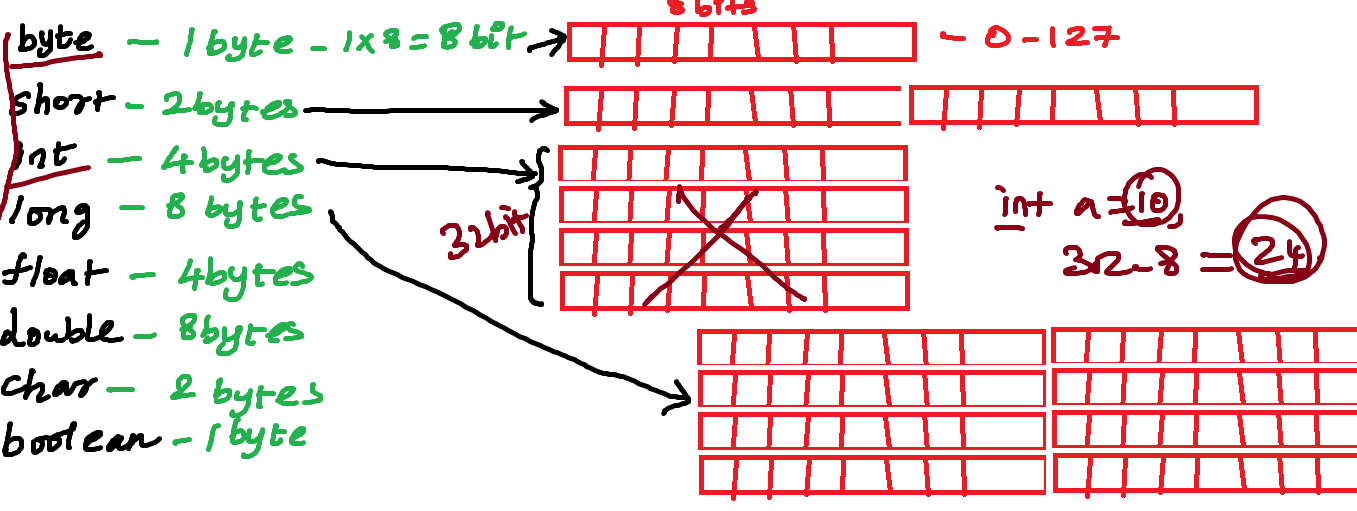
Java:-

Java datatype memory allocation:-



block -

{

statements;

statements;

}

javac filename.java

java filename -> IDE -> Run as-> Java Application

Control Statement:-

By default java executes your statement from top to bottom

Types of control statement:-

1. conditonal control statement

if:-

if(condition){

statements;

}

=============================

if(condition){

statements;

}

else{

statements

}

=============================

if(condition){

statements;

}

else if(condition){

statements

}

else if(condition){

statements

}else{

statements

}

=============================

switch(1) {

case 1: {

System.out.println("statement 3");

System.out.println("statement 4");

System.out.println("statement 5");

}

case 2: {

System.out.println("statement 6");

System.out.println("statement 7");

System.out.println("statement 8");

}

case 3:{

statements;

