JAVA NOTES (core)

CHAPTER 1

INTORDUCTION

PROGRAMING LANGUAGE : Prog. Lang. Is a lang. Which provide syntax to develop prog. And perform operations by computer

WHAT IS JAVA?

Ans : java is an object oriented platform independent prog. Lang.

WHY JAVA?

Java is invented to achieve platform independency for developing internet based business application.

WHERE JAVA IS USED?

*DESKTOP APP.(SE)

*DRIVER APP.(SE)

*GAMING APP.(EE)

*ENTERPRISE APP.(EE)

*WEB APP.(EE)

*MOBILE APP.(ME)

FLAVOURS OF JAVA :

1. SE (STAND. EDDITION)

2.EE (ENTERPISE EDDTION)

3. ME (MICRO EDDITION)

HISTORY OF JAVA :

JAVA WAS MADE BY SUN MICRO SYSTEM(ORACAL) FATHER OF JAVA IS **JAMES GOSBLING.**

Java is open source.

DIFFERENCE B/W JAVA & OTHERS(C, C++) :

1. Static vs dynamic
2. Preprocessor
3. Platform dependent vs platform independent
4. Pointer

FEATURES OF JAVA :

To show the nature of java programming lng. Java has provided the following features.

1. Simple.
2. Object oriented.
3. Platform independent.
4. Architectural neutral.
5. Portable.
6. Robust .
7. Secure.
8. Dynamic.
9. Distributed .
10. Multithreaded
11. Interpreter.
12. High performance.

1. Java is simple prog. Lang. Because

- Java app. Will take less memory and execution I.e. because java have removed almost all the conuson features like pointers etc.
- Java is used in all the simplified syntax from c & c++ .

- **OBJECT ORIENTED :** java is an object oriented programming lang. Because java s able to store data n the form of object only.
- Platform independent : java allows it's applications to compile on one operating system and execute on another os.

- **ARCHITECTURAL NUTERAL** : java allows it's application to compile on one hardware architecture and to execute on another h/w architecture.

- **PORTABLE :** java is able to run it's app. Under all the os. & under all the hardware system.

- **ROBUST :** java is robust prog. lang. because java is having very good memory management system in the form of heap memory management, it's a dynamic memory mang. & system it allocates and deallocate memory for the obj. at the runtime. java is very good exception handling mechanism.

- **SECURE :** java is secure prog. Lang. Because java has provided & include compocent nsde jvm in the form of " security manager"
- Java has provided a separate middleware service. In the form of **jaas** (java authentication & authorization service) in the order to provide web security

- **DYNAMIC :**

- **MULTITHREADED :** java s able p provde very good environment to create & execute more than one thread at a time due to this reason java s multithreaded prog. Lang.

- **INTERPRETIVE :** java is based on interpreter

- **HIGH PERFORMANCE :** because use above all al features.

TYPES OF APPLICATION :

STAND ALONE APPLICATION : An application that can only executed in local system with local call is called stand alone application.

ITERNET/WEB BASED APPLCATION : An application that can be executed with local call also from remote system via network call request s called internet based application .

JAVA INSTALLATION:

- Java jdk s/w from oracale.com
- Open .exe file
- A window is open click on next

- In the next window jdk s/w installation directory path is displayed and shown below
- Change above path
- Click on next then jdk installation is started.
- After jdk installed one more window is open for installation ‘jre’ click on next .
- Close the installer.

SETTING JAVA ENVIRONMENT
(CREATING PATH & CLASS PATH VARIABLE)

Just by instakking jdk s/w we can’t use compiler and jvm s/w from all folders of operating system from command from for compiling & executing java prog. We must store jdk s/w bin & lib folder’s path nsde operating system using two variables.

1. **PATH**
2. **CLASS PATH**

PATH : path is a os level environment variable it is used for storing a s/w binary file path for making them available to command form.

CLASS PATH : it’s used for storing library files o make them available to cmd.

JAVA NAMING CONVENTIONS

java is a case sensitive prog. lang. to use lower case letters and upper case letters separately in java app. java has provided the following convention.

1. **CLASS :** All class name, abstract class name , interface name, anom names must be started with upper case letter and the subsequent symbols must also be upper case letter
   FOR EG. :-
   - String
   - StringBuffer
   - SummerTraining
   - Bank
   - ATM
   - SBIBank
2. **VARIABLE :** all java variable must be with lower case letter but the subsequent symbols must be upper case letters.

FOR EG:- firstName,lastName

rollNo

3. **METHODS** : all java methods must start with lower case letter but the subsequent symbol must be upper case.
   FOR EG:-
   - concat()

   getInputStream()
   msgDisplay()
   sleeping()
   messaging()

4. **KEYWORDS** : all keywords always small in letter
   FOR EG:-
   instanceOF()

5. **PACKAGE :** All java package name must be provided in lower case letter
   FOR EG:-
   - java.lang
   - java.util
   - java.sql
   -
6. **CONSTANT :** All java constant variables must be provided in upper case letter
   FOR EG:-
   MIN_PRIORITY
   MAX_PRIORITY

JAVA PROGRAMMiNG FORMETE

To prepare java basic application we hate two use following structure:

1. Comment section
2. Package section
3. Import section
4. Class/interface section
5. Main class section

1. **COMMENT SECTON :-**
    - SINGLE LNE SECTION "//"
      FOR EG :- //PROGRAM FOR A MSG PRINT
2. **PACKAGE SECTION :-**
3. **IMPORT SECTON :-**

```
Import Java.util.Date
Import java.util.*
```

4. **INTERFACE/CLASS :-**

INTErFACE :-

FOR EG:-

```
Interface Bank
{
  Void deposite();
}
```

CLASS :-

FOR EG :-

```
class Bank
{
  Void deposite ();
   {
    Int, amt = 10
   Int  bal = bal+amt;
   }
}
```

INTERFACE TO CLASS :-

FOR EG:-

```
Class SBIBank implements Bank
{
  void deposite()
   {
     Bal= bal+amt;
   }
Public Static void main()
    {
    }
```

}

TOKENS :-

- Tokens are the smallest unit of the program.
    - **KEYWORDS.**
    - **IDENTIFIER.**
    - **DATA TYPES.**
    - **VARIABLES.**
        a) **CLASS(GLOBAL)**
        b) **LOCAL**
        c) **INSTANCE**
    - **LITERALS.**
    - **OPERATORS.**
    - **SEPERTORS.**

1. **KEYWORDS :-** keywords are reserved word it's having special meaning in java. keywords are ment for communicating with compiler and jvm to inform an operation we want to perform in java prog.

LIST OF KEYWORDS IN JAVA:-

PACKAGE CREATION AND USES

a) PACKAGE
b) IMPORT

CLASS CREATION

a) Class
b) Interface
c) Enum

DATA TYPE AND RETURNTYPE

a) byte
b) short
c) int
d) long
e) float
f) double
g) char

MEMORY ALLOCATION

a) new

1. if
2. else
3. else if
4. switch
5. case
6. default

a) LOOP

1. while
2. do while
3. for

jumping :-

1. continue
2. break
3. return

ACCESSIBILITY MODIFIER

1. private
2. protected
3. public

EXECUTION LEVEL MODIFIER

1. static
2. final
3. abstract

4. native
5. volatile
6. transient
7. synchronized
8. strictFp

ESTABLISHING INHERITANCE LEVEL

1. extend
2. implement

ACCESSES OBJECT AND IT'S MEMBER

1. this
2. super
3. instanceOf

EXCEPTION HANDLING

1. catch
2. finally
3. throw
4. throws
5. try
6. assert

UNUSED

1. go to
2. const

IDENTIFIER

Identifier is a name assigned to programming elements like variables, method, classes, abstract classes, interfaces etc.

FOR EG:- SAME AS NAMING CONVENTION

RULES OF CREATION OF IDENTIFIER

To provide identify in java we have to find following rules and regulation:

- identifiers should not be started with any Number.
  Eg:- Sid 9
- identifiers are not allowing all operators and all special except underscore and dollar symbol.
  Eg:- s_id s$id

- we cannot use space in the middle.

  Eg:- sName

  Identifier should not be duplicated within the same scope , identifiers may be duplicated in two different scope.
  Eg:- int Sid
  Float Sid
  Both are not allowed

DATATYPES

java is strictly typed programming lang. where in java application before representing data first we have to confirm which type of data we are representing. in this context to represent type of data we have to use data type.

Data type diagram:

In java applications data types are able to provides following advantages

1. WE ARE ABLE TO IDENTIFY MEMORY SIZE TO STORE DATA.
   FOR EG;- int i=10 (memory size 4byte)
2. WE are able to identify range values to the variable to assign
   For eg
   Byte b=127

VARIABLES

variable is an identifier which is used to stored values there are three types of variables in java(a/c to use):

- **STATIC VARIABLE (GLOBAL)**:- memory allocation during compile time
  for eg:- static int sub=5;
- **INSTANCE VARIABLE** :- scope class level memory allocation runtime when we create object.
  For eg :- int rollNo;
- **LOCAL VARIABLES :-** inside a block memory allocation during compile time.
  for eg:-
  {
  int c=10;
  }

LITERALS

literals is a constant assigned to variables

FOR EG:-

- integer literals/ integral
- floating point literals
- Boolean literals
- String literals
  (string s =”abc”;) , string str =new string(‘ABC’)

OPERATORS

operator is a symbol, it will perform a particular operation over the provided operands.

to prepare java application java gas provided the following list of operators:

1. AIRTHEMATIC OPERATIONS : (+ , - , / , *)

PROGRAM :-

```
//PROGRAM OF ADDITION
Class Addition
```

```
{
Public static void main()
    {
      int a=10;
      int b=20;
      int c=a+b;
     System.out.println(c)
     }
     }
```

2. Assignment operators :- (= , += , -= , /= , %=)
3. Comparison operators :- (== , not!= , <= , >=)
4. Logical operators :- (and, or, not)
5. Bit wise operators :- (left shift(<<) , right shift(>>))
6. Ternary operators :- (true false)
7. Increment operators :- (++)
8. Decrement operators :- (- -)

