

JAVA

Boilerplate

```
class as{
    Run | Debug
    public static void main(String args[]){
        //code to be executed
    }
}
```

Comments

```
// TYPE 1 : MANY LINE

/*
int x=4;
for(int i=0;i<10;i++){
    System.out.println(x*i);
}
*/

//TYPE 2 : SINGLE LINE

// int x=4;
// for(int i=0;i<10;i++){
//     System.out.println(x*i);
// }
```

Data Types and variable

```
double x=10.1;
// where,
// double = data type
// x = variable
// 10.1 = double literal

float t=10.1f;
// where,
// float = data type
// t = variable
// 10.1 = float literal
```

//RIGHT WAY TO WRITE VARIABLE

```
// Name is case sensitive
int x=8;
int X=9;
//can start with only alphabet,$,_
int _t=7;
int _2=7;
String $="money";
int age=18;
```

//WRONG WAY TO WRITE VARIABLE

```
//cannot start with _
String _="nothing";
// Should not be a keyword (like void )
int void=4;
// white space is not allowed
String my name="rahul";
//must not start with digit(like 1)
String 2="bot";
```

```
int x=2,y=3,z=4;
System.out.println(x+","+y+","+z);//2,3,4
```

initializing multiple similar datatype variable

```
System.out.println((float)3/4);//0.75 3 become float first
System.out.println((float)(3/4));//0.0 because(3/4) is int solved first
System.out.println((3.0/4));//0.75 because(3.0/4) 3.0 is float
System.out.println((3/4.0));//0.75 because(3/4.0) 4.0 is float
```

```
System.out.println(2+'5'+"3"+5.9+3+(int)'2'+1+3);
//2+53(5 ascii value)+"3"+"5.9"+"3"+"50"(2 ascii value)+"1"+"3"=5535.935013
```

```
int x='a';
System.out.println(x);//97 ascii value of a
```

space \t is used to give 2 unit space in a line
robert downey
nextline \n is used to go in a next line
robert
downey

```
System.out.println("cost :"+10);
//cost :10
```

Wrong format to initialize

```
int x=y=6;
```

JAVA

Primitive Data Type

```
// Primitive Data Type

byte b=2;

short s=2;

int i=2;

long l=2;

double d=2.1;

float f=2f;

char c='a';

boolean bool=false;

// String is not a primitive data type. Java.lang package provides the String class therefore, it is an object
type. You can create a string variable directly like any other variables as -
String S="rahul";

//constant declaration with final keyword we cannot update it
final int x=3;
System.out.println(x);
```

Scanner Class

(import `java.util.Scanner`)

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

String s=sc.next();

char c=sc.next().charAt(index:0);

//it is needed to put sc.nextLine before String s1=sc.nextLine();
sc.nextLine();//it will consume press(\n) after input like enter 2 and then (press)enter
String s1=sc.nextLine();

int x=sc.nextInt();

float f=sc.nextFloat();

double d=sc.nextDouble();

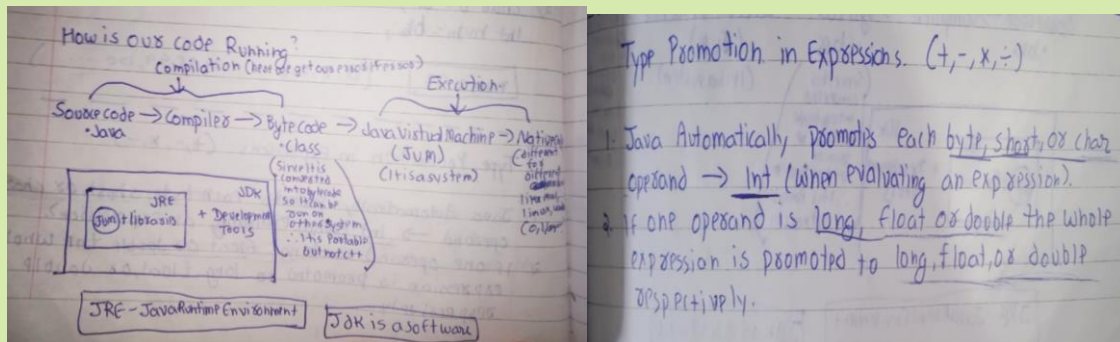
boolean b=sc.nextBoolean();

short sh=sc.nextShort();

long l=sc.nextLong();

byte by=sc.nextByte();
```

JAVA



```
// one way to initialize the variable
int x,y,z;
x=y=z=2;
// second way to initialize the variable
int a=2,b=4;
```

Operator Boolean not change into int in java but in c Boolean convert into int by default