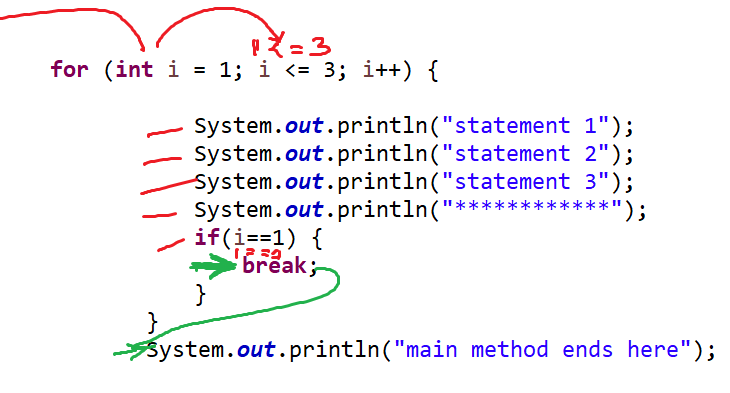
}

default:{

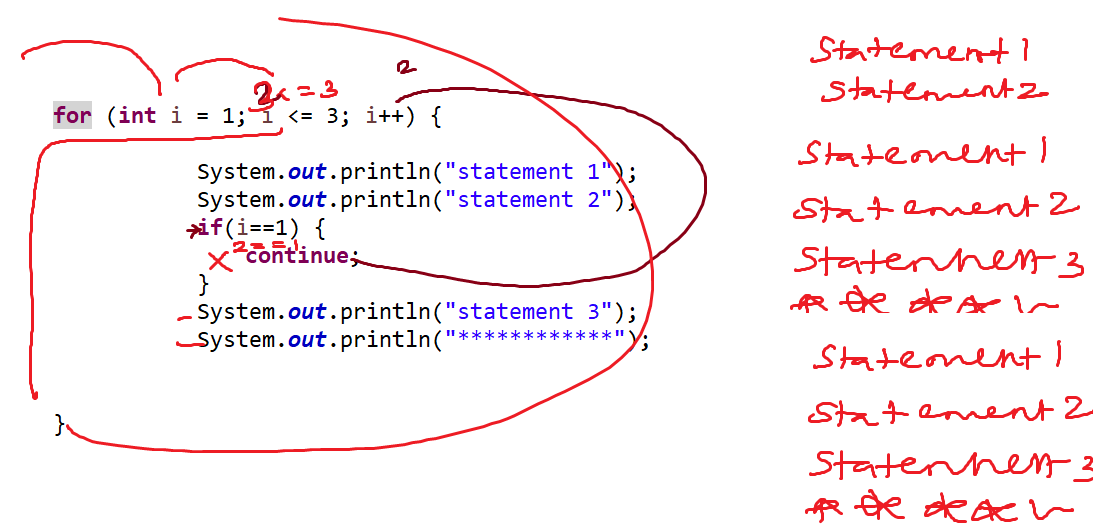
}

2. unconditional control statement

Break



Continue



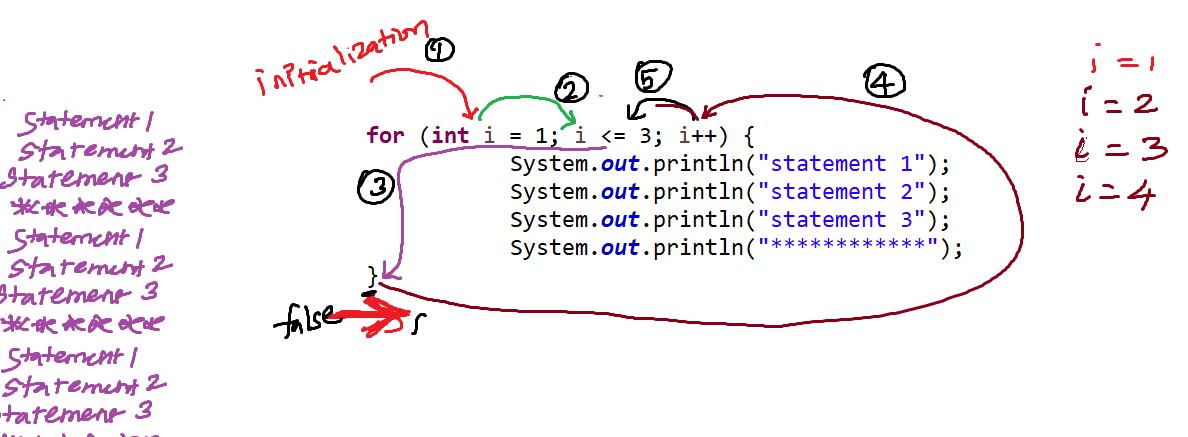
Looping Statement:-

for loop:-

for(initialization;condition;increment/decrement){

statements;

}



=================================

while(condition){

statements;

}

=================================

do{

statements;

}while(condition);

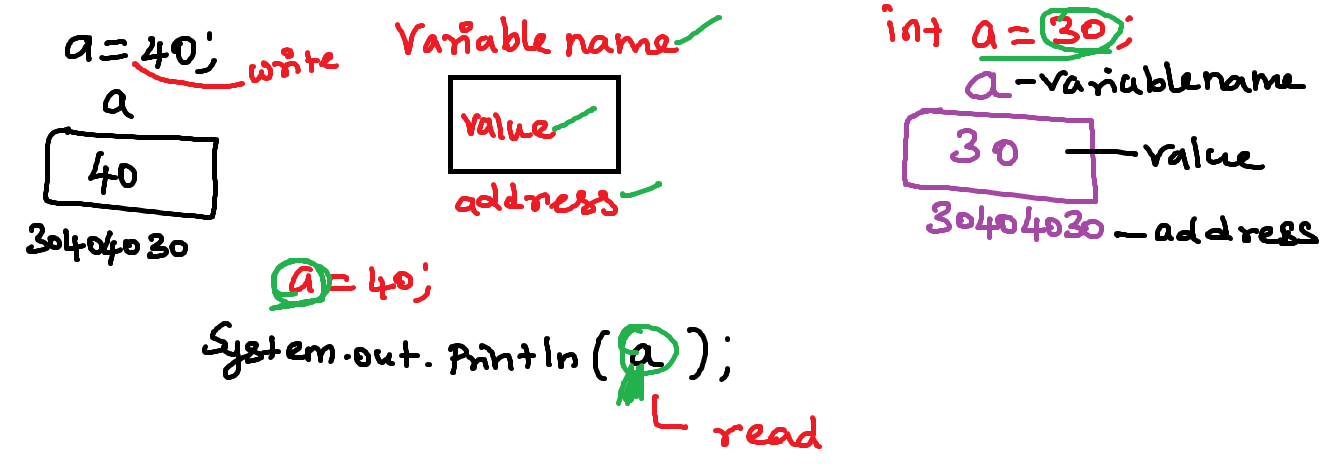
while vs do while:-

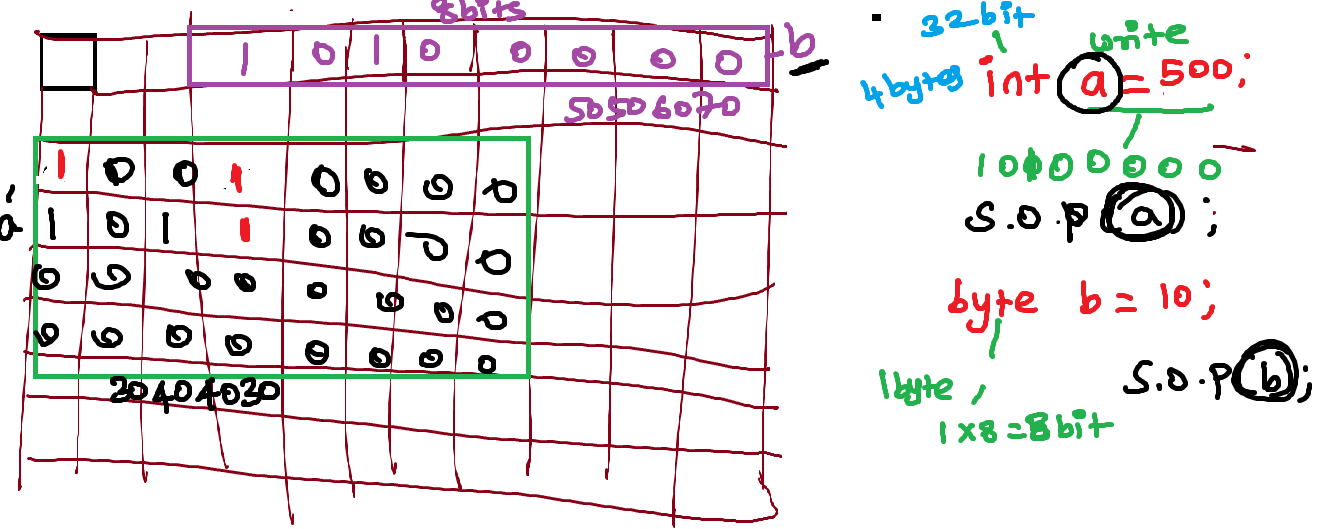
Variable:-

\* It is an identity given to the memory location

\* In java value stored in the memory either read or write can be done through java variable