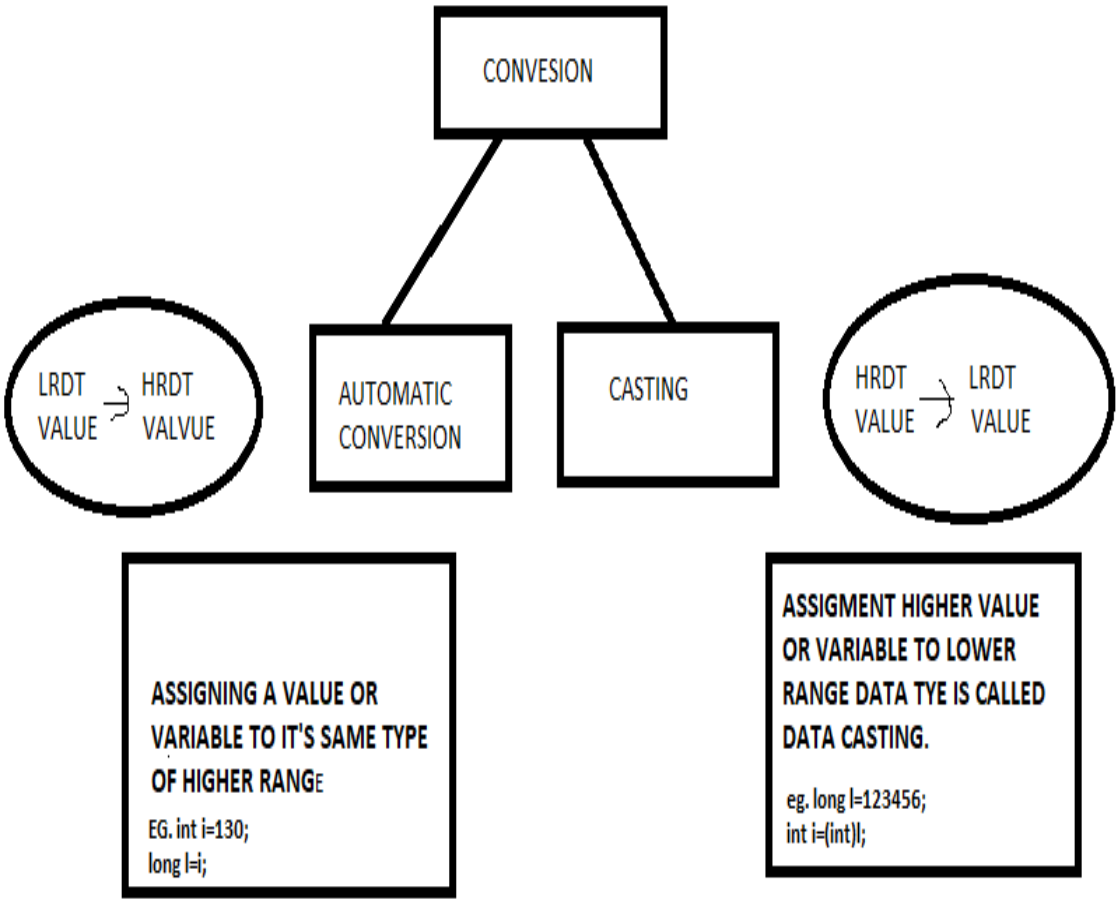
TYPES OF CONVERSION & CASTING

Type conversion is also called automatic or implicit or implicit widening.

Type casting is also called manual or explicit or narrowing.

Converting one type 0f value to another type of value is called conversion to develop type conversion we will use assignment operators.

we have two types of conversion.



Automatic/implicit/widening

---------->

byte short int long float Boolean double

<----------

Manual/explicit/narrowing/casting

JAVA STATEMENTS

statements are collection of expression to design java application java has provided following statement:-

1.**GENERAL PURPOSE STATEMENT :-**

eg :- creating object

2..**CONTROLL STATEMENT** :-

- conditional
- iterative
- transfer

3.**EXCEPTION HANDLING STATEMENT** :-

4.**SYNCHRONIZE STATEMENT** :-

controll statement:-

these statements are used to controll the flow of execution.

controll statement :-

- conditional
- iterative/looping
- transfer

1. **CONDITIONAL STATEMENT** :- these statements are able to allow execute a block at instruction under a particular condition.
   - if (CONDITION)

{

}

```
if(condition)
{
        True Stmt;
}
else()
{
    false stmt;
```

```
}
```

- ELSE IF

```
Eg :-  If()
        {
        }
         else if()
         {
         }
         else()
        {
         }
```

SWITCH:-

Switch blocks contains literals and constants we should pass literals and constant as byte short int char and string only.

```
Switch(var)
case 1:
statement1;
break;
case2:
statement2;
break;
case n:
statement n;
break;
default:
default statement;
break;
```

ITTERATIVE STATEMENT:-

These statements are able to allow jvm to execute a set of instruction repeatedly on the basis of a particular condition.

- while
- do while
- for

1. for loop:- for(Exp1;Exp2;Exp3;)
   {
   }
   In for loop expression 1 is optional we can write any statement like

   System.out.println (“HELLO”);
   expression1 but always it is suggestible to provide loop variable declaration and initialization kinds of statements as a part of expression1.

   In for loop expre ssion one is able to allow atmost one declarative statement it will not allow more than one declarative statement we can declare more than one variable into single declarative statement.

for(int i= 10; ; i++)

in for loop exp 2 is optional we can write for loop even without exp 2

if we write for loop without exp 2 then for loop will take “TRUE” value as expression 2.

WHILE LOOP:-

In java application when we are not aware the no, of iteration in advance before writing loop there we have to utilize while loop .

SYNTAX :-

While

{

}

Eg.

```
Class Ram

{

Public static void main(String[] args)

{

Int a=3;

While(a<=10)

}

System.out.println(a);

a++;

}
```

Difference b/w while & do while?

1. While loop is not giving any guarantee to execute loop body min '1' time to while loop will give guarantee to execute loop body atleast one time.
2. In case of while first conditional expression will be executed if it returns to then only loop body will be executed In case of do while first loop body will be executed then condition will be executed
3. In case of while condition will be executed for present iteration, in case of do while condition will be executed for next iteration .

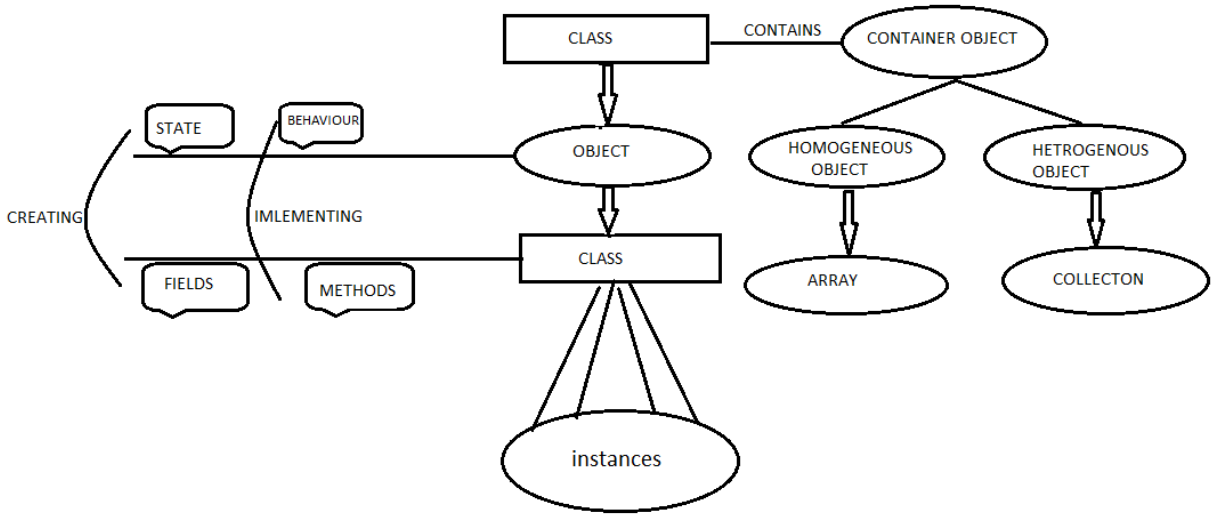
Transfer Control stmt:-

These elements are able to bypass flow of execution from one instruction to another instruction.

1.break

2.continue

3.return



Block diagram of OOPs

IN OOPS CONCEPT WE HAVE TWO TYPES OF OBJECT.

1. business object.(for eg. pen, laptop)
2. container object.(for eg. bus, train)

steps to represent real word object.

1. identify an object of particular type create it using class.
2. identify it's properties, create them data fields.
3. identify when these properties should initialize by providing constructor or setter ().
4. identify it's operations, implement them using methods.
5. using new keywords create instance for real word object of this object type .

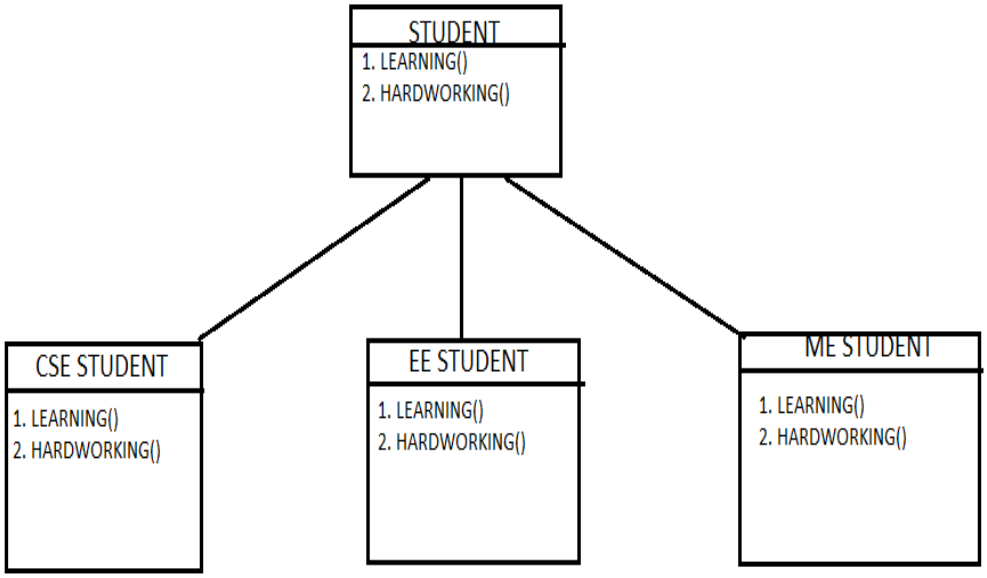
```
class student
{
Int rollNo;
Stringname;
Sting address;
Student (int rollNo, String name, String, String address)
{
This.rollNo =rollNo = rollNo;
This.name = name;
This.address = address;
}
void learning()
{
System.out.println(“I am learning”);
}
void hardworking()
{
System.out.println(“I am hardworker”);
public Static void main(String[] args)
{
Student std1 = new student (103, “yash”, “satna”);
}
std1.learning();
std1.hardworking();
Student std2 = new student (073, “sejal”, “satna”);
std2.learning();
std2.hardworking();
}
}
```