```
class aks{
    Run | Debug
    public static void main(String args[]){
        int a=1,b=2,c=3;
        System.out.println(c>b>a);
        //it will not work but in c it work
        if(c>b>a){
            System.out.println(x:"true");
        }
        else{
            System.out.println(x:"false");
        }
    }
}
```

```
class aks{
    Run | Debug
    public static void main(String args[]){
        int x;
        x=9<5+3&&7;
        //9<8&&7
        //0&&7
        //both are not nonzero therefore 0
        System.out.println(x);
        // it will run in java not in c
    }
}
```

JAVA

```
// there will be error in all 3 code
//logic 1
int x=0;
if(x){
    System.out.println(x:"true");
}
else{
    System.out.println(x:"false");
}

//logic 2
int xx=0;
if(xx=10){
    System.out.println(x:"true");
}
else{
    System.out.println(x:"false");
}

//logic 3
if(int xy=10){
    System.out.println(x:"true");
}
else{
    System.out.println(x:"false");
}
```

Data Type Conversion /widening/implicit conversion

Byte→short→int→ long→ float→double

Type Casting / Narrowing /Explicit Conversion

```
float f=34.56f;
int x=(int)f;
```

Operators

1. Arithmetic
 - a. Binary (mathematical) operator: + - * / %
 - b. Unary operator: ++x x++ --x x—

JAVA

c. Ternary operator: ?:

```
String x = 21 > 3 ? "greater" : "smaller";  
// datatype variable = condition ? true : false
```

2. Relational (== != > >= < <=)
3. Logical (&& || !)
4. Assignment (= += -= *= %= /=)
5. Bitwise (
 - a. &(bitwise and)
 - b. |(bitwise or)
 - c. <<(shift left)
 - d. >>(shift right)
 - e. ~(one's complement)
 - f. ^(bitwise exclusive or)

Bitwise Operator in detail

JAVA

```
//0 false //1 true
// Bitwise AND(&) //if both condition is true then 1 else 0
✓//(Binary Number : 5=101,6=110)
// 1 0 1
// &1 1 0
// -----
// 1 0 0 = 4 in decimal
System.out.println(5&6);//4

// Bitwise OR(|) //if atleast one condition is true then 1 else 0
✓//(Binary Number : 5=101,6=110)
// 1 0 1
// |1 1 0
// -----
// 1 1 1 =7 in decimal
System.out.println(5|6);//7

// Bitwise XOR(^) //if one condition is true and other is false 1 else 0
✓//(Binary Number : 5=101,6=110)
// 1 0 1
// |1 1 0
// -----
// 0 1 1 =3 in decimal
System.out.println(5^6);//3

// Bitwise one's complement/not(~) //if true into 0 and false into 1
✓//trick
//~x=-(x+1)
System.out.println(~3);//-4

//Bitwise left shift (<<)
✓//logic shift binary number backward
// 00101
// 10100 (for 5<<2)
//trick
//5<<2 --> 101<<2 --> 10100 =20 in decimal
//formula x<<y=x*2 ki power y=5*2 ki power 2=5*4=20
System.out.println(5<<2);//20

//Bitwise right shift (>>)
✓//logic shift binary number forward
// 00110
// 00001 (for 6>>2)
//trick
//6>>2 --> 110>>2 --> 1 =1 in decimal
//formula x<<y=x/2 ki power y= 6/2 ki power 1=6/4=(int)1.5=1
System.out.println(6>>2);//1

System.out.println((3>4));//false
System.out.println(!(3>4));//true ! change true into false and vice versa
System.out.println(5==5 && 5>2);//true

System.out.println(5!=5 || 5>2);//true because 5>2
```

Break and Continue Statement

Break(to exit loop)

Continue(to skip specific condition iteration)

JAVA

```
for(int i=0;i<10;i++){
    if(i==5){
        System.out.println(x:"break");
        break;
    }
    System.out.print(i+" ");
}
System.out.println(x:"understood");
```

Output: break

0 1 2 3 4 break
understood

```
for(int i=0;i<10;i++){
    if(i==5){
        System.out.print(s:" here 5 is skip ");
        continue;
    }
    System.out.print(i+" ");
}
System.out.println(x:"understood");
```

Output: continue

0 1 2 3 4 here 5 is skip 6 7 8 9 understood

Math function

```
Math.max(a:2,b:3); //3
Math.min(a:2,b:3); //2
Math.sqrt(a:4); //4
Math.cbrt(a:27); //3
Math.random(); //[0-1)
Math.pow(a:2,b:3); //8
Math.abs(-2); //2
Math.ceil(a:2.3); //3
Math.floor(a:2.3); //2
Math.round(a:2.5); //3
Math.round(a:2.4); //2
```