\* Variable stores data or a value





Methods or functions:-

\* Methods tend to do a particular task

\* Method or function in order to perform logic needs a data

\* Method in java should not cross more than 15 lines of code then create an method and call it

\* Method never executed automatically programmer need to call the method explicitly

\* Method definition -

\* Method call - should invoke the matching method definition

Method Syntax:-

==============

accessmodifier nonaccessmodifier returntype methodname(arglist){

statements;

statements;

}

Eg:-

void add() {

int a = 10;// data

int b = 20;// data

int c = a + b;// logic

System.out.println(c); //printing the value

}

Ways to write methods:-

=========================

1. Function with no argument and no return type

void circle() {

int radius=34;

float area=3.14f\*radius\*radius;

System.out.println(area);

}

2. Function with no argument and with return type

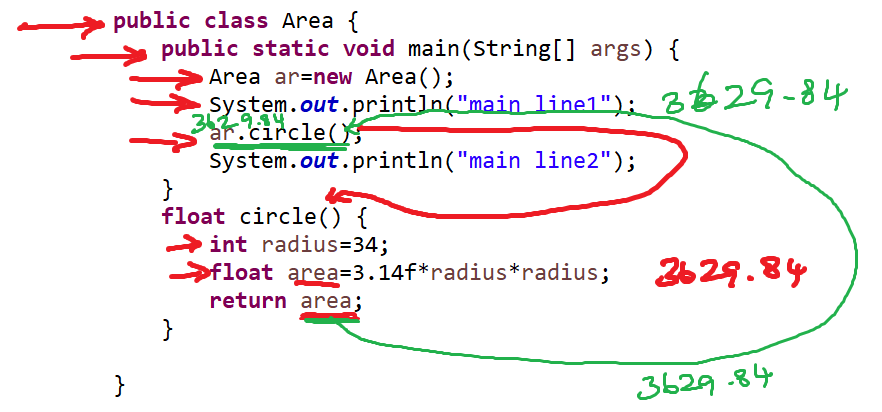
float circle() {

int radius=34;

float area=3.14f\*radius\*radius;

return area;

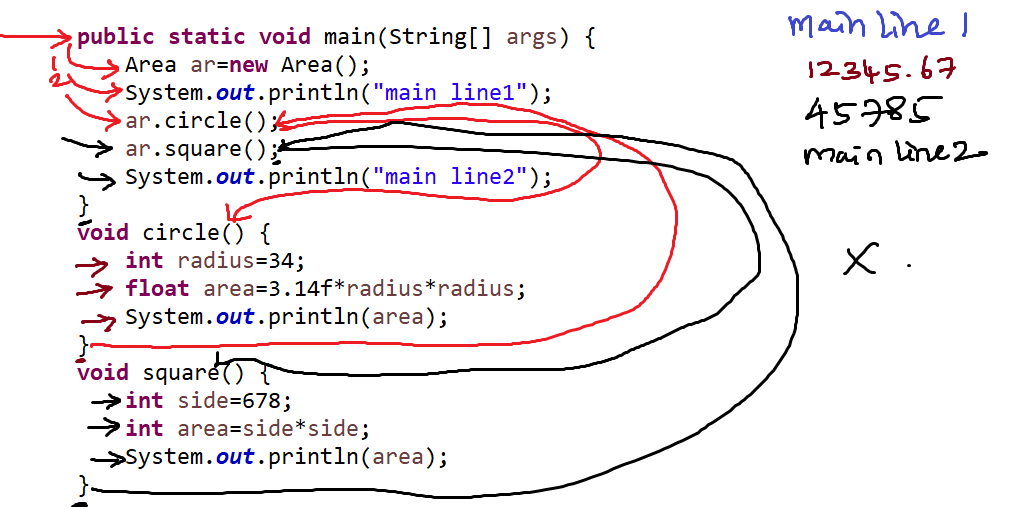
}



3. Function with argument and without return type

4. Function with argument and with return type

**Function call to the function definition and flow control:-**



Object Creation Syntax:-

Classname objectname=new Classname();

Task:-

create a class call as Area

create a two methods

1. main method

2. circle() inside the circle method implement circle area

3. call the circle method from main method

Class:-

Class:-

\* Class is a blueprint of an object

\* class template for an object

\* class contains members

Class members - variable, methods, constructor, static block

Object are used to access members of the class

1. Userdefined java class - Java classes created by programmer

Eg:- Geometric, Area

2. Predefined java Class - The java classes already created and it kept inside the runtime system library are called as predefined classes

Eg:- Scanner, String

Object:-

* + It contains properties and behavior
  + Instance of class called as object
  + Objects are made from template (class)

Method Overloadding:-

\* Method with same name different arguments are called as method overloading.

