :-

* Real-time system inherent complexities that make it difficult build them object oriented techniques make it easier to handle these complexities.

Simulation and modelling system

* It is difficult to model complex system due to the varying specification of variables, object oriented programing provides an alternative approach here simplifying these complex modeling system.

Hypertext and Hypermodel

* OPPS also helps in laying out a frame work for hypertext. Hypertext is similar to regular text, as it can be stored, searched and edited easily.

CIM / CAD / CAM

* OPPS can also be used in manufacturing and design applications, as it allows people to reduce the effort involved. OPPS make possible for the designers and engineers to produce these flow charts and blue prints accurately

~~Encapsulation~~ :- Benefits of OOPS :-

Re-usability :-

write once ~~and use it~~ and use it multiple times" you can achives this by using class.

Redundancy :-

Inheritance is the great feature for data redundancy if you need a same functionality in multiple class you can write a common class for the same functionality and inherit that class to subclass.

Easy maintenance :- It is easy to maintain and modify existing code as new object can be created with small differences to existing ones.

Security :- using data hiding and abstraction only necessary data should be provided. JAVA PROGRAMING thus maintain the security of data.

* write a Program to calculate the addition of two numbers

```
-> class Test
{
public static void main (String args[])
{
 int a=7;
 int b=6;
System.out.println(a+" "+b);
        int sum = a+b;
System.out.println("The Sum of two nos:"+sum)
}
}
```

* write a Program to calculate the modulus of two numbers.

```
> class Test
{
Public static void main (String args[])
{
  int a=5;
  int b=4;
System.out.println("a"+"b");
     int mode=a%b;
System.out.println("modulus of two nos:"+mode);
```

Path TEST

Temporary - way to set JDK 1.8
→ go to cmd
→ then TYPE
set path = "copy address of 1.8 JDK"
→ Then enter
go to c drive
copy the address of Java 1.8 bin folder

Delete

Control + a = select all.

Permanent: way to set JDK 1.8

Step-1 - go to the PC
" -2 - Then right click
" -3 - go to properties
" -4 - then go to the environmental variable
" -5 - Then click new

variable value / Path
Address Address of bin folder
1.8 JDK

Java

Java - Java is a programming language and an platform independent.

→ Java is high level, robust and secure programming language.

→ It is first developed sun microsystem in a year 1995, James Gosling is known as the father of Java.

→ Before Java is named as "OAK"

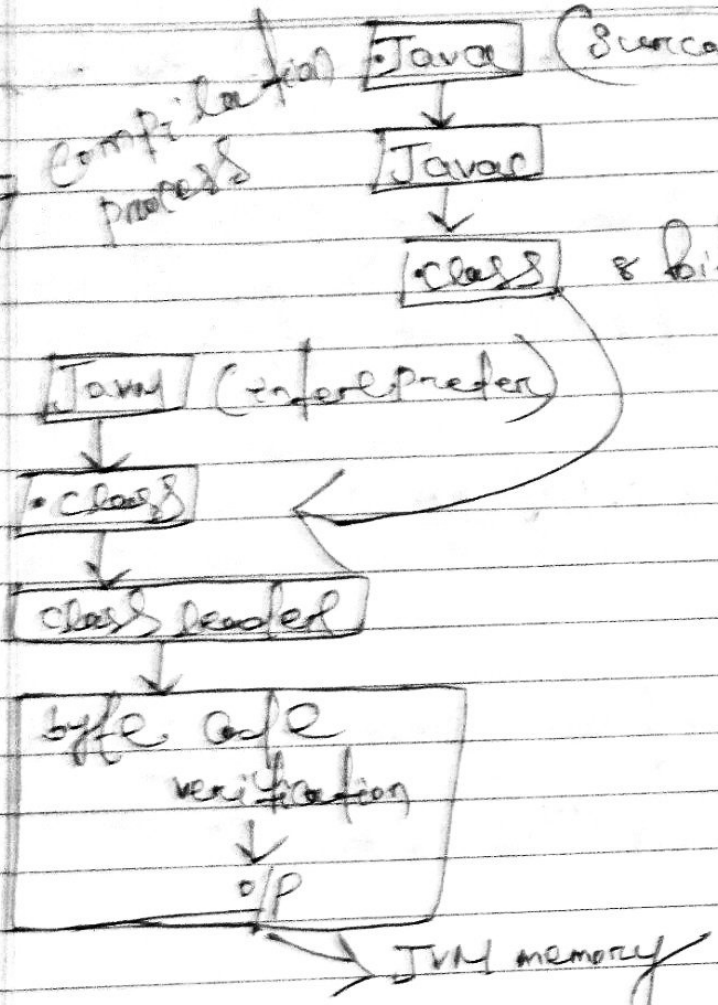
→ Since "OAK" was already a registered company so, James Gosling and his team changed the named from up to Java.

Execution model of Java

→ An execution model specifies how code runs. Every programming language has a execution model, which is specified as the part of language specification.

→ In order to understand the behaviour of a Java program, we must have some model, model of how such a program is execute, here we represent by the Java execution model, which explains the step by step execution of a Java program.

→ Compilation process



★ SOAP to calculate the addition of two numbers by using parseInt

→
```
class Test
{
public static void main (String [] a)
{
  int i,j, sum=0;
i = Integer.parseInt (a[0]);
j = Integer.parseInt (a[1]);
  sum = i+j;
System.out.println ("Addition of two nos;"+sum)
}
}
```

* WAP to calculate the size of the byte

-> class Test

public static void main(String[] args)

System.out.println(SIZE.byte);

First Java Program

class Test

public static void main(String args[])

System.out.println("Hello Java");

-> Save the above file as Test.java

-> To compile:

javac Test.java

-> To execute

java Test

Variables and Data types :-

Variable

-> A variable is a container which holds the value while the Java program is executed. A variable is assigned with a datatype.

-> There are 3 types of variables.

string college "Aryan"
}
using static variable
class student
{
int rool no; // Instance variable
String name;
static String college = "Aryan"
}
Instance variable / Non-static

→ If the variable is the part of class directly but declare without any static key word is known as non-static variable

object

→ The variable only can be called by object name.