```
int X=30;
Math.toDegrees(X); //0.52 rad
Math.toRadians(X); //1718.873 deg
Math.sin(Math.toRadians(X));
// 0(0) 30(1/2) 45(1/root 2) 60(root3/2) 90(1)
Math.exp(X);
Math.log(X);
Math.log10(X);
```

String Function

JAVA

```
String str="rahul kumara";  
//      01234567891011  
str.indexOf(str:"a");//1  
str.indexOf(str:"a",fromIndex:3);//9  
str.lastIndexOf(str:"a");//11  
str.contains(s:"ahul");//true  
str.startsWith(prefix:"ra");//true  
str.endsWith(suffix:"ra");//true  
str.replace(target:"a",replacement:"rt");//rrthul kumtrrt  
String str2="singh";  
str.concat(str2);//rahul kumarasingh
```

```
String x="hello";  
String a="kumar";  
  
x.compareTo(a);//comparing if same=0,small<0,greater>0 && A=a  
x.compareToIgnoreCase(a);//a=A  
x.equals(a);//wheter string a and x are same  
x.length();//number of character in string  
x.charAt(index:2);//help to reach to index of string  
x.trim();//remove extraspace from left and right side of text  
x.toUpperCase();//convert whole string to uppercase  
x.toLowerCase();//convert whole string to lowercase  
x.substring(beginIndex:2,endIndex:3);//between index [2 and 3]  
x.substring(beginIndex:2);//between index[2 to whole string]  
  
Character.toUpperCase(codePoint:2);//convert specific index of string to uppercase  
String t="23";  
int z=Integer.valueOf(t);//convert String to type int  
  
//STRINGBUILDER FUNCTION  
StringBuilder sb=new StringBuilder(str:"hello");  
sb.append(c:"a");//add 'a' to last of string  
Integer aa=10;  
aa.toString();//change object to string
```

```
//      12345678  
String name0="robertdo";  
System.out.println(name0.length());//8  
  
String name3="robert\ndo";  
System.out.println(name3.length());//9 \n =1 therefore 8+1=9  
  
String name="robert\n do";  
System.out.println(name.length());//10 \n=1 space =1 therefore 8+1+1=10  
  
String name2="robert \n do";  
System.out.println(name2.length());//11 \n=1 space =2 therefore 8+2+1=11
```

\n and \t==1

```
System.out.println('c'-'a');//3-1= 2  
System.out.println('a'-'b');//1-2=-1
```

```
int x,y,z;  
x=y=z=2;  
x+=y;x=4 ;           //x=x+y      x=2+2=4  
y-=z;y=0 ;           //y=y-z      y=2-2=0  
z/=(x+y);             //z=z/(x+y)   z=2/4=0  
System.out.println(x+ " +y+ " +z); // 4 0 0
```

Associativity of operator (* and / has both same important and rest bodmas rule is same)

```
int x=2,y=5;  
int exp1 =(x*y/x); //2*5/2=10/2=5  
int exp2=(x*(y/x)); //2*(5/2)=2*2=4
```


JAVA

```
class a{
    public static void printarray(int arr[]){
        for(int x:arr){
            System.out.print(x+" ");
        }
    }
    public static void bubblesort(int arr[]){
        for(int turn=1;turn<arr.length;turn++){
            for(int i=0;i<arr.length-2;i++){
                if(arr[i]>arr[i+1]){
                    int temp=arr[i];
                    arr[i]=arr[i+1];
                    arr[i+1]=temp;
                }
            }
        }
        printarray(arr);
    }
}
Run | Debug
public static void main(String args[]){
    int arr[]={9,8,7,6,5,4,3,2,1};
    System.out.println(x:"before bubble sort");
    printarray(arr);// 9 8 7 6 5 4 3 2 1
    System.out.println(x:"bubble sort");
    bubblesort(arr);// 1 2 3 4 5 6 7 8 9
    System.out.println(x:"after bubble sort");
    printarray(arr);// 1 2 3 4 5 6 7 8 9 (inspite the fact we use function while take value by call by value still it change the real array)
}
```

JAVA

Conditional Branching/Selectional Control/Decision Making

If Statement

```
if(condition_1){  
    //code to be executed  
}
```

if else Statement

```
if(condition_1){  
    //code to be executed  
}  
else{  
    //else code to be executed  
}
```

if only one statement

```
//if only one line of code under if  
if(i==arr.length-1) System.out.println(x:"3");  
  
//or  
  
if(i==3){  
    System.out.println(x:"3");  
}
```

Else if Statement

```
if(condition_1){  
    //code to be executed  
}  
  
else if(condition_2){  
    //else if code to be executed  
}  
  
else if(condition_3){  
    //another else if code to be executed  
}  
else{  
    //else code to be executed  
}
```

Nested if Statement