Payment -> Task is makePayment -UPI, OnlineBabnking,

Cash, cards

Association:-

* + ­­­communication between one class to another class
  + Object of one class can communicate another class
  + Communication means - Access the variable and method of one class to another class

Association types:-

1. IS-A Association– inheritance (extends)

**Eg:- Employee IS A Human**

class Human{

}

class Employee extends Human{

}

**Eg:-**

**Lion IS A Animal**

**Apple IS A Phone**

2. HAS-A Association – create an object of one class to another class

**Eg:- Car has a Engine**

Class Engine{

}

Class Car{

Engine e=new Engine();

}

Has-a Association Type:-

2.1 Aggregation – weak association

2.2 Composition – strong association

**Datatype:-**

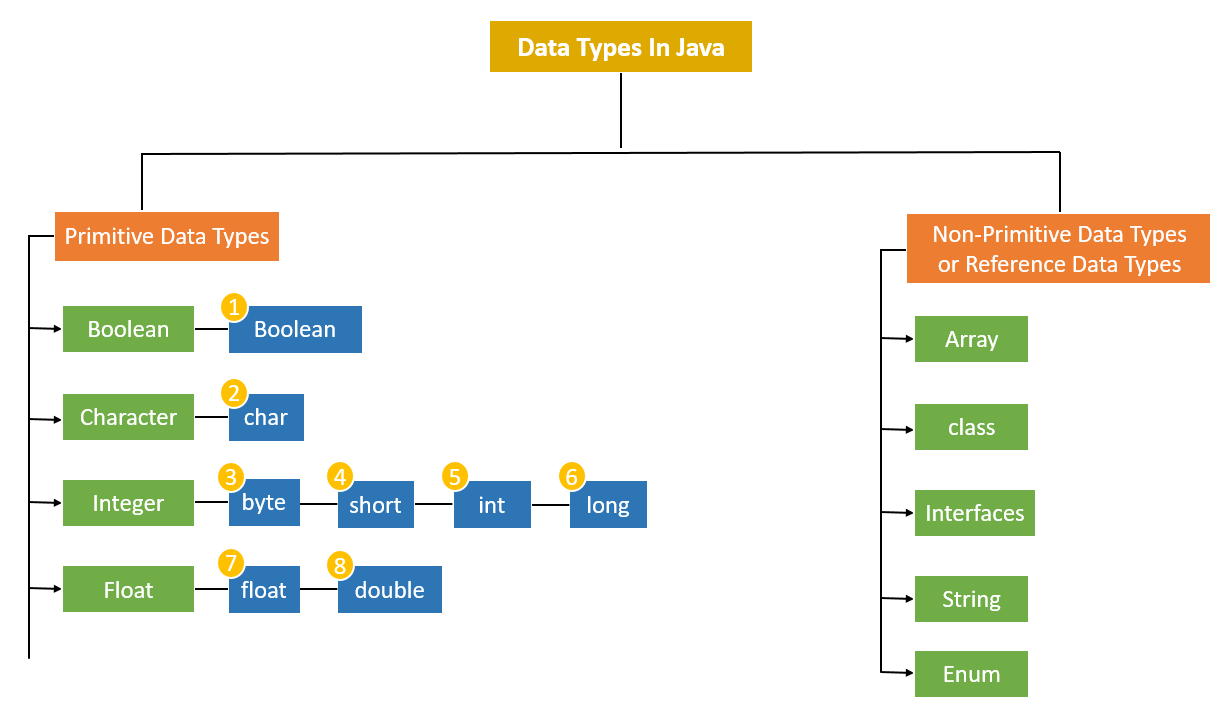
int – stores whole numbers – age, accno, productId

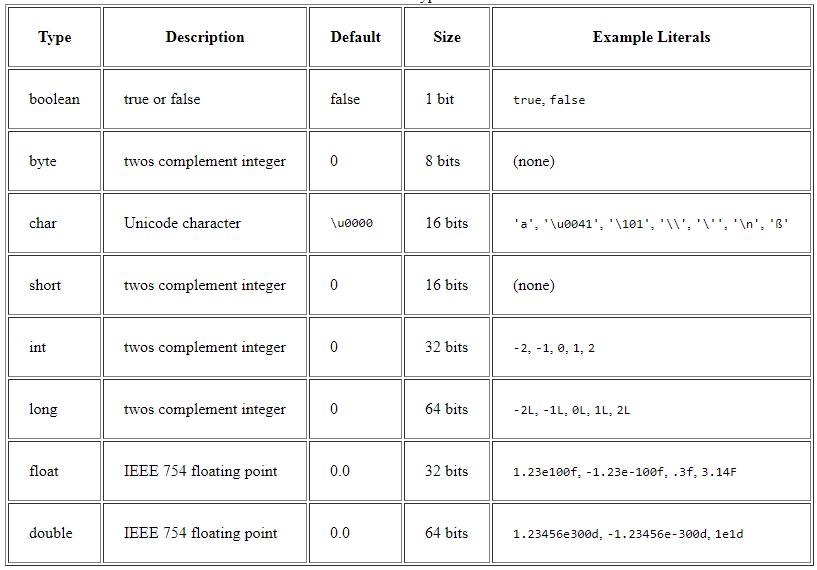
float – store decimal values - gpa,radius,gst

char – single char input – gender, yes or no

boolean – true or false

String – collection of characters





**Java Naming Conventions:-**

**Class Name** - class name should be meaning full

**If the class name is single Word**

Starting character of class name should be capital

Eg:

Student

Transaction

**If the class name is multiple Word**

Starting character of each word should be in capital

Eg:-

MedicalStudent

MedicalTransactionDetails

**Variable:-**

**If the variable is an single word**

All letters must be in lower case

**If the variable name is more than one word**

Starting word all letters in lowercase

Second word onwards each word-starting letter must be in capital

Eg:-

name

employeeName

rateOfInterest

**Method:-**

**If the method is an single word**

All letters must be in lower case ()

**If the method name is more than one word**

Starting word all letters in lowercase

Second word onwards each word-starting letter must be in capital ()

add(){}

displayEmployeeDetails(){}

Packages: - all letters must be in lower case

Interface: - It is similar to class name convention

**Class and Object Instance and static memory allocation:-**

**public** **class** Trainee {

**int** traineeId;

String traineeName;

**int** phone;

**static** String *companyName*="HSBC";

**public** Trainee(**int** traineeId, String traineeName, **int** phone) {

**this**.traineeId = traineeId;

**this**.traineeName = traineeName;

**this**.phone = phone;

}

**void** displayTrainee() {

System.***out***.println(traineeId);

System.***out***.println(traineeName);

System.***out***.println(phone);

}

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

Trainee t1=**new** Trainee(123,"abc",97877);