→ But we can call the static variable by class name, object name and directly. Directly possible with in the same class

The Java virtual machine

→ Java virtual machine (JVM) is an engine that provides runtime environment to drive the Java code or application to convert Java byte code into machines language. JVM is a part of Java Runtime environment (JDE). In other programing language, the compiler produces machine code or a particular system. However, Java compiler produces code for a virtual machine known as

Java virtual machine

How JVM works

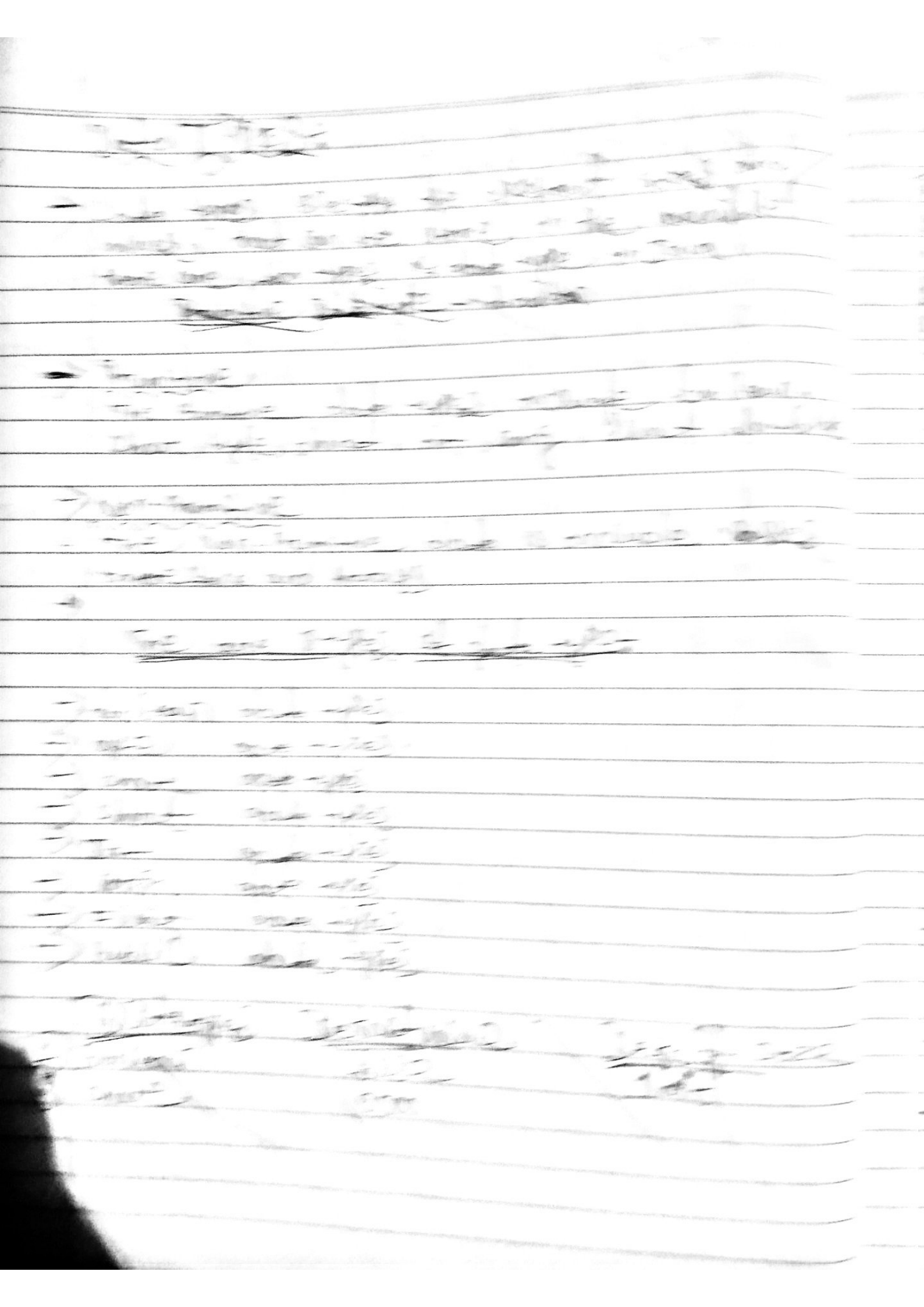
-) First, Java code is compiled into byte code this byte code gets interpreted on different machine.
-) Between host system and Java source, Byte code is an intermediary language.

Final variable

-> If the variable is final then it is a constant in Java
-> Final variable should be initialize at the time of declaration cannot be, we are modify

Final variable

```
class Test
{
Final static int a=20;
    final int b=10;
Public static void main(String[] args)
{
  System.out.println(a);
      a=21; // error
```



	Datatypes	Default value	Default size
1)	Boolean	→ false	1 bit
2)	char	→ \u0000	2 byte
3)	byte	→ 0	1 byte
4)	short	→ 0	2 byte
5)	int	→ 0	4 byte
6)	long	→ 0L	8 byte
7)	float	→ 0.0f	4 byte
8)	double	→ 0.0d	8 byte.

* **Byte :-** → The byte data types is an 8 byte since 2's Complement integer.
→ It has a minimum value of -128 and maximum value of +127
* **SHORT :-** → The short data types is a 16 bit signed 2's Complement integer.
→ It has a minimum value of -32768 and maximum value of +32767.
* **Integer :-**
→ By default the integer data types is 32 bit signed 2's Complement integer
→ which has minimum value -2147483648 and maximum value +2147483647
* **Long :-**
→ The long data type is a 64 bit 2's Complement integer
→ It has a minimum value of $-9.223 * 10^{98}$ and maximum $+9.223 * 10^{28}$
* **Float :-**
→ The float data type is a single Precision 32 bit floating.
→ It has a minimum -2147483648 maximum +2147483647

Double :-

→ The Double data type is a precision 64 bit floating point.

→ It has a minimum $-4.9 * 10^{28}$ and maximum $+4.9 * 10^{28}$

Char :-

→ The char data types is single 16 bit char.