```
if(condition_1){  
    //code to be executed  
    if(condition_2){  
        //that code to be executed  
    }  
    else{  
        //else code to be executed  
    }  
}  
else{  
    if(condition_3){  
        //that code to be executed  
    }  
    else{  
        //else code to be executed  
    }  
}
```

Switch statement

```
int condition =3;  
char condition_2='c';  
  
switch(condition){  
    case 1://code 1  
        break;  
    case 2://code 2  
        break;  
    case 3://code 3  
        break;  
    default://if no case is matched with condition in switch  
}  
  
switch(condition_2){  
    case 'a'://code 1  
        break;  
    case 'b'://code 2  
        break;  
    case 'c'://code 3  
        break;  
    default://if no case is matched with condition in switch  
}
```

JAVA

Loop Statement

1)Exit Controlled Loop/post tested loop (do while loop)

```
do{
    //code to be executed

}while(condition);
```

2)Entry Controlled Loop/pre tested loop (for loop,for each loop, while loop)

<pre>for(int i=0;i<10;i++){ //code to be executed }</pre>	<pre>while(condition){ //code to be executed }</pre>
--	--

```
int i=0;
for(i=1;i<=7;i++){

}
System.out.println(i);
//8 because after 7 it is updaed to 7+1 which break condition become false
```

FOR EACH LOOP

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

//normal way to print array
for(int i=0;i<arr.length;i++){
    System.out.print(arr[i]+" ");
}

//another way to print array
for(int x:arr){
    System.out.print(x+" ");
}
```

```
//both are same
for(int i=0;i<10;i++){
    System.out.println(i);//0 1 2 3 4 5 6 7 8 9
}

for(int i=0;i<10;++i){
    System.out.println(i);//0 1 2 3 4 5 6 7 8 9
}
```

JAVA

Function

Here value is passed in function by call by value

```
class as {
    public static void sum(){
        System.out.println(x:"no parameter");
    }
    public static int sum(int x,int y){
        return x+y;
    }
    public static float sum(float x,float y){
        return x+y;
    }
    public static int sum(int x,int y,int z){
        return x+y+z;
    }
    public static boolean sum(int x){
        if (x>0){
            return true;
        }
        return false;
    }
    public static char sum(char c){
        c++;
        return 'a';
    }
    public static String t(String c){
        return c;
    }
}

Run | Debug
public static void main(String args[]){

}
```

Function overloading:

1. Parameter (same name different parameter)

```
public static int sum(int x,int y){
    return x+y;
}

public static int sum(int x,int y,int z){
    return x+y+z;
}
```

2. Datatype (same name but parameter datatype different)

```
public static int sum(int x,int y){
    return x+y;
}

public static float sum(float x,float y){
    return x+y;
}
```

JAVA

Array

For understanding only

1. `Int dim[row]`
2. `Int dim2[row][column]`
3. `Int dim3[depth][row][column]` (*generally not used but can be*)

One dimensional Array

```
//one way
int mark[]=new int[2];
mark[0]=76;
mark[1]=89;

//another way
int num[]={76,78};
// index : 0 1
//memory :1000 1004 (+4 because it is int)
```

Multidimensional Array

2d and 3d array and many more

```
//one way
int mark[][]=new int[2][2];
mark[0][0]=76;
mark[0][1]=89;
mark[1][0]=76;
mark[1][1]=89;

//another way
int num[][]= {{1,2} , {3,4}};
// index : 00 01 10 11
//memory :1000 1004 2000 2004(row wise)
//memory :1000 2000 1004 2004(column wise)
```

JAVA

OOPS: Class

- Object: Entities in the real-world (ex-p)
- Classes: Group of these entities (ex-pen)
- this keyword: It is used to use already define variable(ex-this.price)

Note: By convention class first letter of name is written in capital letter

This keyword

```
//this keyword uses
class Pen{
    int price=3;
    int getprice(){
        return this.price;//this.price tell price=3;
    }
    int setprice(int price){///price is new price and this.price=newprice
        this.price=price;
        return price;
    }
}
```

How constructor is calling (super keyword is used by java by default if not used)

```
class As{
    Run | Debug
    public static void main(String[] args){
        human h2=new human();
        // animal horse human
    }
}

class animal{
    animal(){
        System.out.println(x:"animal");
    }
}

class horse extends animal{
    horse(){
        System.out.println(x:"horse");
    }
}

class human extends horse{
    human(){
        System.out.println(x:"human");
    }
}

class As{
    Run | Debug
    public static void main(String args[]){
        horse h=new horse();
        // animal horse
    }
}

class animal{
    animal(){
        System.out.println(x:"animal");
    }
}

class horse extends animal{
    horse(){
        super();
        System.out.println(x:"horse");
    }
}
```