Trainee t2=**new** Trainee(124,"bcd",99977);

Trainee t3=**new** Trainee(125,"ecd",88977);

t1.displayTrainee();

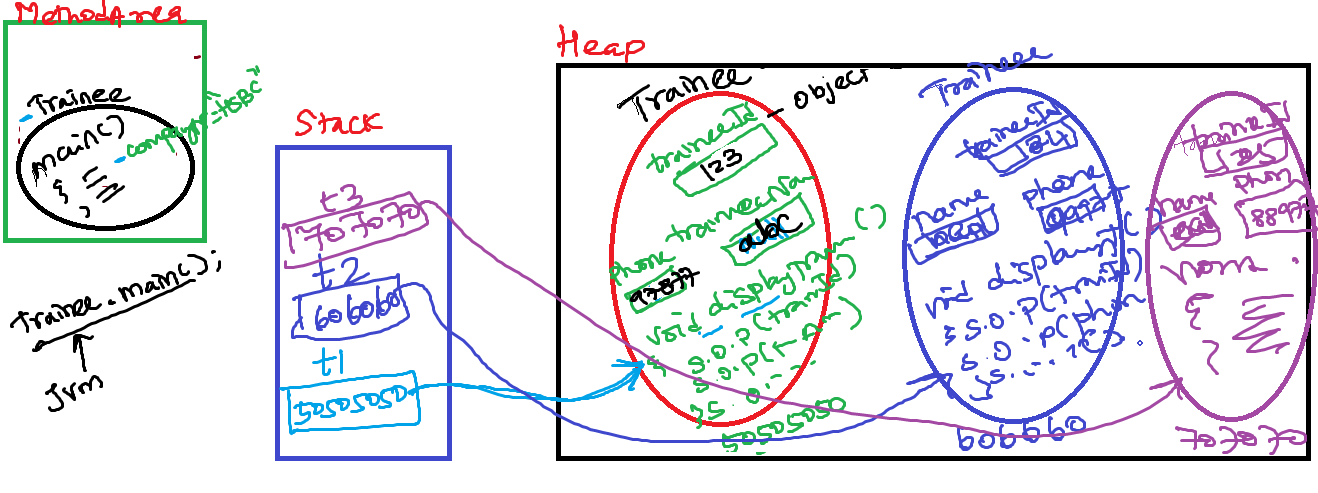
t2.displayTrainee();

t3.displayTrainee();

System.***out***.println(Trainee.*companyName*);

}

}



**Arrays:-**

- Array is stores group or collection of similar type values

- group of values holed by single reference variable

- Array size is fixed. Once size is set, we cannot increase or decrease the size of the array