→ It has minimum -32768 and maximum +32767

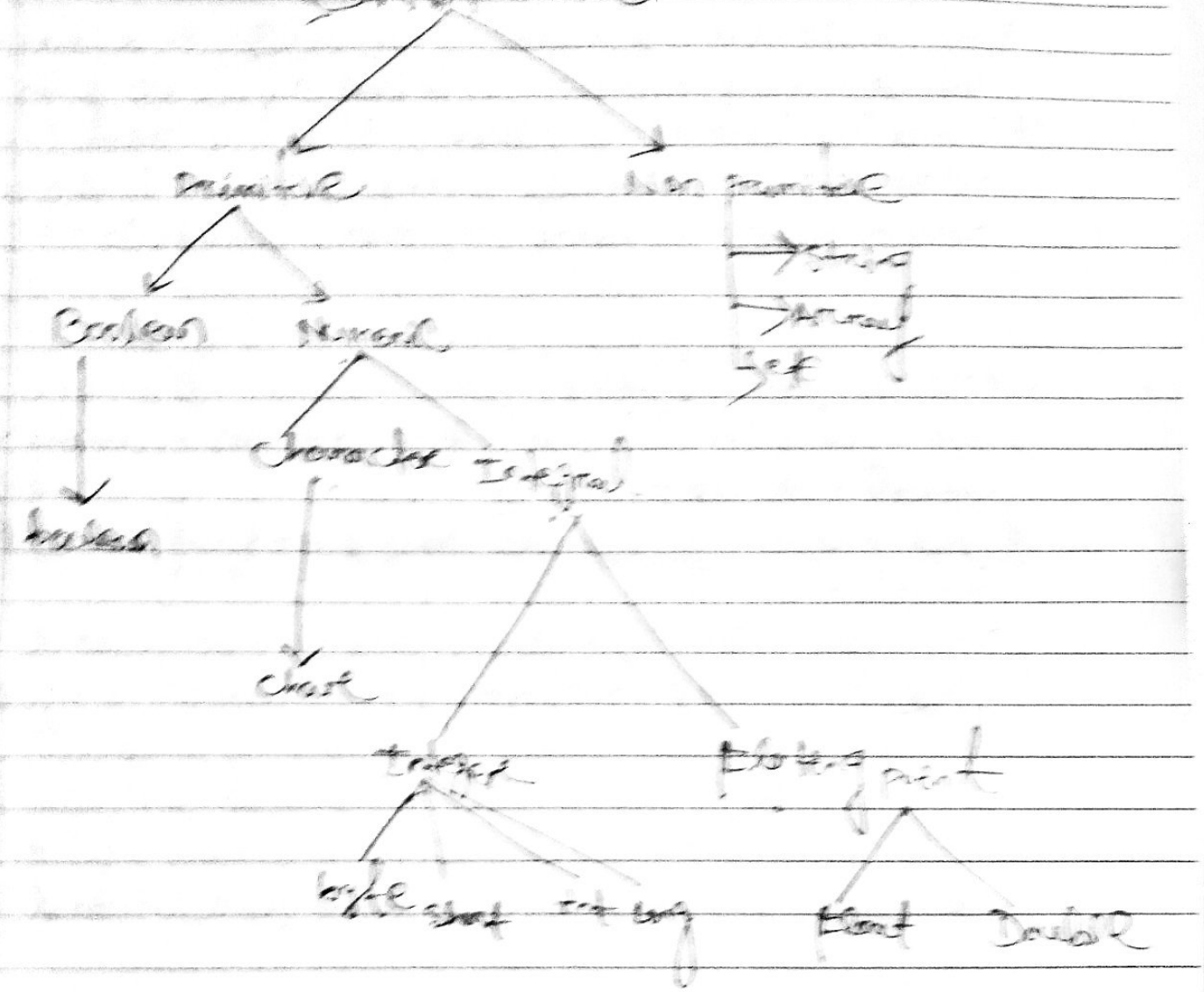
Q Write a program to known the range of data types?

A
```
Class Test
{
public static void main (String [] args)
{
System.out.println (Byte.MAX_VALUE)
}
}
```
(underscore)

*
```
class Test
{
public static void main (String args[])
{
  Int n=29;
  if(n%2==0)
  {
  System.out.println("Number is even")
} else
else
}
```

System and primitive

DATA TYPES



Numeric literals

> Numeric literals are two types
  (i) Exact
  (ii) approximate

i) Exact → An exact numeric literals is a numeric value, contains any decimal point such as 65, -826, -223. Java the Java an exact numeric literals different numbers

1) Single quote :- We can specify the literals to char data type, as single character within single quote.

Syntax :-

```
char ch = 'a';
```

2) char literal as integral literal :-

We can specify the character literals as Integral literals which represents unicode value of the character and that integral literals can be specified either in decimal, octal, hexa-decimal forms. But allowed the range is 0 to 65535.

Syntax :-

```
char ch = 062;
```

3) unicode representation :-

We can specify the character literals in unicode representation '\uXXXX'.
Here, XXXX represents Hexa-decimal number

Syntax :-

```
char ch = '\u0061';
```

4) Escape sequence :-

Every escape character can be specified as character literals.

Syntax :-

```
char ch = '\n';
```

* WAP to illustrate the application of char :-

```
class Test
{
public static void main (String [] args)
```

ex-2

```
class test {
Public static void main (String [] arg)
{
String Temp = "Ram is a good boy";
        String substring = Temp.substring(0,4);
System.out.println(substring);
    }
}
```

Arrays :-

Array is an object in Java which contains similar type of data in continuous memory location.

Syntax :-

or
- data-type [] var.name;
- data-type var.name [];

Ex - int a[];

```
Class Test
{
Public static void main (String args[])
{
int a = 10, b=20, c=30 ;
   int a[] = {10,20,30,40,50};
System.out.println(a[3])
}
}
```

Fixed size static Array :- (declaration)

```
class test
public
class Test
{
Puber. static voif main (string[] args)
{
  inta[] = new int [5];
  a[0] =10;
  a[1] =20;
  a[2] = 30;
  a[3] =40;
  a[4] =50;
For (int i=0; i<5; i++)
  {
System.out.Printen (a[i]);
  }
}
}
```

Advantage of Array :-

1. Code optimization :-
2. It makes the code optimize we can retrive ore, share the data easily
2. Random access :-

we can get any data located at any index Position.

Disadvantages of array :

1. size limit :
   We can store only fixed size of elements in the array it doesn't grow its size at runtime. To solve this problem, collection of framework is used in Java.

→ There are two types of array
1. single dimension of array
2. multi dimension of array

single dimension of array

Array can be constructed in Java by 3 steps
1st one is declaration

(1) Declaration
(2) Instantiation
(3) Initialization

syntax of Declare an array :
   data type [] array Refvar ; or
   data type[] array Refvar ;
   data type array Refvar[]

```
class Test {
public static void main (String[] args)
{
int arr[][] = {{1,2,3},{2,4,5},{4,4,5}};
    for(int i=0; i<3; i++){
      for(int j=0; j<3; j++){
    System.out.Println(arr[i][j]+" ");
      }
      System.out.println();
      }
```

```
double c = b
System.out.println(a);
System.out.println(b);
System.out.println(c);
}
}
```

Explicit casting :-

- This type of casting involves assigning a data type of bigger range to lower range.
- This is done manually as you need to do the casting using "()" with each operand
- If we fail to do the casting a compile time error will be returned by the compiler.