OBJECT:-

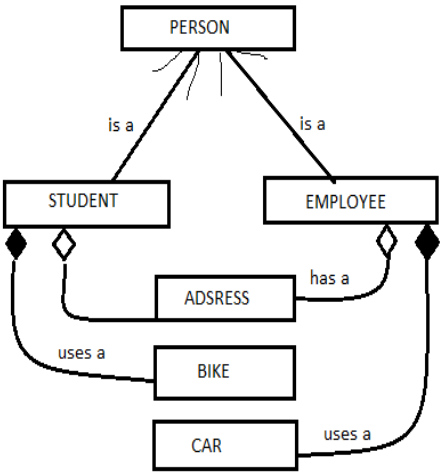
- object is a real word entity which has physical existence.
- object has it’s own identity, state(properties) & behaviour

OBJECT DESIGN MODEL:-

- Single class multiple classes or multiple instances



RELATION B/W OBJECTS:-



CLASS:-

The main purpose of the class is real word entities in java application

FOR EG:- Student, Account, Employee

- to represent entities data in java appl. we will use variables (data fields).
- to represent entities behaviour we will use method.

SYTX:-
Access
Execution Specifier
Class
Class Name

[ACC_SPC] [EXE _SPC] CLASS CLASS NAME [EXTENDS/IMPLEMENTS SUPER CLASS]

{

-------VARIABLE--------

-------METHODS--------

-------BLOCKS-----------

------CONSTRUCTO----

-------INNER CLASS-----

OOPs principles:-

- IN THIS TOPIC WE WILL LEARN ENCAPTULATION.
- INHERITANCE :– method overloading, method hiding method overriding, constructor overloading, constructor chaining, final class, final variable, final method.
- POLYMORPISM :-
  - compile time .
  - run time.

- ABSTRACTION :-
  - **ABSTRACT CLASS.**

- **INTERFACE.**
- **ABSTRACT METHODS.**

oops are the set of suggestion for designing and developing object for achieving dynamic method dispatching and security.

if we develop java application by implementing oops principle, the application will be come scalable means it will except or we will easily plug future inhancement without modifying.

future structure also we can modifying codes in one object without disturbing modifying other object code.

we have three oops concepts.

1. ENCAPTULATION.
2. INHERITANCE.
3. POLYMORPHISM.

NOTE :- abstraction is not oops principle , its object modeling principle.

abstraction will be used in object designing phase for axciding the operation performed by the object.

these principles are not providing any api (application programming interface).

by following these principles we must develop class to achieve benefits that is auto pluging(dynamic metod dispatching).

DYNAMIC METHOD DISPATCHING:-

it means executing already invoked method from different sub classes dynamically based on the object stored.

dynamic method dispatching is also called run time polymorphism.

ENCAPSULATION:-

encapsulation is the process of hiding data from direct access from external world, providing it's access by setter() or getter() with proper validation.

WRAPPING UP:- wrapping up means providing access to variable by methods.

IMPLEMENTATION OF ENCAPSULATION:-

- declare all variables inside a class as "private" so that other class programmer can't access directly.
- define a pair of setter() or getter() method for each private variables with validation (if needed for storing and reading values.
  project—file/new/javaproject/project name/poject1/next/finish.
  project1
  1 jre(1.8)
  1 src(source)

PACKAGE:- right click on src/new/package name/java.encapsulation

1 –> java.encapsulation

CLASS :- right click on java.encapsulation/new/class name/ bike/ next/finish/

1 bike.java

1. for program file/new/java project/project name/ the double click on poject name/src/click on src/new/class/class name/right click on project name and create main class.
2. object memory creation with non static variable (instance) with default value.
3. object initialization with given values.
4. string the reference of object in reference variable.

NEW VARIABLE AND CONSTRUCTORS RESPONSIBILTY :- new keyword is responsible for creating objects memory, storing, default value then invoking constructor for initialization.

DISCRIPTION OF ABOVE PROGRAM :-

- in bike class we develop encapsulation by declaring variable as private and by providing setter and getter method.

- in setter we have return written if else condition for checking given value right or wrong.
- if correct we stored given value inside non-static variable(gear).

```
//Bike.java
Public class bike()
{
Private int gear;
Public int get gear ()
{
Return gear;
}
Public void set gear(int gear)
{
If (gear<0 || gear >5)
{
Throw new illegalargument exception(“gear no should b/w 0-5”);
}
This.gear=gear;
}
}

//customer.java
Import.java.util.scanner:
Public class customer()
{
Public static void main(string[] args)
{
Scanner sc = new scanner(System.in);
Bike.tvs=new Bike;
While(true)
{
System.out.println(“change gear”);
Int gear =sc.nextint();
Sc.nextlne();
Try{
Tvs.set gear(gear);
If(gear==0)
```

```
{
System.out.println(“Bike is neutal”);
}
Else
{
System.out,println(“Bike is running”);
}
}
Catchillegal argument exception e)
{
System.out.println(e.getmessage());
}
}
}
```

the exception thrown from bike class will be from there es passed to this is customer class type from there is passed to catch block ,it is stored in catch block parameter

inside catch we call predefine method getmessege() for reading and displaying message that is passing from bike class.

OBJECT CREATION PROCESS :-

the object creation stmt like

```
bike pulsar = new bike( );
```

has four steps :-

referenced variable memory creation

if given value is wrong we are throughing exception using through keyboard so that we are terminating method execution informing the error to method caller.

so that method caller will send correct input.

METHOD IN JAVA

method is a set of instruction which represents object operation/action.

eg:- [access specifier] [execution specifier] return_type method name (parameter) [throw exception list]

```
public static void main sleeping( )

{

system.out.println ("I AM SLEEPING");

}
```

java methods are able to allow access modifier like public , protected private or default

protected(default) and private

java methods are able allow execution specifier like static, final, abstract, native , synchronize, strict

purpose of return type is to specify which type of data present method is returning.

method name is identifier it can be used to recognize the method individually.

parameter list can be used to pass some input data to the method in order to perform and action.

throws keyword can be used to bypass deligate the generated the exception from present method to caller method.

to describe method information there are two approaches.

1. method signature

2. method prototype

QST :- WHAT IS DIFFERENCE B/W METHOD SIGNATURE AND METHOD PROTOTYPE.

ANS :- method signature is a description of method , it includes method name and parameter list.

EG :- forName( )

forName(string class name)

method prototype is description of method it will include access modifier, execution level specifier, return type modifier, parameter list, throw exception list

EG :- PUBLIC STATIC CLASS FORNAME(STRING CLASS_NAME)

throws class not found exception

there are two types of methods in java with respect to object state manipulation

1. **mutator method.**
2. **accessor method.**

QST :- **WHAT IS DIFFERENCE B/W MUTATOR METHOD AND ACCESSOR METHOD.**

mutator methods are used to set or modify data in object.

eg :- all get xxx( )

CONSTRUCTOR:-

constructor is a java features it can be used at the time of object creation “the roll of constructor in object creation is to provide initial values inside the object”. in java applications constructors are executed exactly ate the time of creating objects , not before creating object and not after creating objects.

the main utilization of constructors is to provide initializations for class level variables (instance).

- constructors must have same name of the respective class.
- constructors are not having return type.
- constructor not allow execution modifier like static, final, etc.
- constructors able to allow access modifier like public, protected, <default> and private.

- constructors are allows throws keyword n its syntax to bypass exception from present constructor to the caller.

sytx:-[access specifier] class name ([parameters list]) [throws execption list]

{

//INSTRUCTIONS;

}

FOR EG : public Student (int rollNo, String name, String address)

{

this.rollNo= rollNo;

this.name=name;

this.address=address;

}

NOTE 1:- IF WE PROVIDE CONSTRUCTOR NAME OTHER THAN CLASS NAME THEN THE COMPILER WLL RISE AN ERROR LIKE"INVALID METHOD DECLARATION ,RETURN TYPE IS REQUIRED" BECAUSE COMPILER HAS TREATED THE PROVIDED CONSTRUCTOR AS NORMAL JAVA METHOD WITHOUT RETURN TYPE.

NOTE 2:- IF WE PROVIDE RETURN TYPE WITH CONSTRUCTOR THEN THE COMPILER WILL NOT RISE ANY ERROR AND JVM WILL RISE ANY EXCEPTION.

Note 3:-if we provide execution level modifier like static, final etc. Then compiler will rise an error like"modifier xxx"not allowed here.

:-

there are two types of constructor in java.

1. default
2. user define
3.

1. **DEFAULT CONSTRUCTOR**:- if we have not provided any constructors explicitly in java class then compiler will provide a zero arg constructors automatically, here the compiler will provided zero arg constructor is called as default constructor.

2. **USER DEFINE CONSTRUCTORS**:- these constructors are provided by developers as per their application required if we provide any constructor explicity without any parameters. then that constructor is called zero arg constructor.
   if we provide any constructor with atleast one parameter. then hat constructor is called parameterized constructors.

SETTER METHOD:-

a method who's name is starts with "set" following by field name(variable) is called setter method.

setter method is also called sett xxx method or mutator method. it is a void, parameterized method.

parameter type and name should be same as field and its name tat is initializing in the method. in this method logic should be assigning parameters to the field

QST :- DIFFERENCE B/W CONSTRUCTOR AND SETTER METHOD IN INITIALIZING OBJECT.