- Array is random access

Drawback:-

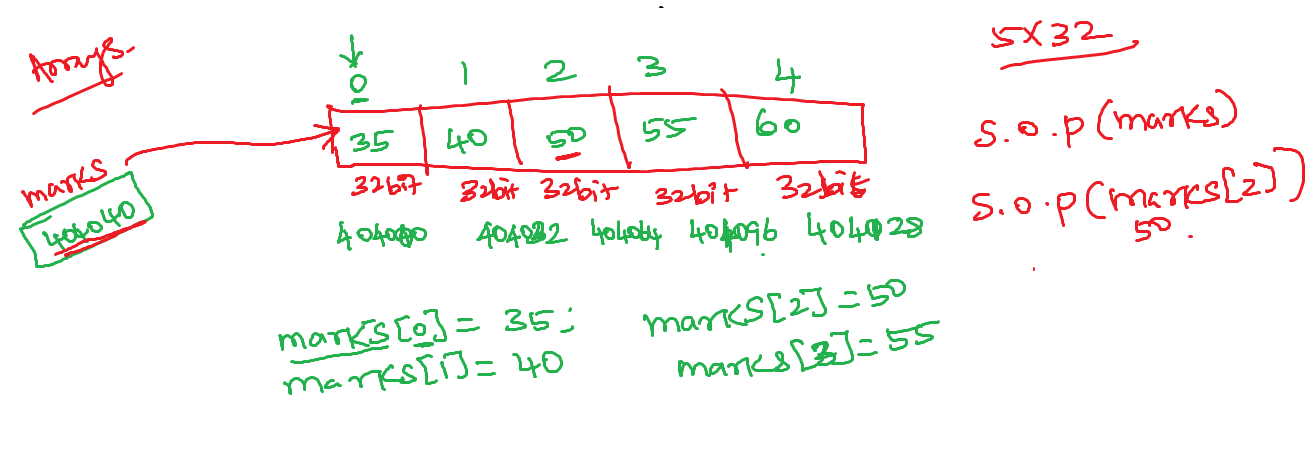
\*\*\*\*\*\*\*\*\*

Insertion and deletion is very difficult need to move

Your data back and forth physically

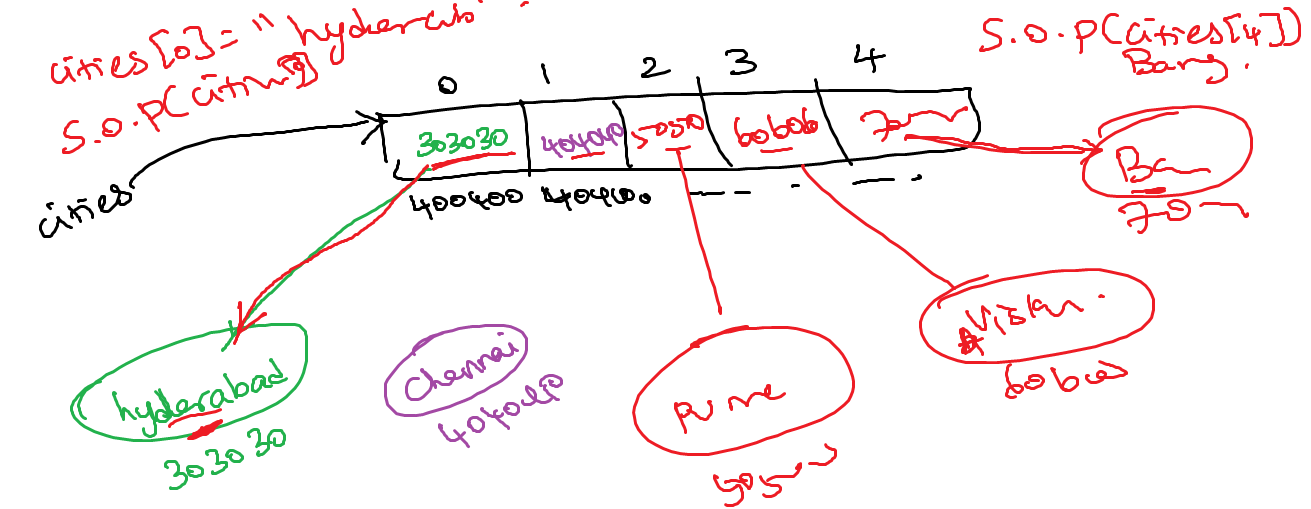
**Array Stores Primitive values:-**

int[] marks=new int[7]; // group of primitive values

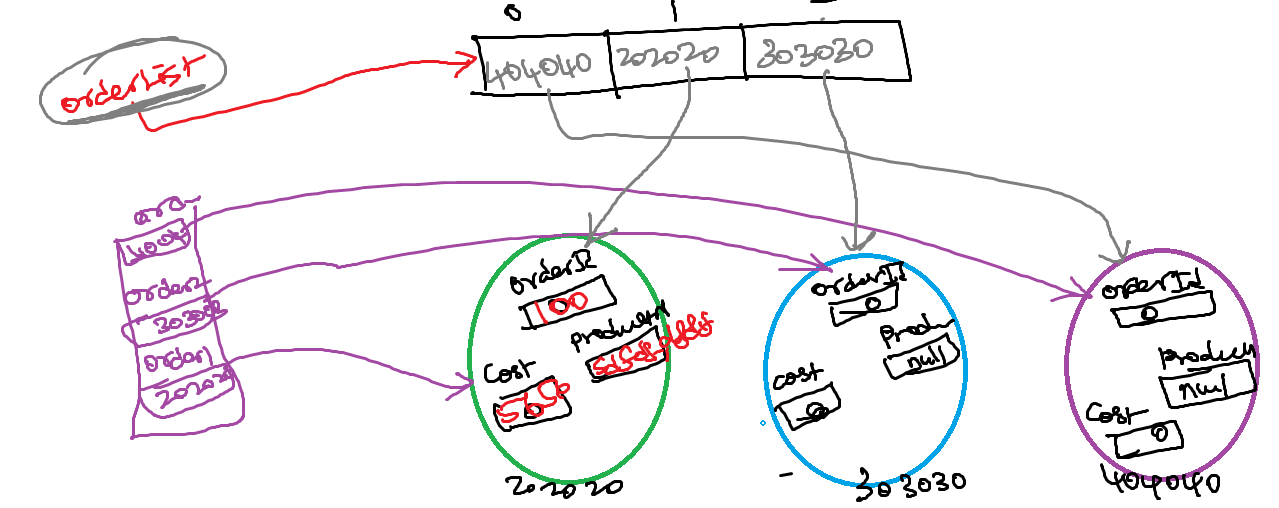


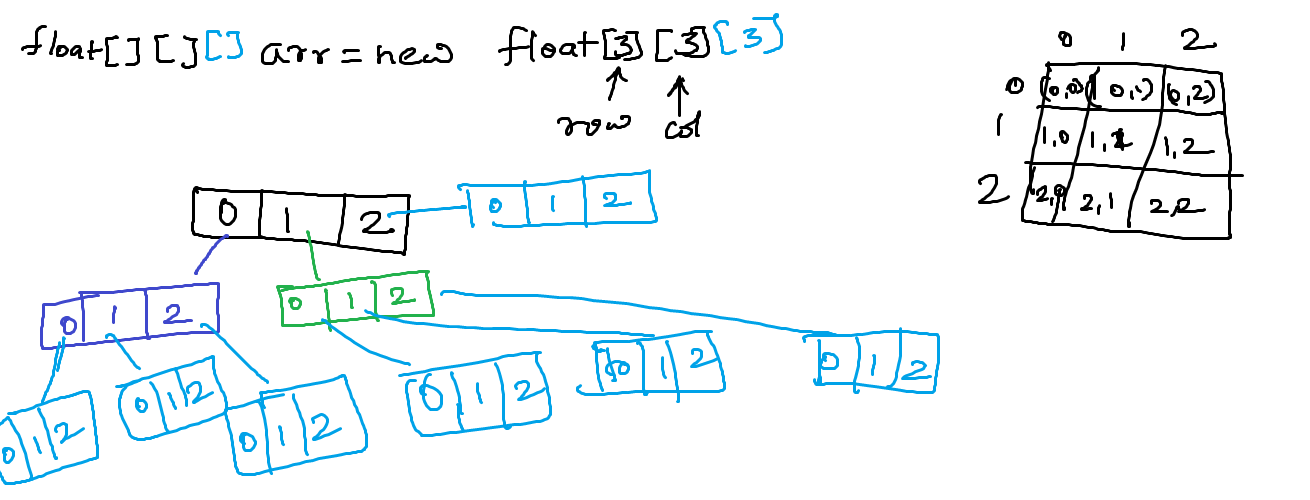
**Array Stores Reference values (objects)**

String [] cities=new String [3]; //group of String predefined reference objects



Order[] orderList=new Order[3]; //group of player user defined reference objects





Association:-

* + ­­­communication between one class to another class
  + Object of one class can communicate another class
  + Communication means - Access the variable and method of one class to another class

Association types:-

1. IS-A Association– inheritance (extends)

**Eg:- Employee IS A Human**

class Human{

}

class Employee extends Human{

}

**Eg:-**

**Lion IS A Animal**

**Apple IS A Phone**

2. HAS-A Association – create an object of one class to another class

**Eg:- Car has a Engine**

Class Engine{

}

Class Car{

Engine e=new Engine();