Double → float → long → int → char → short → byte

```
class main {
psvm(String[] args)
{
double d = 57.17;
int i = (int) d;
System.out.println(d);
System.out.println(i);
}
}
```

Operators in Java :-

- Operators in Java is a symbol that is used to perform operation.

ex +, −, *, /, %.

- There are may types of operator in Java which are given Java.

→ unary operator
→ arithmetical operator
→ relational operator
→ bitwise operator
→ logical operator
→ ternary operator
→ assignment operator

unary <
- postfix
- prefix — exp++, exp--

unary <
- postfix → i++, j++, i--, j--
- prefix → ++i, ++j, --i, --j
  ++exp, --exp

Arithmetic <
- multiplicative - *, /, %
- additive +, -

shift - <<, >>, >>>
relation - <, =, >, >=

bitwise <
- bitwise and → &
- bitwise exclusive → ^
- bitwise inclusive → |

logical <
- logical AND → &&
- logical OR → ||

Ternary - ternary - ?:

Assignment - assignment =, += , -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=

Expression :-

→ Java expression consist of a variable, operator, literals and method.

→ Example :- int score

Score = 90; //

or

int score = 90;

→ Here, score = 90 is an expression that returns and integer.

→

* Double a = 2.2, b = 3.4, result;

result = a+b;

Here, a+b is an expression.

Control flow statement :-

→ Java compiler executes the code from top to bottom. The statement is a code, are execut according to the order in which they appear.

→ However, Java provides statements that can use to control the flow of Java code, such statment are called of control flow statment. It is one of the fundamental features of Java which provides a smooth flow of program.

→ Java provides 3 types control flow statment

1. Desision statment.
   - → If statement
   - → switch statment
2. Loop statement
   - → do while loop
   - → while loop
   - → For loop
   - → For - each loop

3. Jump statement
- break, statement
- Continue statement

MODULE-3

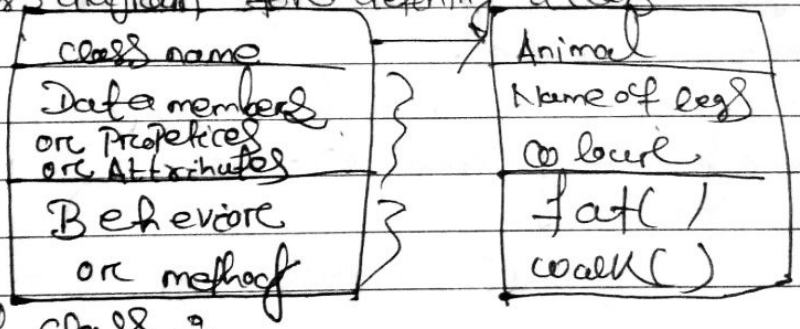
OBJECTS AND CLASSES:

Concept and Syntax of class:

→ A class is a way of binding the data and associated method in a single unit.

→ Any Java program if we want to ~~develop~~ develop then that should be develop than that should be developed with respective class only i.e, without class there is no Java Program

→ A class in Java can contain data member, method, constructor, block class diagram fire defining a class



Syntax of class:

```
class <class-name>
{
  data member
  method member
}
```

→ Here, class is a keyword which is used for developing or creating user defined datatypes

Concept of methods

→ A method is a collection of statements that perform some specific task and return the result to the caller.

→ A method can perform some specific task without

returning anything.
Syntax of a method :-
The syntax to declare a method is return type method name
() {
// method body
}
Eg :-
int addNumber() {
// code
}
Defining methods,

→ A method is a block of code which only runs when it called.
→ You can pass the data known as parameters into a method.
→ methods are used to perform certain actions, and they are also known as functions.
e.g :-

```
Public class main
{
Static void main method()
}
```

Creating an object

→ The object is a basic building block of an OOPs language. In Java, we can't execute any program without creating an object.
→ There is a various way to create an object in Java that we will discuss in this section.
→ Java provides three ways to create an object.

welcome(msg)
Syntax of creating an object = new className();
Public class Test
{
void show( )
{
S.o.p.ln (welcome to ...
}
P s v m (String[] args)
{
Test t = new Test();
t.show();
}
}
}
Access of the members of a class from another class in Java
→ To access the members of a class from other class
→ first of all, import the class
→ create an object of that class
using this, object access the members of that class
Suppose there is a class in a package called my package with a method named display()
package mypackage
Public class Test {
Public void display() {
System.out.println("Hello");
}
}
You can access it
import mypackage
Public class my class {
Public static void main (String [] args)

→ using new keyword
→ using clone() method
→ using newInstance() method of the class.

Using new Keyword:

Syntax of creating an object in Java:-

```
class Name objTest = new class name();
Public class Test
{
void show()
System.
System.out.Println("welcome to Arjun");
}
public static void main(String [] args)
{
Test t, new Test();
t.show()
}
}
```

→ Access of the members of a class from another class in Java:-

→ To access the members of a class from other class.
→ First of all, import the class
→ create an object of that class,
using this object access, the members of that class

* suppose, there in a class in a package called my package, with a method named display

```
Package my Package;
Public class Test
{
public void display()
{
System.out.println("Hello");
```

Access specified in Java

access, per the Java allows, to set the SKOP or accessibility or visibility of data members be it is a field c class or method

Types of access modifiere en Java

Java provides, four types, of access specified that we can use within classes and other entities, there,

(i) Default

whenever the access level, is not specified, then it is as to be a default. The so SKOP of the, default value is with the, package.

(ii) Public

This is the most common access level, and whenever the, Public access specified is used within and entities, that particular entity is, accesable, through, out from, within the out side, the package out side, the class.

(iii) Protected

The accessable has a scKop that is within the package a protect entity is also accesable, outside the package through out inherited class. if you don't make, the child class it cannot be, access it from out side the, package.