Static keyword: used to share the same variable or method of a given class ex-static property, static function, static blocks, static nested (class class A { class B{ } })

Super keyword: use to refer immediate parent class object

JAVA

Constructor

```
class As{
    Run | Debug
    public static void main(String args[]){
        Pen p=new Pen();//nonparameterized constructor
        System.out.println(p.company);//rolex
        Pen p2=new Pen(price:5);//parameterized constructor
        System.out.println(p2.price);//5
    }
}

class Pen{
    String company="rolex";
    int price;

    // non parameterized constructor
    Pen(){
        System.out.println(x:"nonparameterized constructor ");
    }
    //parameterized constructor
    Pen(int price){
        this.price=price;
        System.out.println(x:"parameterized constructor");
    }
}
```

Class vs object vs constructor

```
animal a =new animal();
//class object = new constructor
```

Shallow copying constructor

```
class As{
    Run | Debug
    public static void main (String args[]){
        Student s1=new Student();
        s1.marks[0]=36;
        s1.marks[1]=63;
        s1.marks[2]=34;
        Student s2= new Student(s1);
        s1.marks[0]=9;
        for(int i=0;i<3;i++){
            System.out.println(s2.marks[i]); // 9 63 34
        }
    }
}

class Student{
    int marks[];
    Student(){
        marks=new int[3];
    }
    Student(Student s1){
        marks=new int[3];
        this.marks=s1.marks;
    }
}
```

Deep copying constructor

```
class As{
    Run | Debug
    public static void main (String args[]){
        Student s1=new Student();
        s1.marks[0]=36;
        s1.marks[1]=63;
        s1.marks[2]=34;
        Student s2= new Student(s1);
        s1.marks[0]=9;
        for(int i=0;i<3;i++){
            System.out.println(s2.marks[i]); // 36 63 34 (it will not be 9 63 34)
        }
    }
}

class Student{
    int marks[];
    Student(){
        marks=new int[3];
    }
    Student(Student s1){
        marks=new int[3];
        for(int i=0;i<3;i++){
            this.marks[i]=s1.marks[i];
        }
    }
}
```

JAVA

1. Encapsulation (wrapping up data and methods under a single unit .it also implants data hiding using access specifier like default private protected)

```
class As{
    Run | Debug
    public static void main(String args[]){
        Pen p=new Pen();

        //private variable set and get
        System.out.println("price "+p.getPrice()); //knowing price
        p.setprice(newprice:13); //changing price value
        System.out.println("setprice "+p.getPrice()); //knowing price after set

        //default variable set and get
        System.out.println("color "+p.color); //knowing color
        p.color="grren"; //changing color
        System.out.println("new color "+p.color); //knowing color after set
    }
}

// defining class always below main class

class Pen{
    //default variable
    String color="yellow";
    //private variable
    private int price=3;
    //to get private variable value
    int getPrice(){
        return price;
    }
    int setprice(int newprice){
        price=newprice;
        return price;
    }
}
```

2. Inheritance (passing of property and methods from parent class (base class) to child class (derived class))

```
class As{
    Run | Debug
    public static void main(String[] args) {
        human f=new human();
        f.breath(); //yes although human class dont have any breath funcnion
        f.eat="food";
        System.out.println(f.eat); //food although human class dont have eat property
        fish h=new fish();
        System.out.println(h.live); //true from human property
        h.breath(); //yes from animal function
    }
}

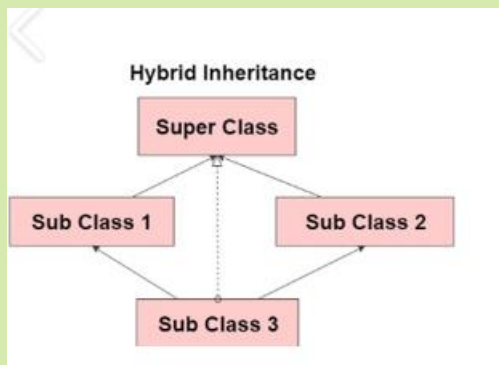
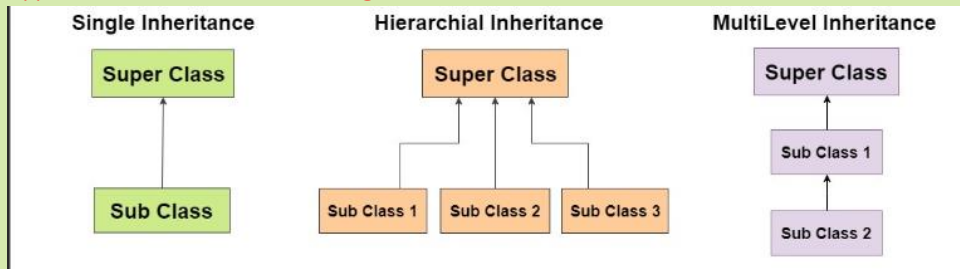
class animal{
    String eat;
    void breath(){
        System.out.println(x:"yes");
    };
}

class human extends animal{ //human get animal
    boolean live=true;
}

class fish extends human{ //fish get human so it also get animal
}
```