- **CONSTRUCTOR :-** initializes object at the time of object creation.
- **SETTER :-** setter method initializes object after object creation.

OBJECT INITIALIZATION FLOW :-

CONSTRUCTOR :-

- object memory created with default values.
- initialization with given values.
- finally reference is stored in reference variable.

SETTER :-

- object memory created with default values.
- referenced stored in reference variable than after .
- then after object is initialized with given value when we call setter method.

CONSTRUCTOR :-

- constructor is called by new keyword in object creates n element.

SETTER :-

- setter method is explicity called by programmer with reference variable.

CONSTRUCTOR :-

- we can't initialize variable of object individual from different constructors**.**

SETTER :-

- it's possible to initialize same objects different variable from different setter ( ) using same reference variable.

STATIC MEMBER EEXCUTION FLOW :-

when a class is loaded into a jvm. jvm execute static member from this class in the given order.

- ❖ it's provides memory to all static variable with default value based on their data type is store assigned values in static variable execute all the static block in the order they are accelerate from top to bottom.
  finally main method is executed
  static method is executed only if it is called from any one of the above three member.

  memory allocation class static member execution

NON STATIC MEMBER EEXCUTION FLOW :-

when an object is created jvm will execute non static member give class in following order

- ❖ non static variable ate provided memory with default value based on their data type.
- ❖ all non static variable are initialized or non static blocks logic executed in the order they can declared from top to bottom.
- ❖ finally the involved constructor is executed only if it is called from any one of the above member.

  memory allocation non static member execution

we can’t execute a class just with non static member,m we must have atleast one static member.

STATIC KEYWORDS

static is a java keyword it will improove sharebility in java application static keyword will be utlised in following based.

STATIC VARIABLE :-

- static variables are normal java variables which will be recognized and executed exactly at the time of loading the respective class byte code to the memory.
- static variables are shared there last modified values to the future objects and to the past object of respective class .
- in java application static variable will be accessed ether by using the respective class reference variable or by using the respective class name directory.

STATIC METHODS:-

- static method is a normal java method, it will be recognised and executed and the time when we access that method.
- static metyhod will be accessed either by using refference variable orby using respective class name.

STATIC BLOCK

static block is a set of instuction which will be recognised and executed at the time of loading the respective class byte code to the memory.

static block are able to allow static member of current class directly, static blocks are not allowing not static member of the current class.

QST:- DIFFERENCE B/W STATIC BLOCK AND STATIC METHOD.

ANS :-

STATIC:-

- **IT DOESN'T HAVE NAMES ONLY CONTAINS KEYWORD(static).**
- **IT WILL HAVE RETURN TYPE AND NAME.**

- **WE CAN'T CALL STATIC BLOCK.**
- **WE CAN CALL STATIC BLOCK.**

- **IT IS AUTOMATICALLY EXECUTED BY JVM WHEN CLASS IS INTO JVM.**
- **PROGRAMMER SHOULD CALL STATIC METHOD FOR EXECUTION.**

- **AS PER PROJECT STATIC BLOCK CONTAIN INITIALISATION LOGIC.**
- **STATIC METHOD CONTAINS EITHER INITIALISATION LOGIC OR BUSINESS LOGIC.**

- **IT'S EXECUTED ONLY ONCES IN A CLASS LIFE TIME BECAUSE CLASS IS LOADED INTO JVM ONLY ONCE.**
- **IT HAS CHANCE TO EXECUTE MORE THAN ONCE AS MANY TIME AS WE CALL.**

- **INSIDE A STATIC BLOCK WE CAN'T PLACE RETURN .**
- **IT ALLOW TO PLACE RETURN STATEMENT.**

INSTANCE CONTEXT OR MEMBER:-

in java, for every class loading a separate context will be created called as static context. and for every object a separate context will be created called as instance context.

in java instance context is represented in the form of following element.

INSTANCE VARIABLE :-

- Instance variable is a normal java variable who's value will be varried from one instance will to another instance of an object
- Instance variable is a variable which will be recognized and initialized just before executing the respective class constructor.
- In java application instance variable must be declared in class level and new we declared as local variable and static variable.

INSTANCE METHODS:-

- Instance method is a normal java method it is a set of instructions, it will represent an action of an entity.
- In java applications instance , methods will be executed when we access that method.

INSTANCE BLOCK :-

- Instance block is a set of instructions which will be recognized and executed just before executing the respective class constructor.
- Instance block having same power of constructors, it can be used as like constructors.

INHERITANCE :-

It is a process of creating new class deriving from an existing class "for extending and reusing functionality of an existing object".

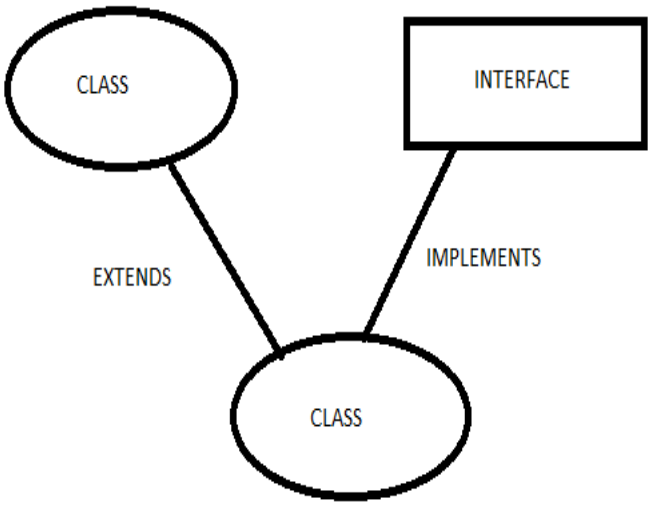
Extending and reusing functionality of an object is called an inheritance.

NOTE:- INTERNALLY OUR CLASS IS DERIVED OR EXTENDED FROM JAVA INBUILT CLASS JAVA.LANG.OBJECT .

When we extend a class from existing class all non private properties inharited to super class for reusing them into sub class.

Inheriting means we will get permission for accessing the existing class non private members from new class without creating existing class object.

Using extends or implement keyword we will develop inheritance in java inheritance is also called is-a relation



EXTENDS:-

extend keywords represent extending or reusing personality of existing class.

IMPLEMETS:-

implements keywords represents implementing or providing logic to existing class.

```
public class SuperClass {
     static int a=10;
              int b=20;
              static void m1()
              {
```

```
                System.out.println("m1 from SuperClass");
        }
        void m2()
        {
                System.out.println("m2 from SubClass");
        }

}
```

```
public class SubClass extends SuperClass {
static void m3()
{
      System.out.println("hello");
      System.out.println(a);
      m1();
      //System.out.println(sub.b);
      //m2();
}
void m4()
{
      System.out.println("hi");
      System.out.println(a);
      m1();
      System.out.println(b);
      m2();
}
public static void main(String[] args) {
      m3();

      SubClass sub=new SubClass();
      sub.m4();
}
}
```

Super class static variable of static method can be accessed directly from subclass but their name from both static member & non static member

Nsv (non static variable) and non static method of we can access only in non static member of sub class.

If we want access non static variable and non static method of super class in static member of sub class then we must create object of super class

ADVANTAGES OF INHERITANCE:-

- Reusebility.
- Overriding or implementing.
- Adding more functionality.

In designing object we will develop three types of classes.

1. Super class.
2. Sub class.
3. User class.

In above diagram we have two concepts.

1. Loose coupling.
2. Lcrp(loose coupling run time polymorphism).
    1. Loose coupling :- loose coupling means storing different sub class objects of same type in a single variable, this approach will give you a advantage of run time polymorphism(dynamic method dispatching).

       Run time polymorphism means calling method only once and executing it's logic from different sub classes of same type who's object is stored in reference variable.

POLYMORPHISM:-

Polymorphism is the process of providing different behaviour to the same object. Polymorphism means providing different implementation to the same method.

TYPES OF POLYMORPHISM :-

1. COMPILE TIME

2. RUN TIME.

METHOD OVERLOADING, HIDING, OVERLOADING

- Providing new implementation to super class <u>static method</u> in subclass is called method hiding.
- Providing new implementation to super class to <u>non static method</u> in sub class is called method overriding.
- Defining multiple static or non static method with same name but with different parameters is called method overloading.
-

Note :- we can't override or hide method in same class, it is treated as duplicate method & compiles wire will throw error.

We can only hide or override methods in subclass but we can overload method either in same class or it's subclass.

THIS KEYWORD :-

This is a java keyword it can be used to represent current class object in java application we are able to utilize this keyword four ways

1. **TO REFFER CURRENT CLASS VARIABLE.(this.variable name)**
2. **TO REFFER CURRENT CLASS METHOD. (this.method name)**

```
public class ThisMethod {
 void m1()
 {
       System.out.println("this method");
 }
 void m2()
 {
      this.m1();
 }
 public static void main(String[] args) {
      ThisMethod tm=new ThisMethod();
      tm.m2();
}
}
```

3. **TO REFFER CURENT CLASS CONSTRUCTORS.**

```
package Constructor;

public class Constructor1 {
      public Constructor1() {
            this(10);
            System.out.println("0 ag constructor");
}

            public Constructor1(int i) {
                  this(20.0f);
                  System.out.println("int arg constructor ");
      }
            public Constructor1(float f)
            {
             System.out.println("float arg constructor");
            }

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

```
            Constructor1 cn= new Constructor1();

      }

}
```

4. TO RETURN CURENT CLASS OBJECT.