(iv) Access control :

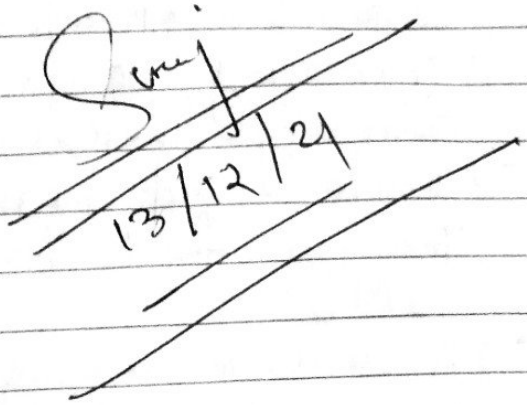
→ In Java accesscontrol, tells, the Program, how much, access a variable, class or method is given.

→ Access control is important because, it affects, visibility based on the different access control.

→ when a variables, or method access is not specified public, or private, it will have default visibility

→ Default visibility is packed Private, It is visible, to all, acces, in the, same package,

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



METHODS AND MESSAGES:

→ A Java method is a collection of statements that are grouped together to perform an operation when you call the system.out.println() method for e.g the system actually executes several statements in order to display a message on the console.

→ Now we will learn how to create our own methods with one, without return values, invoke a method with one without parameters, and apply method abstraction in the program design

Creating method:

Syntax:

```
public static int methodname(int a, int b){
    // body
}
```

Here

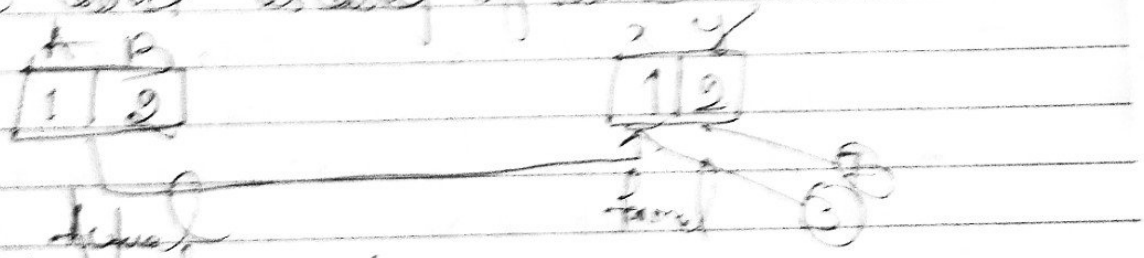
public, static modifiers
int - return type
methodname - name of the method
a, b - formal parameters
int a, int b - list of parameters

Messages:-
-> messaging is a method of communication between software components or applications
-> A messaging client send message to receive message from any other client
-> Each client connects with messaging agent that provides facility to send and receive messages
JMS:-
-> A Java messaging service is a Java API that allows applications to create, send, receive messages
-> JMS is a specification and JMS implementational as JMS provider
-> some JMS providers are ActiveMQ, OpenMQ, oracleAQ
Parameter passing:-
There are different ways in which parameter data can be passed into and out of method and function. Let us assume that a function B() is called from another function A(). In this case A is called the "caller function" and B is called the "called function or callee function". Also, the arguments which A sends to B are called actual arguments and parameters of B are called formal arguments.
Types of Parameters:-
Formal Parameter
A variable and its type as they appear in the prototype of function or method
Syntax function name (data type variable name)

Actual parameter:
The variable or expression corresponding to a formal parameter that appears in the function or method of the calling environment.
func-name(variable, constant);

Different methods of passing:

1. Pass by value:
Changes made to formal parameter don't get transmitted back to the caller. Any modifications to the formal parameter variable inside the called function or method object of the program statement execution are not reflected in the actual parameter in the calling environment. This method is also called as Call by value.



2. Call by Reference:
Changes made to formal parameter do get transmitted back to the caller through parameter passing. Any changes made to the formal parameter are reflected in the actual parameter in the calling environment. Actual parameter refers a reference to the original location. This method is also called as Call by reference.

Inheriting and Interfacing:

→ Java object class, the super class of all the Java classes. All Java classes implements the object class by default

MODULE-5 INHERITANCE

Inheritance in Java (also known as derivation)

→ The term inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object.

→ It is an important part of OOPs (object oriented programming system).

→ The idea of inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of parent class. Moreover, you can add new methods and fields in your current class also.

→ Inheritance represents the IS-A relationship which is also known as parent-child relationship.

Use of inheritance

→ For method overriding.

→ For code reusability.

Method overriding:

If a subclass (child class) has the same method as declared in the parent class, it is known as method overriding.

Reusability:

As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

Types of inheritance:

→ On the basis of class, there can be three types of inheritance in Java:

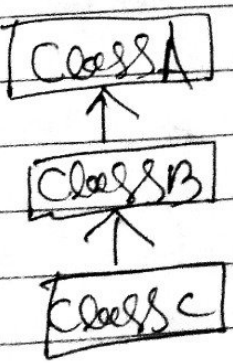
1. Single inheritance
2. Multilevel inheritance
3. Hierarchical inheritance.

→ As you can see, in the example, give below BabyDog class inherits the Dog class which again inherits the Animal class,

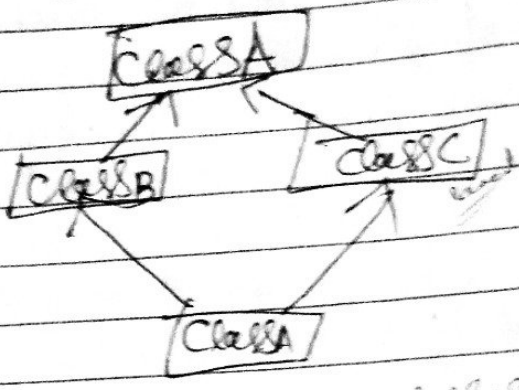
→ so there is a multilevel inheritance.

```
class Animal {
void eat()
{
System.out.println("eating");
}
}
class Dog extends Animal {
void bark(){
System.out.println("barking");
}
}
class BabyDog extends Dog{
void weep()
{
System.out.println("weeping");
}
}
class Test inheritance? {
P sv m (String [] args)
BabyDog d = new BabyDog();

d.weep();
d.bark();
d.eat();
}
}
```



Hierarchical Inheritance:



→ When two or more classes inherits a single class, it is known as hierarchical inheritance.