JAVA

Type of inheritance same logic will work as human extends animal



3. Runtime/dynamic polymorphism (many forms for function only)

```
class As{
Run | Debug
public static void main(String[] args) {
    animal a=new animal();
    human h=new human();
    fish f=new fish();
    a.eat();//food
    h.eat();//cooked food
    f.eat();//small fish
}
}

class animal{
    void eat(){
        System.out.println(x:"food");
    }
}

class human extends animal{
    void eat(){
        System.out.println(x:"cooked food");
    }
}

class fish extends human{
    void eat(){
        System.out.println(x:"small fish");
    }
}
```

JAVA

Variable shadowing

Same property in each parent and child class and its child classes

```
class As{
Run | Debug
public static void main(String[] args) {
    animal a=new animal();
    human h=new human();
    fish f=new fish();

    System.out.println(a.eat);//food
    System.out.println(h.eat);//plants and animal
    System.out.println(f.eat);//small fish
}
}
class animal{
    ⚡ String eat="food";
    void breath(){
        System.out.println(x:"yes");
    };
}
class human extends animal{
    String eat="plants and animal";
}
class fish extends human{
    String eat="small fish";
}
```

JAVA

4. Abstraction (hiding all unnecessary details and showing only important parts while encapsulation hide important data like password)

```
class As{
    Run | Debug
    public static void main(String[] args){
        // animal a=new animal();//itwill not work
        horse h= new horse();
        h.walk();
        human h2=new human();
        h2.walk();
    }
}

abstract class animal{
    abstract void walk();
}
class horse extends animal{
    void walk(){
        System.out.println(x:"walk on 4 leg");
    }
}
class human extends animal{
    void walk(){
        System.out.println(x:"walk on 2 leg");
    }
}
```

5.Interfaces (part of abstraction)

Total abstraction using interface

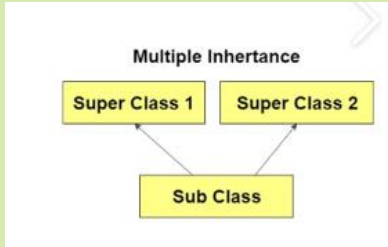
```
class As{
    Run | Debug
    public static void main(String[] args){

        queen q=new queen();
        q.moves();//queen moves
    }
}

interface chessplayer{
    void moves();
}
class queen implements chessplayer{
    public void moves(){
        System.out.println(x:"queen moves");
    }
}
```

JAVA

Multiple inheritance is not supported by java but can be implement using interface



```
class As {  
    Run | Debug  
    public static void main(String[] args) {  
        Bird myBird = new Bird();  
        myBird.speak(); // Outputs: Tweet tweet!  
        myBird.layEggs(); // Outputs: Laying eggs...  
    }  
}  
  
// Define an interface for speaking  
interface Speakable {  
    void speak();  
}  
  
// Define an interface for laying eggs  
interface EggLayer {  
    void layEggs();  
}  
  
// Implement the interfaces in a class  
class Bird implements Speakable, EggLayer {  
    public void speak() {  
        System.out.println(x:"Tweet tweet!");  
    }  
    public void layEggs() {  
        System.out.println(x:"Laying eggs...");  
    }  
}
```

JAVA

```
public class inheritance {  
    Run | Debug  
    public static void main(String[] args){  
        Vehicle obj1 = new Car();  
        obj1.print();  
        Vehicle obj2 = new Vehicle();  
        obj2.print();  
    }  
}  
  
class Vehicle{  
    void print(){  
        System.out.println("Base class(Vehicle)");  
    }  
}  
  
class Car extends Vehicle{  
    void print(){  
        System.out.println("Derived class(Car)");  
    }  
}
```

for overriding (Derived class (Car))
(Base class (Vehicle))

Override function will occur as print function in both vehicle and car

```
public class inheritance {  
    Run | Debug  
    public static void main(String[] args){  
        Vehicle obj1 = new Car();  
        obj1.print1();  
        Vehicle obj2 = new Vehicle();  
        obj2.print();  
    }  
}  
  
class Vehicle{  
    void print(){  
        System.out.println("Base class(Vehicle)");  
    }  
}  
  
class Car extends Vehicle{  
    void print1(){  
        System.out.println("Derived class(Car)");  
    }  
}
```