```
package Constructor;

public class A {
      A getRef1()
      {
            A a=new A();
            return a;
      }
      A getRef2()
      {
            A a=new A();
            return this;

}
    public static void main(String[] args) {
      A a=new A();
      System.out.println(a);
      System.out.println(a.getRef1());
    System.out.println(a.getRef2());
    }
}
```

o/p =

```
          Constructor.A@5a07e868
          Constructor.A@76ed5528
          Constructor.A@5a07e868
```

SUPER KEYWORD:-

Like this keyword, super is also a keyword.
"An implicit non static final reffernced varible".

- Super keyword type is current class superclass.
- It is use for accessing super class variables, methods and constuctors.
- Like this keywors super keyword has also two forms.

TYPES OF CLASSES:-

THERE ARE TWO TYPES OF CLASSES:-

1. **USER DEFINED.**
2. **PRE DEFINED.**

1. **USER DEFINED :-**
   - **CONCRETE CLASS.**
   - **ABSTRACT CLASS.**
   - **INTERFACE CLASS**
   - **FINAL CLASS.**

- **CONCRETE CLASS :-** Concrete class is a fully implemented class. It contains concrete method. To declare concrete methods
  FOR EG :- //Student.java

- **ABSTRACT CLASS :-**Abstract class is a partially implemented class. Abstract class is declared by using abstract keyword and if any class contains abstract method then we should declare that class as abstract class.
  FOR EG:- **//A.java**

```
package abstractClass;

public abstract class A {
	abstract void m1();
	abstract void m2();
	void m3()
	{
```

```
            System.out.println("hello");
      }

}
```

//B.java

```
package AbstractClass;

public class B extends A {

      @Override
      void m1() {
            // TODO Auto-generated method stub
            System.out.println("abstract m1");
      }

      @Override
      void m2() {
            // TODO Auto-generated method stub
            System.out.println("abstract m2");
      }
public static void main(String[] args) {
      A a=new B();
      a.m1();
      a.m2();
      a.m3();
}
}
```

INTERFACE:-

Interface is a fully unimplemented class.interface contains abstract method to declare interface we will use interface keyword

FINAL CLASS :-

Final class is a class which is not allow inheritance in java application super classes never be declared as final class but sub classes may be final.
FOR EG :-

```
 final class a
{
}
class B extends A
{
}
```

o/p :- invalid

DIFFERENCE B/W CLASS AND OBJECT:-

- **CLASS IS A GROUP OF ELEMENT HAVING COMMON PROPERTIES AND BEHAVIOURS.**
- **OBJECT IS AN INDIVISUAL ELEMENT AMONG THE GROUP OF ELEMENTS HAVING PHYSICAL BEHAVIOUR AND PROPERTIES.**

- **CLASS IS VIRTUAL.**
- **OBJECT IS REAL.**

- **CLASS IS VIRTUAL ENCAPSULATION OF PROPERTIES AND BEHAVIOUR.**
- **OBJECT IS PHYSICAL ENCAPSULATION OF PROPERTIES AND BEHAVIOUR.**

- **CLASS IS GENEALISATION.**
- **OBJECT IS INITIALISATION.**

- **CLASS S MODEL OR BLUPRINT FOR THE OBJECT.**
- **OBJECT IS AN INSTANCE OF CLASS.**

WHAT ARE THE DIFFERENCE B/W CONCRETE METHOD ABSTRACT METHOD?

- Concrete method is a method it will have both method declaration & method implementation.
  There is no special keyword to declare a method as concrete .

  Syntax :- return method name([parameter])
  {
  //IMPLEMENTATION
  }

  VOID ADD(INT I , INT J)
  {
  INT K =I+J
  }

- Abstract method is a method it will have only method declaration . To declare abstract method we must use keyword.
  SYNTAX :- abstract return type method name([parameter])
  Eg:- abstract void add (int I, int j);

  NOTE :-
  1. Concrete methods are allowed in classes & in abstract classes but abstract methods are allowed in abstract classes and interfaces.
  2. Concrete methods will provide less sharebility but abstract method will provide more sharebility.

WHAT ARE THE DIFFERENCE B/W COCRETE CLASSES AND ABSTRACT CLASSES?

- Concrete classes are able to allow concrete methods but abstract classes able to allow concrete method and abstract method.
- To declare concrete class only “class” keyword will be sufficient but in abstract class to declare we will be use “abstract” keyword along with class keyword.
- For concrete class we are able to create both reference variables and object but for abstract class we are able to create only reference variable.
- Concrete class are less shareable but abstract class are more shareable.

WHAT ARE THE DIFFERENCE B/W ABSTRACT CLASS AND INTERFACE ?

INNER CLASSES OR NESTED CLASS:-

Declaring a class inside a class is called inner class.

```
SYNTX :- CLASS  OUTERCLASS
         {
          CLASS INNER CLASS
               {
```

```
                    }
        }
```

In java application inner class are able to provide following advantages.

1. MODULARITY.
2. ABSTRACTION.
3. SECURITY.
4. SHAREBILITY.
5. REUSEBILITY.

1. **MODULARIRY :-** in java application we are able to improve modularity by using package ,if we want to divide our implementation in a class in the form of modules then we have to use inner classes.
   FOR EG :-

```
class Account
{
Class SavingAccount
{
Void saving(){
}
}
Class CurrentAccount
{
Void current(){
}
}
```

2. **Abstraction :-** if we declare any variable or method or in an inner class then that variable and method are having scope up to that class only, it's not available to other inner class so that inner classes are improve abstraction.

```
class Account
```

```
{
Class SavingAccount
{
Int amt=100;
}
Class CurrentAccount
{
Int bal=10+amt;
}
```

3. **SECURITY:-** it's not possible to declare a class private but it is possible to declare an inner class as private so that inner classes are able to improve security.

```
class Account
{
Private Class SavingAccount
{
}
Private Class CurrentAccount
{
}
```

4. **SHAREBILITY :-** it's not possible to declare a class as static but it's possible to declare an inner class as static so that inner class are able to improve sharebility.

```
class Account
{
Static Class SavingAccount
{
}
static Class CurrentAccount
{
}
```

5. **REUSEBILITY :-** in java application we can extend our inner class to another inner class so that inner classes are able to improve code reusability.

```
class Account
{
 Class SavingAccount
{
}
Class zero balsaving extends SavingAccount
{
}
}
```

There are 4 types of inner classes in java.

1. **member inner class.**
2. **Static inner class.**
3. **Method local inner class.**
4. **Anonymous inner class.**

1. **MEMBER INNER CLASS :-** declaring a normal class [non static class] inside a class is called as member inner class.

SYNTX :- CLASS OUTER_CLASS

```
{
CLASS INNER_CLASS
{
}
}
```

OUTER.INNER REFERANCCE VARIABLE = NEW OUTER.NEW INNER

Note :-

1. **by using outer class reference variable we can access only outer class member.**
2. **By using inner class reference variable we can access only inner class member.**
3. **By default all outer class member are available in inner class out all inner class member are not available in outer class.**

```
public class OuterClass {
     void m1()
     {
      System.out.println("i am m1");
     }
     class InnerClass
     {
      void m2()
      {
            m1();
            System.out.println("i am m2");
      }
      void m3()
      {
            System.out.println("i am m3");
      }
     }
     public class InnerClass1 extends InnerClass {
      void m4()
      {
            m1();
            System.out.println("i am m2");
      }
      void m5()
      {
            System.out.println("i am m3");
      }

    }

     public static void main(String[] args) {

           OuterClass.InnerClass o=new OuterClass().new InnerClass();
           o.m2();
           o.m3();
           OuterClass.InnerClass1 o1=new OuterClass().new
InnerClass1();
           o1.m4();
           o1.m5();

     }
}
```

2. **STATIC INNER CLASS :-** Declaring a static class inside a class is called as static inner class.

   SYTNX :-

   CLASS OUTER CLASS
   {
   STATIC CLASS INNE
   {
   }
   }

   OUTER.INNER REFERANCCE VARIABLE= NEW OUTER.INNER

```
public class Outer1 {
	static int a=10;
		   int b=20;
	static class Inner
	{
		void m1()
		{
			System.out.println(a);

			System.out.println("hello i am m1");
		}
		void m2()
		{
			System.out.println("hello i am m2");
		}
		}

	public static void main(String[] args) {
	 Outer1.Inner a=new Outer1.Inner();
	 a.m1();
	 a.m2();
	}

}
```

All static member of outer class are available in but we can't access not static member of outer class in inner class.

3. **METHOD LOCAL INNER CLASS:-** declaring a class inside a method is called method local inner class.

```
//METHOD LOCAL INNER CLASS
public class Outer2 {
	void m1()
	{
	class local
	{
		void m2()
		{
			System.out.println("hello i am m2");
		}
	  }
	    local l=new local();
			l.m2();
	}

	public static void main(String[] args) {
	Outer2 o2=new Outer2();
	o2.m1();