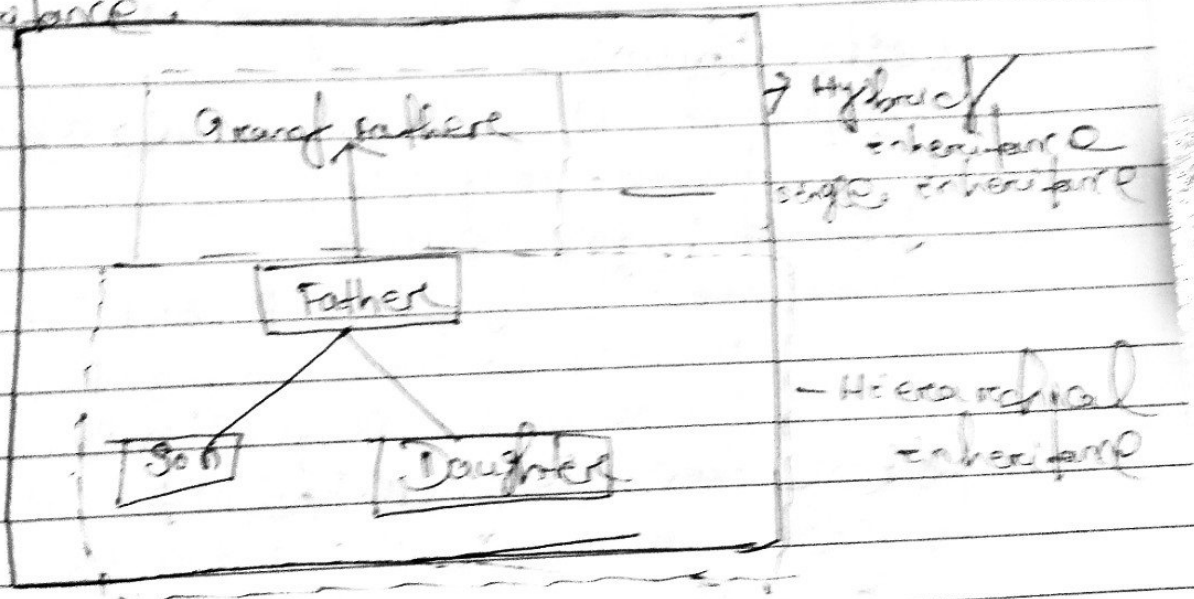
→ In the example given below, Dog and Cat classes inherits the Animal class, so there is hierarchical inheritance.

```
class Animal {
  void eat()
  {
  System.out.println("eating");
  }
}
class Dog extends Animal {
  void bark()
  {
  System.out.println("barking");
  }
}
class Cat extends Animal
  {
  void meow()
  {
  System.out.println("meowing");
  }
}
```

class test inheritance {
public static void main(String[] args)
{
Child c = new Child();
c.calc();
}
}

Hybrid inheritance :-

Hybrid means, group of more than one. Hybrid inheritance is the combination of two or more types of inheritance.



→ In the above figure, grandfather is a base class. The father class inherits the properties of the grandfather class. Since father and grandfather represents single inheritance.
→ Father, the father class is inherited by the Son and Daughter class.
→ Thus, the father becomes the Parent class for Son and Daughter.
→ These classes represent the hierarchical inheritance.

→ continued of it depends the [illegible]

MODULE-6 Polymorphism

Polymorphism ÷
Polymorphism in Java is the ability of an object to have many forms.
Types of Polymorphism ÷

There are two types of polymorphism in Java
1. Compile time polymorphism
2. Runtime polymorphism

Polymorphism ÷
Polymorphism in Java is the ability of an object to have many forms.
Types of Polymorphism ÷
There are two types of Polymorphism in Java
1. Compile time Polymorphism
2. Runtime Polymorphism

Compile time Polymorphism ÷
A polymorphism which is exists at the time of compilation is called compile time or early binding or static polymorphism
e.g. — method over loading

Method overloading ÷

whenever a class contain more than one method have same and different type of parameters called method over loading.

Syntax – return-type method name(

return type method-name (para1, para2)

class A
{
void add()
{
int a=10, b=20, c;
c=a+b
SOPln(c);
}
void add(int x, int y)
{
int c;
c = x+y;
SOPln(c);
}
void add(int x, double y)
{
double c;
c = x+y;
SOPln(c);
}
psvm (String [] args)
A p = new A()
r.add();
r.add(100,200);
r.add(50,45.32);

method same parameter different ex overloading

parameter

Runtime Polymorphism?

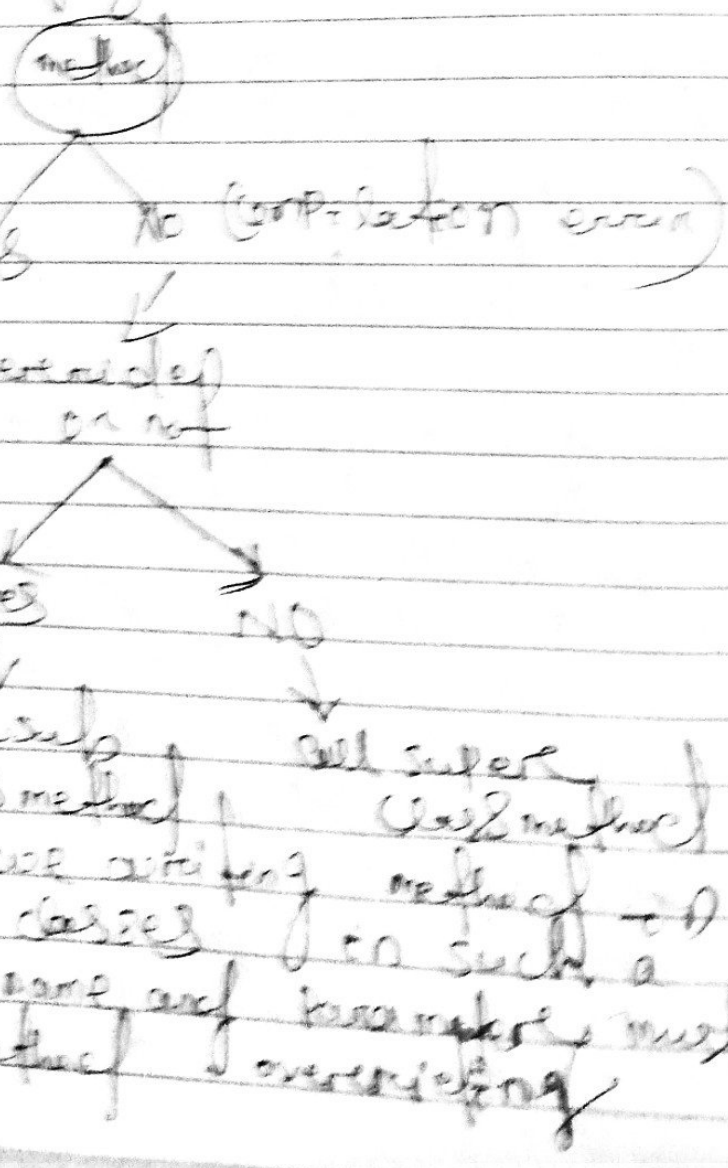
A Polymorphism which exist at the time of execution of program is called runtime polymorphism.

e.g = method overriding.