}

Has-a Association Type:-

2.1 Aggregation – weak association

2.2 Composition – strong association



**packages:-**

arrange your classes

visibility

naming collision

package name should be reverse of your domain name

package name should be lowercase letters

in.icici.account

SavingAccount.java

CurrentAccount.java

Statement.java

in.icici.card

CreditCard.java

DebitCared.java

Statement.java

in.icici.loan

PersonalLoan.java

HousingLoan.jav0061

Statement.java

payment

Payment.java

Transaction.java

hsbcbankapp --> www.icici.in

in.icici.account

SavingAccount.java

CurrentAccount.java

Statement.java extends SavingAccount.java

**in.hsbc.card**

CreditCard.java

DebitCared.java

Statement.java

**in.hsbc.loan**

PersonalLoan.java

HousingLoan.java extends SavingAccount

Statement.java

**in.hsbc.payment**

Payment.java

Transaction.java

**Access Modifiers:-**

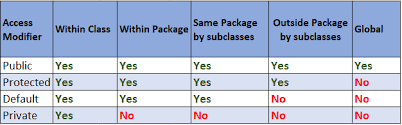
- Scope or visibility - how far it can be accessable

**private –** private members can access only inside the class outside other classes cannot access

**default -** package level visibility. Default members can access with in same class and other classes from the same packages. Other class from other packages cannot even if it subclasses also

**protected -** similar to default however the subclass of other packages classes can acces

**public –** public members can access from anywhere within the project



Non Access modifier

It changes the default behavior

Static, final, abstract, synchronized, native, volatile

**class members:002D**

- variable - static or instance

- method - static or instance

- constructor

**Inheritance:-**

Inheritance all the members of the base class will be inherited to the child class. Never inherit constructorof base class to the child class. Constructor stays in the same class