	}

}
```

If we declare a class inside a method then the scope of that class is available upto the method respective only, we have to create object for that inner class in the respective method only.

QST :- IS IT POSSIBLE TO WRITE AN INTERFACE INSIDE A CLASS
ANS :- YES IT'S POSSIBLE TO PROVIDE AND INTERFACE INSIDE A CLASS BUT THE RESPECTIVE IMPLEMENTATION CLASS MUST BE PROVIDED WITH IN THE SAME OUTER LOOP
QUS :- IS IT POSSIBLE TO WRITE A CLASS INSIDE AN INTERFACE.
ANS :- YES IT'S POSSIBLE TO WRITE A CLASS INSIDE AN INTERFACE, IF WE DECLARE A CLASS INSIDE AN INTERFACE THEN THAT CLASS IS CONVERTED AS A STATIC

INNER CLASS, WE CAN ACCESS MEMBER OF THIS INNER CLASS LIKE STATIC INNER CLASS MEMBER.

4. **ANONYMOUS INNERCLASS :-** anonymous inner class is nameless inner class, these are used to provide implementation for abstract class and interface .

   SYTX:- abstract class name/interface name

   {

   }

   Class Outer class

   {

   Name ref.varb =new name()

   {

   //implementation

   }

   }

   In java if we want to provide implementation for only abstract class members or interface members and if we want to create objects with interface identify with abstract class identify then we have to use anonymous inner class

Wrapper class

In java applications collection objects are able to store only objects they will not store primitive data type if we want to store primitive data in collection object then we should use Wrapper class.

Wrapper classes are used to convert primitive data type into object type.

Java has provided following Wrapper class with respect to primitive data type.

Conversion from primitive type to object type

1. By using parameterized constructors from Wrapper class

Public XXX(XXX value)

Eg:- int a= 10;

Integer i= new Integer(a);

2. By using valueOf method override by Wrapper class.

Eg: int a=20;

Integer i= Integer valueOf(a);

3. By using Auto-Boxing*(mostly used)

Auto-Boxing approach was provided by JDK 5.0 version in this approach no need to use predefine methods and constructors ,simply we have to assign primitive variable to wrapper class reference variable.

Eg:- int a=20;
Integer i= a;

4. Conversions from object type to primitive type.

1) By using xxxvalue () {method from Wrapper class}

Eg:- Integer i=new Integer (10);

Int a = a.i.intvalue();

2) By using Auto-unboxing.

Eg:-Integer i= new Integer (10);
Int a=i;

5. **Conversion from string type to object type.**
   1) By using string parameterized constructor from Wrapper class.
      Eg:- string s= "10"
      Integer i= new Integer(s);

   2) By using string valueOf () method
      Eg:- string s= "10";
      Integer i= Integer.valueOf(s);

6..CONVERTION FROM OBJECT STRING TYPE :-

1. BY USING toString ()
   EG:- Interger in = new Integer(10);
   String s= new in.toString();

2. By using '+' concatenation operator.
   Eg:- Integer in= new Integer(10);
   String s= " " +in;

6. **Conversion from String type to primitive Type.**

By using parse xxx() from Wrapper class

Eg:- String s= "10";

int i = Integer.parseInt(s);

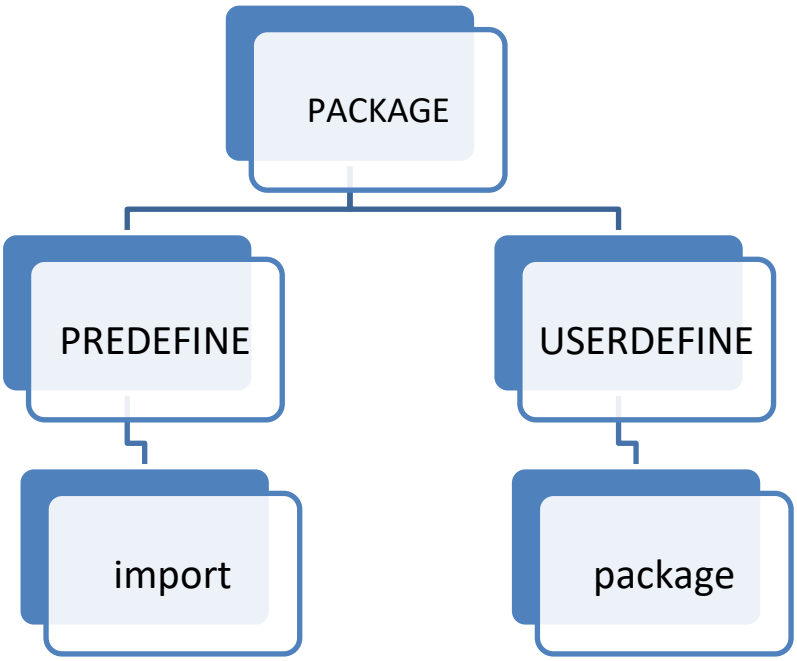
{mostly used in Command Line Argument}

PACKAGES:-

Package is the collection of related classes & interfaces as a single unitpackage is a folder contain .class file representing related classes and interface.in java application are able to provide following advantages :-

1. MODULARITY.
2. ABSTRACTION.
3. SECURITY.
4. SHARABILITY.
5. REUSEBILITY.

THERE ARE TWO TYPES OF PACKAGES:-



1. **PREDEFINE PACKAGE :-** these packages are provided by java programming language along with java s/w.

1. **Java.lang package:-** this package is default package in java application, no need to import this package to the present java file, when we save java file with .java extension then automatically java.lang package is imported internally.

Package is able to provide all fundamental classes and interfaces which are required to prepare basic java application these classes and interfaces are

- **Byte , Short , Int,Long . . .**
- **String , StringBuffer , StingBuilder . . .**
- **Object , System , Class . . .**
- **Exception , AirthmeticException , NullPointerException . . .**
- **Thread , Runnable , Clonable , Comparable . . .**

2. **Java.io package :-** this package is able to provide predefine classes and interfaces in order to perform i/o operations in java.

- **InputStream , ByteArrayInputStream, FileInputStream . . .**
- **OutputStream , ByteArrayOutputSteam , FileOutputStream . . .**
- **Reader, CharReader , FileReader . . .**
- **Writter , CharWritter , FileWritter . . .**
- **Serializable . . .**

3. **Java.util package (Related to System Utility) :-**

- **Collection , List , Set , Queue . . .**
- **ArrayList , Vector , Stack , LinkedList . . .**
- **HashSet , LinkedHashSet , ShortedSet , NavigableSet , TreeSet. . .**
- **Queue , PriorityQueue . . .**
- **Map , HashMap , LInkedHashMap . . .**

4. **Java.awt (abstract window tools) :- GUI (grafical user interface)**
    - **Component , Label , Textfield , TextArea , Button , CheckBox , List , Choice , Frame . . .**

5. **Javax.swing :-** this package is able to provide predefine libraries to prepare gui application.
    - ❖ JComponenet , JTextField , JPasswordFields , JButton , JCheckBox . . .

QUST :- WHAT AE DIFFERENCE B/W AWT(java.awt) & SWING (javax.swing) ?

ANS :-

1. Awt components are platform dependent.

1. Swing components are independent.

2. Awt components are heavy weight gui components.

2. Swing components are light weight components .

3. Awt components reduse the application performance.

3.swing components improves application performance.

6. java.net :- If we prepare any java application without using client server architecture then that java application is called as stand alone application.

If we prepare java application on the basis of client server architecture then that java application is called as distributed application.

To prepare distributed application java has provided following distributed technologies.

1. Socket programming.
2. Remote method invocation (rmi).
3. Corba(common object request broker architecture).
4. Ejb (enterprise java beans).
5. Web service.

Java.net PACKAGE PROVIDES FOLOOWING CLASSES:-

- ❖ Socket.
- ❖ Server socket.
- ❖ URL , URI , URN.
- ❖ URlConnection.

7.java.rmi :-

- **Remote.**
- **RemoteException.**
- **Naming.**

8.java.sql :- THE PROCESS of intacting with database from java application is called as JDBC

- Driver , DriverManager , Connection , Statement , PreparedStatement , ResultSet . . .

2. **USER DEFINE PACKAGES :-** these packages are define by developers as per there application requirnment.
   SYNTAX :- package package_name

ARRAY:-

- array is indexed based fixed number of multiple values object with similar types.
- array is collection of multiple values of same type.
- array is a container object ,it can contain multiple value.

FOR EG:- int arr[] = new int[5];

IN JAVA ARRAY IS REFFERENCED TYPE

- To represent array object we must be use special notation [ ].
  For eg :- if we want to store integer value we must create int variable
  Int a =5;

  If we want to store multiple integer values then we must use int [ ]
- Array object is created with no of locations = the number of values we specify.
- Each location is created with an index , starts with 0 .
- For reading values from array object and modifying values in array object we must use array index.

  DECLARATION:-
  SYNTAX :- DT arrayName = new DT[size];

INITIALISATION :-

SYNTAX :- DT arrayName[]= {values};

Or

arrayName[indexed]=value;

ADNAVTAGES OF ARRAY :-we can store multiple values on one group, we can pass and return all values from one method to another method.

Limitatiom of array :- it's fixed in length after array creation we can't change its length.

Length property :- property means non static variable of an object. Length is a final instance variable meant for storing length of an array object i.e. Contain a integer number which represents number of values stored in array object.

Cloning :- we can create duplicate array by using cloning to create duplicate array we will use clone method.

```
package simplearray;

public class SAEClone {
	public static void main(String[] args) {
		int arr[] = {10,20,30,40,50};
		int clone[]=new int[5];
		clone=arr.clone();
		for(int i=0;i<clone.length;i++) {
			System.out.println(clone[i]);
		}

	}

}
```

ARRAY OF OBJECT :- we can create object same as array of primitive data type

```java
package ArrayOfObject;

public class Student {

	int rollNo;
	String name;
	public Student(int rollNo, String name)
	{
		super();
		this.rollNo=rollNo;
		this.name=name;
	}

}
```

```java
package ArrayOfObject;

public class Test {

	public static void main(String[] args) {
		Student st[]=new Student[6];
		st[0]=new Student (004,"CHOTI BACCHI HO KYA!!");
		st[1]=new Student (026,"deshuu baby");
		st[2]=new Student (039,"kammoooo darling");
		st[3]=new Student (073,"sejjuuuuuu");
		st[4]=new Student (077,"sluutiiiiii");
		st[5]=new Student (103,"MAI GAREEB HU!!!");



		for(int i=0;i<st.length;i++)
		{
			System.out.println(st[i].rollNo+" "+st[i].name);
		}

	}

}
```

PASSING ARRAY AS ARGUMENT:-

TAKING INPUT FROM USER :-

```
import java.util.Scanner;

public class Average {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("enter the vaue of elements");
		int n=sc.nextInt();
		int arr[] = new int[n];
		System.out.println("enter "+ n +" elements in array");
		int sum=0;
		for(int i=0;i<n;i++)
		{
			arr[i]=sc.nextInt();
			sum=sum+arr[i];
		}

		System.out.print(sum/n);

	}

}
```

JAGGED ARRAY :-

A jagged array is array such that no arrays can be different size that is. We can create a 2d array but with a variable no of columns in each row these types of arrays are alspo known as jagged array.

DECLARTION :-

DATA TYPE array_name [] [] = new DATA TYPE[row size] [ ] ;

Array_name[row index] = new DT [col size];

Eg :- int arr[ ] [ ] = new int [3] [ ];

Arr[0] = new int[2];

Arr[1] = new int[3];

Arr[2] = new int[1];

Program :-

```
public class Jagged {
	public static void main(String[] args) {
		int arr[][] = new int[3][];
		   arr[0]=new int[1];
		   arr[1]=new int[2];
		   arr[2]=new int[3];
		   int c=1;
		   for(int i=0;i<arr.length;i++)
		   {
		      for(int j=0;j<arr[i].length;j++)
		      {
		            arr[i][j]=c++;

		      }

		   }
		   for(int i=0;i<arr.length;i++)
		   {
		      for(int j=0;j<arr[i].length;j++)
		      {
		            System.out.print(arr[i][j]+" ");

		      }
		      System.out.println();