Syntax:

```
class A
{
void show()
{
}
}
class B extends A
{
void show()
{
}
}
```

Method overriding rules:



Packages : Putting classes together

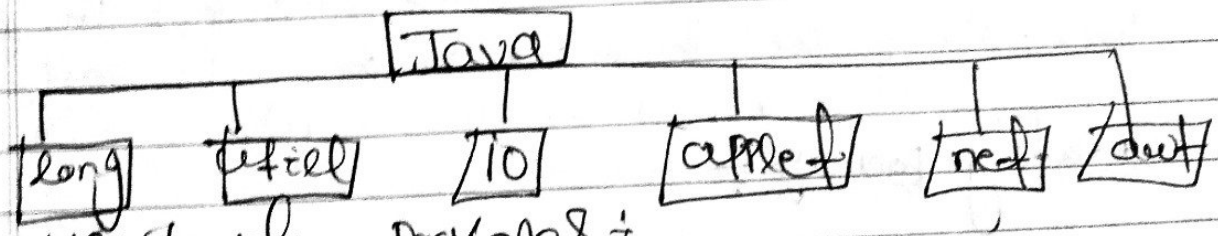
MODULE - 4

Introduction :

- A Java Package is a group of similar types of classes, interfaces and Sub-package.
- Package in Java can be categorized in two form, built-in package and user-defined package
- There are many built in package, such as java, lang, awt, javax, swing, net, io, util, sql etc.

JAVA API Packages :

Java API Application Program interface provides a large number of classes grouped into different packages according to the functionality. most of the time, we use the packages available with the Java API. following figure shows the system Packages that are frequently used in the program.



Using system packages :

Java language : language support classes. They include classes for primitive types. string, math, function, threads, exception.

Java utility : language utility classes such as Vectors, hash table, random numbers, date etc.

Java.io : input/output support classes. They provide facilities for the input and output of data.

Java.applet: classes for creating and implementing applets.

Java.net: classes for networking. They include classes for communicating with local computers as well as with internet servers.

Java.awt: set of classes for implementing graphical user interface. They include classes for windows, buttons, lists, menus and so on.

Naming Convention:

-> Java naming convention is a rule to follow as you decide what to name your identifiers such as classes, package, variable, constant, method etc.

-> But, it is not forced to follow. So it is known as convention not rule. These conventions are suggested by several Java communities such as Sun Microsystems and Netscape.

-> All the classes, interface, package, methods and fields of java programming language are given according to the Java naming convention. If you fail to follow these conventions, it may generate confusion or erroneous code.

Creating package:

Creating a package is a simple task as follows

-> choose the name of the package.

-> include the package command as the first line of code in your source file.

-> The source file contains the classes interfaces etc you want to include in the package

-> compile to create the java packages

PAKAGE IN JAVA

what is Package?

A Package is a group of similare types of classes, interfaces and sub-packages, Package can be categorized in two from.

(i) Built-Package

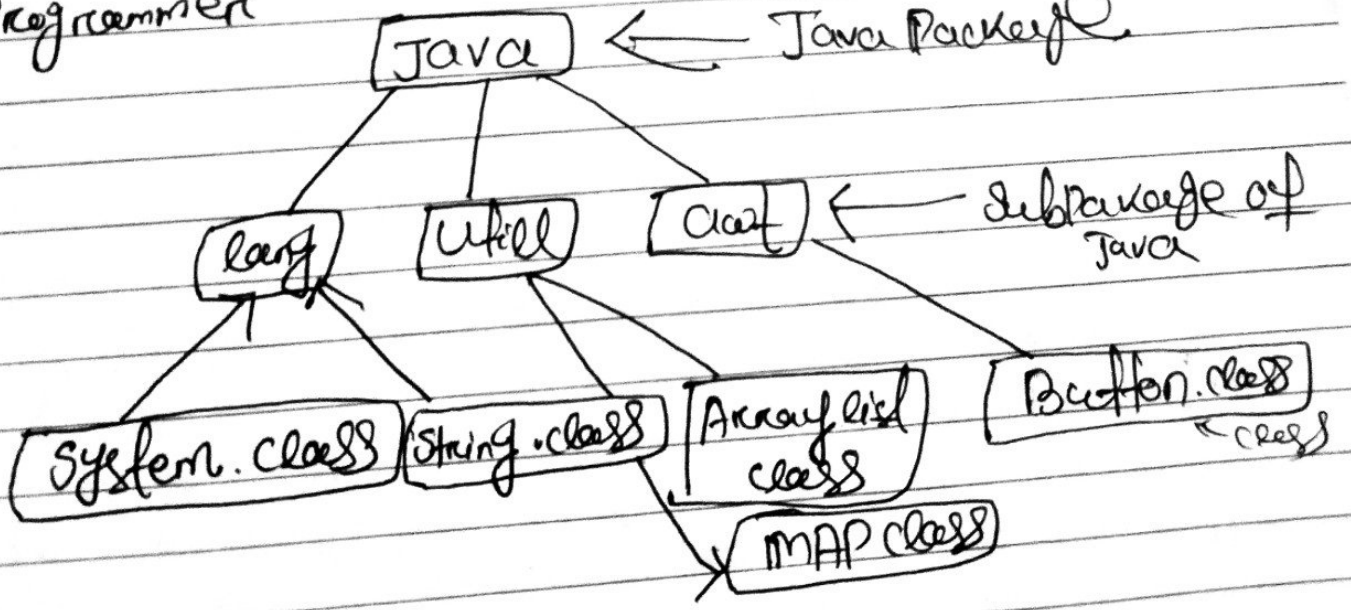
(ii) user-defined Package

Advantage of Package

- Package is used to categorize the classes and interfaces so that they can be easily maintained.
- Package provides access Protection
- Package removes naming collision.

Built-in Package:

Predefined packages are those which are developed by Sun micro systems and supplied as a part of JDK (Java Development Kit) to simplify the task of Java Programmer



User defined Packages

A user defined Package is on which is developed by Java Programmers to simplify the task of the Java Programmers, to keep set of classes, interfaces and sub Packages which are commonly used. Any class or interface is commonly used by many Java Programmers that class or interface must be placed in packages "package" Keyword is used to create a package. whenever we create user defined package we must use package statement as a first executable statement.