ERROR

error because vehicle only recognize print not print 1 which is present in car

JAVA

Recursion

1. Base case
2. Work
3. Inner case

```
class As{
    public static void rec(int x){
        //base case
        if(x==1){
            System.out.println(x+1);
            return;// we cant use break it is used in loop or in switch
        }
        //work
        System.out.println(x);
        //innercall
        rec(x-1);
    }
}
Run | Debug
public static void main(String args[]){
    int x=10;
    rec(x);
    //output 10 9 8 7 6 5 4 3 2 1
}
```

```
class As{
    public static void removeduplicate(String x,StringBuilder sb,int index,int map[]){
        if(index==x.length()){
            System.out.println(sb);
            return;
        }
        char curr=x.charAt(index);
        if(map[curr-'a']==1){
            removeduplicate(x, sb,index+1,map);
        }
        else{
            map[curr-'a']=1;
            removeduplicate(x, sb.append(curr),index+1,map);
        }
    }
}
Run | Debug
public static void main(String args[]){
    removeduplicate(x:"aababac", new StringBuilder(str:""), index:0,new int[26]);
    //abc
}
```

JAVA

Array list

- similar to array but with dynamic size where primitive type of data like int can't be stored directly. When we need extra space to store variable, in array list then it doubles the size

```
import java.util.ArrayList; // import

class As{
    //function call
    public static void arr(ArrayList<Integer> list,int x){
        //code to be executed
    }

    Run | Debug
    public static void main(String args[]){
        ArrayList<Integer> list=new ArrayList<>();///for storing integer (not int)
        ArrayList<Float> list2=new ArrayList<>();///for storing float
        ArrayList<Double> list3=new ArrayList<>();///for storing double
        ArrayList<String> list4=new ArrayList<>();///for storing string
        ArrayList<Byte> list5=new ArrayList<>();///for storing byte
        ArrayList<Short> list6=new ArrayList<>();///for storing short
        ArrayList<Character> list7=new ArrayList<>();///for storing character (not char)
        ArrayList<Long> list8=new ArrayList<>();///for storing long
        ArrayList<ArrayList<Integer>> list9=new ArrayList<>();///for storing arraylist for 2d array

        //operation element

        System.out.println(list);//
        list.add(e:2);// 2
        System.out.println(list);
        list.add(e:3);// 2 3
        System.out.println(list);
        list.add(e:8);// 2 3 8
        System.out.println(list);
        list.add(index:0,element:3);//3 2 3 8 (at index 0 ,3 is added nd rest is shift)
        System.out.println(list);
        list.remove(index:1);//3 3 8 (at index 1 2 is removed)
        System.out.println(list);
        list.set(index:0,element:4);//4 3 8 (at index 0 (3) is changed to 4)
        System.out.println(list);
        list.contains(o:2);//false
        System.out.println(list.contains(o:2));

    }
}
```

Input using scanner class

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

ArrayList<Integer> list =new ArrayList<>();

for(int i=0;i<10;i++){
    int x=sc.nextInt();
    list.add(x);
}
```

JAVA

Swapping logic (for updating list)

```
import java.util.ArrayList;
class As{
    Run | Debug
    public static void main(String args[]){
        ArrayList <Integer> list=new ArrayList<>();
        list.add(e:2);//0
        list.add(e:5);//1
        list.add(e:9);//2
        list.add(e:3);//3
        list.add(e:6);//4
        System.out.println(list);
        int one=1;
        int three =3;
        //swaping logic
        //we havent use becuse it will not update the value of list
        // int temp=p;
        // p=k;
        // k=temp
        //this logic will update the list
        int k=list.get(one);
        list.set(one,list.get(three));
        list.set(three,k);

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

Array list Function

```
import java.util.ArrayList; //for using arraylist
import java.util.Collections;//for using sorting function
class As{
    Run | Debug
    public static void main(String args[]){
        ArrayList <Integer> list=new ArrayList<>();
        list.add(e:2);//0
        list.add(e:5);//1
        list.add(e:9);//2
        list.add(e:3);//3
        list.add(e:6);//4

        //Sorting function Collections.sort(x); //ascending
        System.out.println(list);//[2, 5, 9, 3, 6]
        Collections.sort(list);

        //Sorting function Collections.sort(x,Collections.reverseOrder()); //descending
        System.out.println(list);//[2, 3, 5, 6, 9]
        Collections.sort(list,Collections.reverseOrder());
        System.out.println(list);//[9, 6, 5, 3, 2]
    }
}
```