		   }

	}

}
```

STRING MANUPULATION :-

String is a sequence of characters those are placed in double quotes.

the sequence of character those are placed in double quote " " are consider as instance or object of a string class.

STANDARD DEFINITION :- string is a final class given for representing sequence of characterd to stored in jvm fom java program in easy way.

String class internally maintain a character array object for storing given character instance this char [ ] declared as private and final.

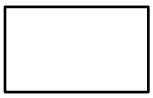
Eg:-string s= new String(“abc”);

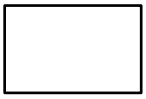
Final class String

{

Private final char[] ch;

}





STRING HANDLING :-

Performing different operation on string data is called string handling .

The operation such as : concatenating , replacing character , changing character case , retrieving a character or substring , comparing two string , etc.

To perform these operation we have no need to develop logic, string class have several methods to perform these operations.

All we need to do is, create an object from string class, store data in that object, call appropriate method from string class.

1. **Equals method :-** equals methods compare two string object with their content by considering character’s case and returns tree only if character and their cases are same other wise return false.

2. **EqualsIgnoreCase:-** equal ignore case method compares string by their content it returns true if character's are same other wise return false. it not consider character case.

OUS 1 : WHAT IS DIFFERENCE B/W FOLLOWING TWO STATEMENT
String = "ABC" OR String s = new String.

To create string object if we use first statement then string object will be created at a string constant pool area only in method area .
If we use second statement then one string object will be created at heap memory as per new keyword and another string object will be created at string constant area inside method area.

QUS : IN HOW MANY WAYS WE CAN STORE STRING DATA IN JVM.
ANS : THERE ARE FOUR WAYS :-

1. **CHAR[] OBJECT.**
2. **STRING CLASS OBJECT.**
3. **STRING BUFFER CLASS OBJECT.**
4. **STRING BUILDER CLASS OBJECT.**

WHY STRING IS GIVEN AS IMMUTABLE?

For maintaining original string data and modifying string data separately in different object string is given as immutable object for the reason string object as key for storing an object in map type connection object if we store modify result in the string object the object mapped with this string object can't be retrieve from this collection.

WHAT IS IMMUTABLE AND MUTTABLE OBJECT ?

IMMUTABLE :- the object that is not allowed us to modify it's data after storing is called immutable object.

FRO EG :- String

String s1 = new Sting ("vanshika");

S1 concat(“tiwari”);

Sopln (s1);

OBJECT THAT ALLOWED US TO MODIFY DATA IN IT’S MEMORY IS CALLED MUTABLE OBJECT.
For eg :- String buffer object.
StringBuffer sb = new StringBuffer(“yash”);
Sb.append(“yash”);
Sopln(sb);

2.Public int length();

This method will return an integer value representing the number of character existing in the string including space.

Eg:-

```
        Public static void main(String[] args)
          {
String s1= “student”;
String s2= “Student”;
String s3= “STUDENT”;
```

3.public String concat(String3);

4.public int compareTo(String);

5. public int compareToIgnoreCase(String);

6. public boolean startwith(string s)

7. public boolean endwith(String s)

8. public boolean contains(String s)

9.public string replace(old character, new character)

10.public char charAt(index)

11.Public int indexOf(string s):_ It will return an index value where the first occurance of the sepcified string.

12. **Public int LastIndexOf(string)** :- it will return an index value where the last occurance of the specified string.
13.
14. **Public string substring(int startIndex)**
15. **Public string substring(start index, int end index)**
16. **Public string toLowerCase()**
17. **Public string toUpperCase()**
18. **Public string trim()**
19. **Public string split()**

W.A.P. TO VARIABLE LOGIN DETAIL(USERNAME& PASSWORD)

W.A.P. TO COUNT TOTAL CHARACTERS. & W.A.P. TO COUNT NO OF VOVELS & W.A.P. TO COUNT NO OF CONSONANTS IN GIVEN STRING…

String buffer is also used to store sequence of character StringBuffer is mutable and thread safe.

1)public synchronized int length

2)public synchronized int capacity()

3)public synchronized stringBuffer append()

4)public ensureCapacity(int)

EnsureCapacity method can be used to set a particular capacity value toStringBuffer object

If the provided capacity value is less then 16 then StringBuffer objectwill take 16 as capacity value .If the provided capacity value is greater then 16 StringBuffer object t and less then 34 then StringBuffer will take 34 as capacity value .

If the provided capacity value is greater the 34 then String Buffer object will take specified value as capacity value.

5) public insert()

This method is used to set new char in given index.

6) setCharAt()

This methd is used to set new char in given index.

7) delete (start index, end index)

This method is used to delete StringBuffer content from given index. Up to last index.

8) deleteCharAt()

This method will delete specified character from given index.

9)setLength()

This method is used to set length of StringBuffer object content.

10)reverse()

This method is used toreturn reverse String.

Ques:- what is difference b/w String and StringBuffer?

StringBuilder:- StringBuilder was introduced in JDK 1.5 it is also used to store sequence of characters but String Builder is not synchronized (“not a thread safe”) All method provided by StringBuffer is also used in StringBuilder.

Ques:- what are difference b/w StringBuffer& StrinjgBuilder?

Ans:-

WHY EXCEPTION?

End user knowingly and non knowingly may enter wrong values because of these wrong values program execution will be terminated normally.

What is exception?

There are two types of errors in java.

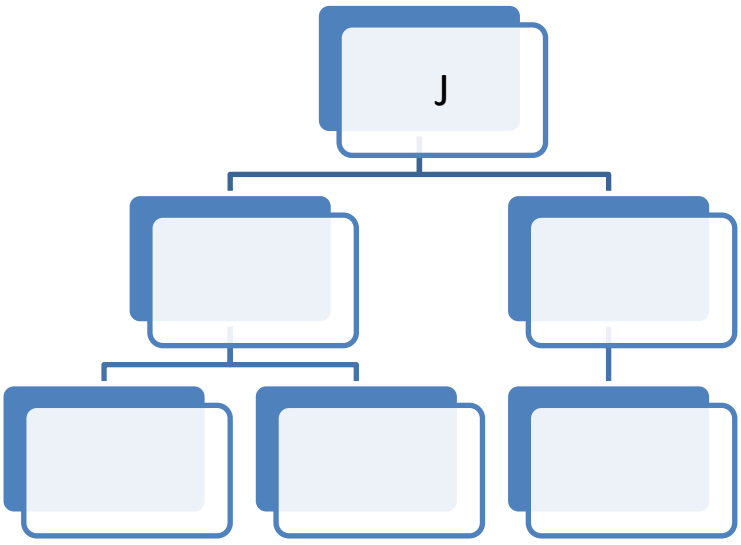
1. Compile time error. (syntax error)
2. Run time error. (exception)

1. **Compile time error :-** these are the problems by compiler and compilation time these errors are generated due to developers mistake.
   FOR EG:- string s = “abc”;

2. **Run time error :-** run time error generated by end user due to wrong input. These are the problems for which we are unable to provide solutions programmatically and these errors are not identify by compilers.

There are two types of exceptions…

1. **Predefine exception.**
2. **User define exception.**



1. **PRE DEFINE EXCEPTION** :- these exceptions are define by

programming language and provided along with java s/w

There are two types of predefine exception.

1. Checked exceptipon.
2. Unchecked exception.

What is diffeence b/w checked exceptipon or unchecked exception.

- Checked ecxeption is an exception recognised at compilation time but not occurred at compilation time.
- Unchecked exception are not recognised at compilation time, these exceptions are recognised at run time by jvm.

There are two types of checked exception.

1. Pure checked exception.
2. Partially cheked exception.

- **Pure checked exception :-** if any checked exception is having only checked exception as sub class then this excepiton is known as checked exception.

- **Partially cheked exception** :- if any checked exception contains atleast one sub class as unchecked exception than that checked exception is called partially checked exception.

Ques:- what is Exception handling?

Exceptional handaling means terminating program excution normally or smoothly by applying different operation
By using exception handling we will perform following task :-

- **Identify errors**
- **Stop abnormal termination.**
- **Display user understandable error message**
- **Prompting message to read next correct value**

The exception message will displayed by JVM are not understandable to end user, its programmers responsibility to display user understandable message for exception raised in the program. For this we must use keyword try and catch.

try:- try is a block where we grouped set of stmt in which exception can raised.

Catch:-catch is used for catching exception these are raising from try block stmt**.**

Catch is a parametrized block , the parameter should be the exception class name whose exception we want to catch .Inside catch block we want to catch .Inside catch block we place stmt to take action against the exception we got.

The action could be

Displaying user understandable msg

Displaying the caught exception

Rethrowing this exception

Catch must be placed immediately after try

throw keyword:- throw is a java keyword it can be used to raised exception instantially as per there developer Application requirement.

Syntax:-throw new Exception_Name("exception description);

//program

There are two ways to handle exception

1)throws keyword

2)try catch finally

throws keyword:- it is a java keyword it can be used to bypass the generated exception from the present method or constructor to the caller method or constructor.
In java application throws keyword keyword will be used in method declaration not in method body. In java application throws keyword
allow an exception class name it should be either same as the generated exception or super class to the generated exception or super class to the generated exception. It should not be sub class to the generated exception .

throws keyword exception allows more than one exception in method prototypes .In java application throws keyword will be utilize mainly for checked exception.

Finally:- In finally block we placed definitely executable stmt.

Qus:- what is difference b/w throw &throws?
Qus:- can we use another try block inside try block?
Ans_ yes, we can

//program

MULTITHREADING

PROGRAM :- PROGRAM IS A SET OF INSTRUCTIONS TO REPRESENT A PARTICULAR TASK.

PROCESS :- PROCESS IS A FLOW OF EXECUTIONTO PERFORM A PARTICULAR TASK.

WHAT IS DIFFERENCE B/W SEQUENTIAL, PARALLEL AND CONCURRENT FLOW OF EXCUTION.

SEQUENTIAL FLOW :-IN SEQUENTIAL FLOW TASK IS EXECUTED AFTER COMPLETION OF FIRST TASK.

Eg:- one faculity ---- multiple classes , handling comes under sequential flow for this one room is sufficient.

Parallel flow:- executing two tasks at a time without any pauses is called parallel flow.

Eg:- two different faculty handling two different classes at a time is called parallel flow.

CONCURENT FLOW :- in concurrent flow multiple task are executed at a time but with pauses , when a our task execution is paused other task get chance to execute where in parallel flow both task is executed in their own path.

For eg :- single facility handling multiple classes having pauses.

Qus:-WHAT IS DIFFERENCE B/W PROCCES AND THREAD.

Process :- process is heavy weight, to handling it system has to consume more memory and more execution time , it will reduce application performance.

Thread :- thread is light weight to handle it system has to consume less memory and less execution time , it will improve application performance.

There are two thread models to execute applications

1. Single thread model.
2. Multi thread model.

1. **Single thread model :-** it will allow only one thread to execute application it will follow sequential execution flow, it will increase application execution time and reduce application performance.
2. **Multithread model :-** it will allow more than one thread to execute application it will follow concurrent execution flow, it will reduce application execution time and it will improve application performance.

Multithreading:- multithreading is process of creating multiple custom threads for executing multiple independent task to complete execution at less time.

Qus:-How we can create thread ?

Ans:- we can create thread:-

1.extending thread class.

2.Implementing Runnable interface.

1. **extending thread class:-** In this Approach we have to declare a class, It must be extended from **java.lang.Thread** class

```
class    class_Name extends Thread
{
          //implementation

}
```

2. **Implementing Runnable interface:-** In this Approach we have to declare a class, it must be implemented from **java.lang.Runnable** interface.