EX TO create a package :-

```
package mypack;
public class simple
{
    public static void main(String args[])
    {
        System.out.println("welcome to packages");
    }
}
```

To Compile: javac -d . simple.java

To Run: java mypack.simple

Note - T-d is a switch that tells the compiler where to put the class file. It represents destination. The (dot) Represents the current folder and user may place the class file any directory by giving the directory name e.g javac -d E:/raja/ravi/impp

* How to access Package from another package?

There are three ways to access the package from outside the package.

1. import package-name.*;
2. import package-name.classname
3. Fully Qualified name.

Note: If you use package.* then all the classes and interfaces of this package will be accessible but not of packages.
The import keyword is used to make the classes and interface of another package accessible to current package.
Ex: my package name:
Save by A.Java
package pack;
public class A
{
public void msg()
{
system.out.println("Hello");
}
}
Save by B.Java
package mypack;
import pack
class B
{
public static void main(string args[])
{
A obj = new A();
obj.msg();
}
}
Output:
\SASWAT\DESKTOP\demo > javac -d .AJava

Example of package name: class-Name
// save by A.java

```
package pack;
public class A
{
      public void msg()
      {
            System.out.println("Hello");
}
// save by B.java
package mypack;
import pack.A;
class B
{
      public static void main(String args[])
      {
            A obj = new A();
            obj.msg();
      }
}
```

Output:

C:\users\SASWAT\DESKTOP\demo>javac -d. A.java
" " " " "
" " " "

Hello

If you use fully qualified name then only declared class of this package will be accessible. Now there is no need to import. But you need to use fully qualified name every time when you are accessing the class or interface.

EXAMPLE FULLY QUALIFIED Nm

```
// Save by A.java
Package Peek
Public class A
{
        Public void msg()
          {
          System.out.println("Hello");
          }
}
// Save by B.java
Package mypackage;
class B
{
    Public Static void main(String args[])
      {
            Peek.A obj = new Peek.A(); // using fully
                                          Qualified nam
            obj.msg();
      }
}
```

Java hiding classes:

Hiding Classes:

→ When we import a package with in a Program only the classes declared as Public in that package will be made accessible with in this Program. In other words the classes not declared as Public in that package will not be accessible with in this Program.

→ We shall Practically make use of the above fact. Some times, we may wish that creation classes in a Package should not we made accessible to the

Non-public classes are need not declare with the public keyword when we do so, those classes will be restricted from being accessed by the importing classes. Hence, consider example

```
package mypack;
public class myclass {
    public void display() {
        System.out.println("my class");
    }
}
class Datei class
{
public void display() {
    System.out.println("Datei class")
}
}
```

Here, the class 'Datei class' which is not declared public is hidden from outside of the package mypack. This class can be seen and used only by other classes in the same package. Note that a Java source file should contain only one public class and may include any number of non-public classes

```
import mypack.*
public class JavaApp {
psvm (String [] args) {
Datei class da = new Datei class();
    da.display();
}
}
```

Java Compiler would generate an error message for the code Dealer class ss = new dealer class();

Static import in Java:-

In Java, static import concept is introduced in version. with the help of static import, we can access the static members of a class directly without class name or any object.

e.g import static Java.lang.System class static import Example {

```
    psvm (string) args)
    {
    out.println("Hello");
    out.println("Java");
    }
}
```

MODULE-8

Java files AND I/O:-

STREAM:-

A stream is a sequence of data. In Java, a stream is composed of bytes. It is called a stream of water that continues to flow.

Reading and writing to files:-

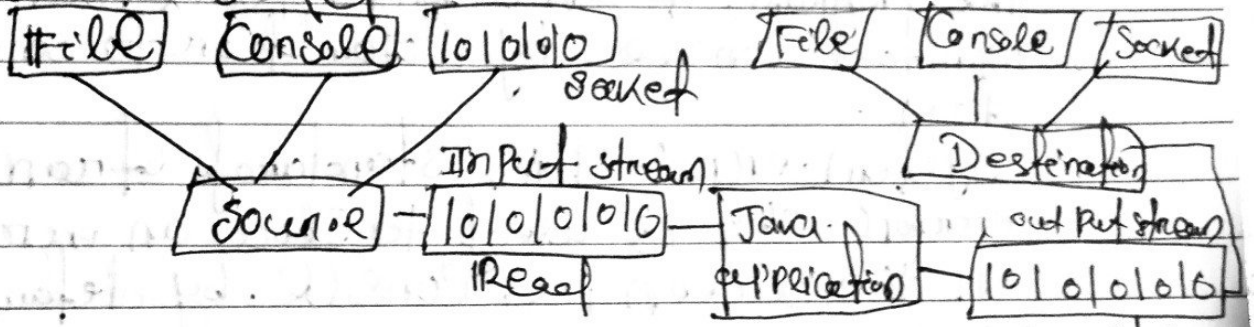
Java FileWriter and FileReader classes are used to write and read data from text files (they are character stream classes). It is recommended not to use the FileInputStream and FileOutputStream

stream classes, if you have to read and write any textual information as these are byte stream classes.

File Reader:
It is useful to read data in the form of characters from a "text" file

output stream / input stream:

Output stream:
Java application uses an output stream to write data to a destination. It may be a file, an array, peripheral device or socket.



Input stream:
Java application uses an input stream to read data from a source. It may be a file, an array, Peripheral device, and socket.

opening and closing streams:
Streams have a base
Streams have a base, stream (closeable) method and implement Auto close, and but nearly all the Stream instances do not actually need to be closed after use. Generally, only streams whose source is an I/O channels.
(such as those returning by files)
lines (Path, charset) will require
laxer)

Predefined streams :-

Three Predefined streams are standard streams are available in java.lang.system class

system.in → This is the standard Stream for input data. This stream is used for reading data for the program from the keyboard by default.

system.out → This standard stream for output. This streams is used for writing data by the program to an output device, seen as a monitor console by default or to some specified file.

System.err :- This standard stream for error. This is used to show an error message on the screen i.e console by default for the user.

File handling methods :-

The File class The file class have many useful methods for creating and getting information about files.

methods	type	Description
canRead()	Boolean	Tests whether the file is readable or not
canWrite()	Boolean	Tests whether the file is writable or not