JAVA

Multidimensional Array List

Ex-2d array list

```
import java.util.ArrayList;
class As{
    Run | Debug
    public static void main(String args[]){
        ArrayList<ArrayList<Integer>> Mainlist=new ArrayList<>();

        //creating list
        ArrayList<Integer> list1=new ArrayList<>();
        list1.add(e:2);
        list1.add(e:33);
        list1.add(e:22);
        list1.add(e:3);
        System.out.println(list1);// 2 33 22 3
        ArrayList<Integer> list2= new ArrayList<>();
        list2.add(e:21);
        list2.add(e:32);
        System.out.println(list2);//21 32
        ArrayList<Integer> list3=new ArrayList<>();
        list3.add(e:232);
        list3.add(e:3);
        System.out.println(list3);//232 3

        //adding list 1 2 3 to mainlist
        Mainlist.add(list1);
        Mainlist.add(list3);
        Mainlist.add(list2);

        System.out.println(Mainlist);// [[2, 33, 22, 3], [232, 3], [21, 32]]

        //printing array list
        for(int i=0;i<Mainlist.size();i++){
            for(int j=0;j<Mainlist.get(i).size();j++){
                System.out.print(Mainlist.get(i).get(j)+" ");
            }
            System.out.println();
        }
    }
}
```

Modulus function property

```
start=(start+1)%n; // 3 4 5 1 2
end=(end-1+n)%n; // 3 4 5 1 2
// end start
```

//it will allow to rotate whole start and end in that particular range
// if start reach to end then it will go to index 0 and if end go to
// 0 then it will reach to end index

```
// 3 4 5 1 2
// start end
```

JAVA

LinkedList (Collection of nodes)

Data type of address is node i.e. x555 y600 x700 y400

After defining node we need to connect them using a.next=b property



```
class As{
    public static class Node{ // we can use other variable name like node,vishal,x,y anything
        int data;
        Node next;           // we can change the variable name of next to next ,x ,y anything

        // Node(int x){
        //     data=x;
        // }

    }

    Run | Debug
    public static void main(String args[]){
        Node p1=new Node();
        p1.data=3;           //to void this we can use below in node class
                             //node(int x){data=x;} or node(int data){this.data=data;}
        Node p2=new Node();
        p2.data=5;

        System.out.println(p1.data);//3
        System.out.println(p2.data);//5
        System.out.println(p1.next);//null
        System.out.println(p2.next);//null

        // System.out.println(p1.next.data);//will not work error

        System.out.println(p1);//As$Node@12f40c25
        System.out.println(p2);//As$Node@3ada9e37

        // 3 5 (not connected)
        p1.next=p2; // 3->5 (connecting p1 to p2) means p1.next=p2
                             // p1.next=As$Node@3ada9e37
        System.out.println(p1.next);//As$Node@2f92e0f4
        System.out.println(p2);//As$Node@2f92e0f4
        System.out.println(p1.next.data);//5 data of p2
    }
}
```

In function public static void display (Node x) {}

```
class As{
    public static class Node{
        int data;
        Node next;
    }

    public static void display(Node x) {
        Node k=x;
        //using for loop when iteration is known
        for(int i=0;i<4;i++){
            System.out.print(k.data+" ");//5 3 8 9
            k=k.next;
        }
    }
}
```

```
Run | Debug
public static void main(String args[]){
    Node p1=new Node();
    p1.data=5;
    Node p2=new Node();
    p2.data=3;
    Node p3=new Node();
    p3.data=8;
    Node p4=new Node();
    p4.data=9;
}
```

```
class As{
    public static class Node{
        int data;
        Node next;
    }

    Run | Debug
    public static void main(String args[]){
        Node p1=new Node();
        p1.data=5;
        Node p2=new Node();
        p2.data=3;
        Node p3=new Node();
        p3.data=8;
        Node p4=new Node();
        p4.data=9;
        //5 3 8 9
        p1.next=p2; //5 --> 3 8 9
        p2.next=p3; //5 --> 3 --> 8 9
        p3.next=p4; //5 --> 3 --> 8 --> 9

        System.out.println(p1.data);//5
        System.out.println(p1.next.data);//3
        System.out.println(p1.next.next.data);//8
        System.out.println(p2.next.data);//8 another way
        System.out.println(p1.next.next.next.data);//9
        System.out.println(p3.next.data);//9 another way

        Node k=p1; //we use other node data type k1 to store header other wise header will be changed
        //using for loop when iteration is known
        for(int i=0;i<4;i++){
            System.out.print(k.data+" ");//5 3 8 9
            k=k.next;
        }

        //using while loop when iteration is not known
        System.out.println(p4.next);//null we will use this concept
        Node k1=p1;
        while(k1!=null){
            System.out.print(k1.data+" ");//5 3 8 9
            k1=k1.next;
        }
    }
}
```

JAVA