```
Class    class_Name implements Runnable
{
  //implementation
```

}

Extending thread class-

1.declare a user define class.
2.extend this class from thread method(java.lang.Thread)
3.override thread class with the implementation defined thread class with implementation representing a particular task. Which we want to perform by creating a thread.
4.In main class , in main method , create object for user defined class.
5.access thread class provided start() method on user defined thread class object reference variable**.**

The main intension of start() methods is to create new thread and to access run() method bypassing the generated thread.

//Mythread.java

```
public class Mythread extends Thread {

	public void run()
	{
		for(int i=1;i<=10;i++)
		{
			System.out.println("User Thread"+i);
		}
	}



    }
```

//Test.java

```
public class Text {
```

```
	public static void main(String[] args) {
		Mythread mt= new Mythread();
		mt.start();
		for(int i=1;i<=10;i++)
		{
			System.out.println("Main Thread"+i);

	}


}
	}
```

Implementing Runnable interface:- [10:16 PM, 8/9/2022] Mata Ram: In java application create thread if we use first approach the we have to declare an user define class and it must be extended from java.lang.thread class , in this context it is not possible to extend other classes , if we want extend any other class along with thread class then it will represent multiple inheritance , it is not possible in java .

Class Mythread extends frame, Thread
{
//implementation
}

To overcome above problem we have to use runnable interface .

Class Mythreads frame implements Runnable
{
//implements
}

1. Declare user defined class.
2 implement java.lang.runnable interface.
3. Provide implementation in Run() method which we want to execute by creating a thread.

4. In Main class, in main method create a thread and access user defined thread class Run()methods.

//Mythread.java-

```
package run.inter;

public class Mythread implements Runnable {

	@Override
	public void run() {
		for(int i=1;i<=10;i++)
		{
			System.out.println("User Thread"+i);

	}

}
	}
```

//Test.java

```
package run.inter;

public class Test {


	public static void main(String[] args) {
		Mythread mt= new Mythread();
		Thread th= new Thread(mt);
		th.start();
		for(int i=1;i<=10;i++)
		{
			System.out.println("User Thread"+i);


	}

}
	}
```

In multithreaded programming we must find from which class run() method is executing for this purpose we must follow below clues

we must observe the constructor used in thread object creation if no args constructor used in current class run() method will execute from thread class or from sub class

Thread th=new Thread();

th.start();

Run() method execute from thread class.

Mythread mt= new Mythread();

mt.start();

run()method execute from Mythread class.

thread t= new Mythread class

t.start().

Runnable r= new Mythread();

r.start();

//unresolved compilation problem

Method start() is undefined for the type Mythread.

Thread's life cycle ---new/born

Ready /Runnable

Running

Wait , sleep,suspend, blocked

Dead

Thread class Library:-

1.public thread()
2.public Thread (String name)
3. public Thread (Runnable r)
4. public Thread (Runnable r, String name)
5. public Thread (ThreadGroup tg, Runable)
6. public Thread ThreadGroup tg , String name)
7. public Thread (ThreadGroup tg, Runnable r, String name)

Daemon Thread

These threads are running internally to provide services to same other thread and it will be terminated along with thread which are taking service.

To make a thread as daemon thread we have to use following methods.

Public void setDaemon(boolean b)
m.set daemon(true);

To check thread is daemon thread or not we have to use following method.
Public boolean is Daemon()

Eg:- in Java garbage collector is thread Running internally inside JVM and it will provide garbage collection services to JVM and it will be terminated along with JVM automatically.

SYNCHRONIZATION

Java application if we execute more than one side on a single data items then there may be a chance to get data inconsistency.

It may generate wrong result in Java application

In Java application provide data consistency in the above situation we have to use synchronization is a mechanism it able to allow only one thread at a time

Synchronization is going on bases of locking mechanism, if we send multiple thread at a time to sing to nice had which is having highest priority once a thread gets lock from lock manager then that thread is eligible to enter in synchronised area

In java application we are able to achieve synchronization

in Java application in the following to ways

1) Synchronized methos
2) Synchronized block

Synchronized method:- it is a normal Java methods it will allow one thread at a time to execute instructions it will not allow more than one set at a time it will allow other thread after compilation of present thread execution.

// A.java :-

```
package synchronization;

public class A {

	synchronized  void m1()
	{
		for(int i=0;i<=10;i++)
		{
			System.out.println(Thread.currentThread().getName());
```

```
		}
	}
	}
```

//Mythread.java

```
package synchronization;

public class Mythread extends Thread {
	A a;
	Mythread(A a)
	{
		this.a=a;
	}
	public void run()
	{
		a.m1();

	}

		}
```

//Mythread2.java

```
package synchronization;

public class MyThread2 extends Thread {
	A a;
	MyThread2(A a)
	{
		this.a=a;
	}
	public void run()
	{
		a.m1();

	}

		}
```

//Mythread3.java

```
package synchronization;

public class Mythread3 extends Thread {
	A a;
	Mythread3(A a)
	{
```

```
		this.a=a;
	}
	public void run()
	{
		a.m1();

	}

		}
```

//Test.java

```
package synchronization;

public class Test {
	public static void main(String[] args) {
	A a= new A();
	Mythread m1= new Mythread(a);
	MyThread2 m2= new MyThread2(a);
	Mythread3 m3= new Mythread3(a);
	m1.start();
	m2.start();
	m3.start();




	}

	}
```

If we use synchronised method to achieve synchronization throughout the method irrespective of actual requirement. If we need synchronization up to a block inside the synchronized method then it will provide unnecessary synchronization for the remaining part of method it will increases execution time and reduce application performance.

Syntax:- Synchronized(Object a)

{

}

It is set of instructions it able to allow only one side at a time to execute instruction it will not allow more than one thread at a time.

//A.java:-

```
package synchronization.block;

public class A {
	 void m1()
	{

		for(int i=0;i<=10;i++)
		{
			System.out.println(Thread.currentThread().getName()+"not syn");
		}
		synchronized (this) {
			for(int i=0;i<=10;i++)
			{

	System.out.println(Thread.currentThread().getName()+"syn");

			}

		}

	}
	}
```

Mythread.java:-

```
package synchronization.block;

public class Mythread extends Thread {
	A a;
	Mythread(A a)
	{
		this.a=a;
	}
	public void run()
```

```
	{
		a.m1();

	}

		}
```

Mythread2.java:-

```
package synchronization.block;

public class MyThread2 extends Thread {
	A a;
	MyThread2(A a)
	{
		this.a=a;
	}
	public void run()
	{
		a.m1();

	}

		}
```

Mythread3.java:-

```
package synchronization.block;

public class Mythread3 extends Thread {
	A a;
	Mythread3(A a)
	{
		this.a=a;
	}
	public void run()
	{
		a.m1();

	}

		}
```

Test.java:-

```
package synchronization.block;

public class Test {
	public static void main(String[] args) {
	A a= new A();
	Mythread m1= new Mythread(a);
	MyThread2 m2= new MyThread2(a);
	Mythread3 m3= new Mythread3(a);
	m1.start();
	m2.start();
	m3.start();









	}

}
```

Jdbc

Terminologies:-

1. **Presentation logic :- HTML/css/javascript jquery/bootstarp angular/react asp/jspp**
2. **Persistence :-** the process of saving and managing data for long time(permanently) is called persistence. The data stored in local, global variable and objects are allocated memory in RAM area which is a temporary memory so we can't get that data after the execution of application.
   in order to overcome this problem we need to make out an application writting data to persistence like :- files , data base s/w.

persistence store:- the place where data will be saved for long time. For eg. Files or database s/w.

persistence data :- the data of persistence store is called persistence data.

Persistence operation:- insert , update, delete, select operation performs on the persistence data.

These are also called as CRUD/CURD/SCUD

Persistence logic :- a logic that is reqired to perform persistence operation is called persistence logic.

For eg. I/o streame code , IDBC

Persistence technology :- the technology that can be use to develop persistence logic is called persistence technology. **For eg. IDBC.**

"persistence technology can be used to write persistence logic fo performing persistence operation on persistence data of persistence store."

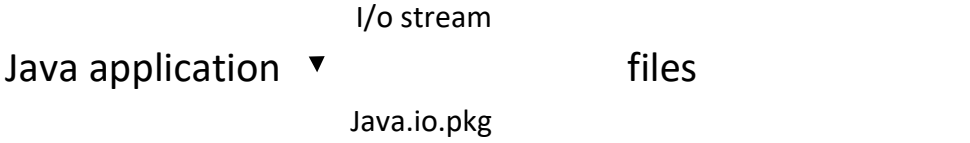
Every basic application contains following logic:-

1 presentation logic :- the logic that gives user interface to end user to supply inputs and to view results.

2 business/service logic :- the main logic that generates the result.

3 Persistence logic :- the logic that performs persistence operations.

Files and data base s/w allocates memory in secondary storage like HDD due to this their data remains permanent.



JDBC

Java application ▼ DB s/w

Javax.sql pkg

Limitations with file system:-