BasicPhone

calls

sms

SmartPhone extends BasicPhone

camera

whats app

**Types of IS-A Inheritance**

1. Simple Inheritance

Base class

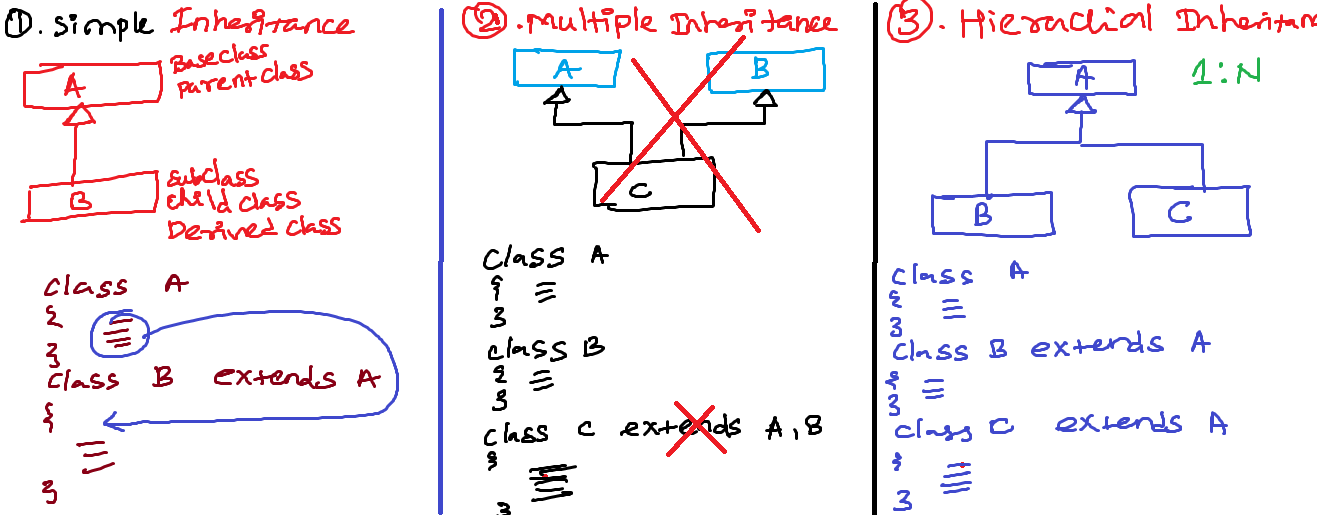
Sub Class

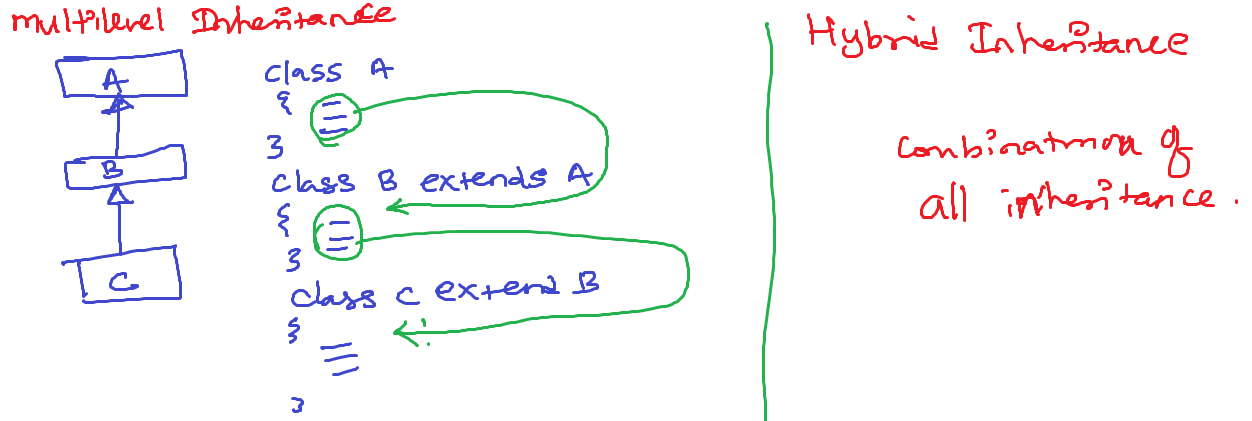
2. Multiple Inheritance

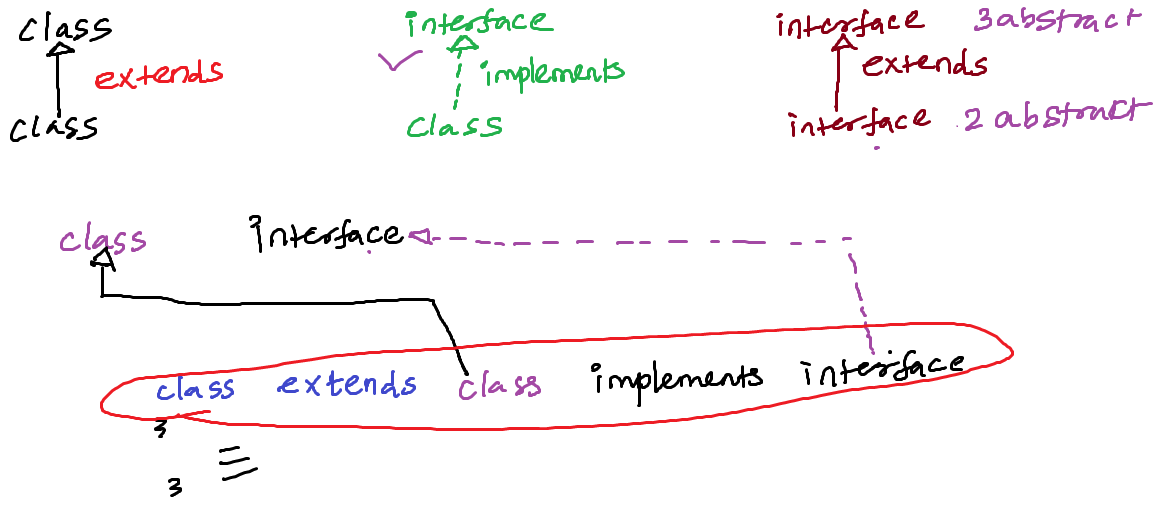
3. Multilevel Inheritance

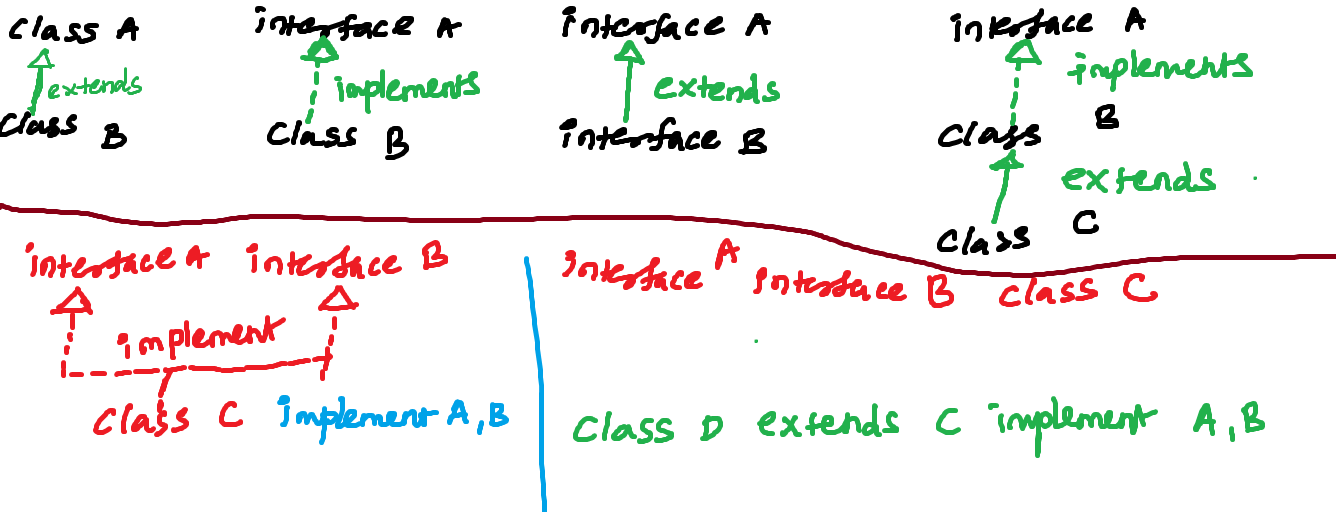
4. Hierarichal Inheritance

5. Hybrid Inheritance









Polymorphism:-

Poly - many

morhism - forms

Two types of polymorhism:-

1. Method overloadding

method with same name different types arg

void sendParcel(IndiaPost){

//implementation

}

void sendParcel(Courier){

//implementation

}

void sendParcel(ByFriends){

//implementation

}

void sendParcel(BySelf){

//implementation

}

2. method overidding

Method with same name same arg

class Courier{

void sendParcel(Courier){

//implementation1

}

}

class MasterCourier extends Courier{

void sendParcel(Courier){

//implementation2

}

}

Method Binding or Function call Resolving:-

1. Compile time - javac filename.java

2. Runtime - java filename

Two types of polymorphism:-

1. compile time polymorphism or static binding or early binding

method overload

2. runtime polymorphism or dynamic binding or late binding

method override

add(10);

void add(){

}

void add(int a){

}

void add(int a,int b){

}

void add(float a){

}

final keyword in java:-

1. final keyword can be used

before variable

before method name

before classname

Abstraction:-

Hidding an complex details giving an essential details

public boolean billPayment(int amount,String paymentMode);

abstract method

If class is an abstract then u cant make objects for an

abstract class

final class

- IS-A not possible (extends wont work)

- HAS-A association (by creating an object)

abstract class:-

- IS-A is possible (extends work)

- HAS-A association (by creating an object)d

