19/08/19

Core Java.

Sun Microsystems - in 1995 James Gosling

oak - Before named as

Java - Now

* JAVA SE :- Java standared edition.
* JAVA ME : Java Micro edition - only for objects small e devices (calculator)
* JAVA EE :- Java enterprise edition.

⇒ when we have to develop any application we can use and we are using JAVA Micro edition.

* standalone application
⇒ we have to install the app in our own device

* Webapplication
⇒ we can share application

⇒ To develop standalone application we are using Java SE

⇒ We are developing an application for business then it is called as enterprise edition.

Java standared Edition :-

Java is second most popular programming language

⇒ Java is platform Indepedndant programming language. once we written the code and compiled and then we can use that program in any operating System.

Methods :- A Java method is a collection of statements that are grouped together to perform an operation.

Java Runtime Environment :- (It is a kind of application)

source file.
It consists of Source code called lines of code

Src.js
Src.java.

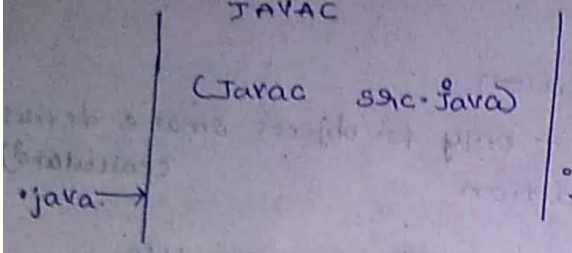
⇒ Java source file consists of Java Source code onece we written we can use anywhere

⇒ If we want to save that source code it extends with .Java (extension .Java) ⇒ It becomes Java source file.

To compile sourcefile.

JAVA COMPILER :-

⇒ compiler checks for any syntax Errors (Rules).



⇒ Java compiler compiles the Java. src.java extension file for the syntax errors before the execution

⇒ It checks for Rules and Syntantical errors.

⇒ After the compilation of src.java we get an output file called .class

→ .java is an Input and .class is an output.

Java Virtual Machine.

Java JVM enables the JAVA to compile and toget the result

Java has two parts:-

Declaration :- what is the program

Definition :- Body of the program (with that what we have to do.

* Declaration {

Definition

}.

Java consists of.

classes

Interfaces

Classes :- This is Java source. it consists.

public class class-name {

Statements ---

statements ---

}

. JRE interpreter

Object:-

⇒ objects are elements of a program that has some data which is also known as states

⇒ objects also have behaviours which needs they can perform certain operations

⇒ those behaviours in Java is called as Methods.

⇒ we can store some data of that perticular object.

Java is a object oriented programming languag.

object

object have some states

state represents the data.

Behaviour represents the Methods.

Keywords:-

predefined meaning in the programming language

{Java is a case sentitive language}

⇒ Java has 50 keywords
or 53
literals can also be classified as keywords

True.

False.

⇒ Actually we have 50 keywords and three are the literals. it has also some predefined meaning in the programming language.

Identifiers:- Identifier is a name given to (elements) or class. of a program

Ex: HelloJava is a Identendifier

⇒ while creating a Identifier Rules to be declared:

⇒ Always starts with Alphabates

⇒ Always it can not start with a digit.

⇒ It start with alphabates After that it can have n number character after the character

⇒ Identifiers are also ~~known~~ as case sensitive.

⇒ Keywords cannot be Identifiers.

⇒ The only special characters are allow for the creation of Identifiers. i.e, $ and _ other than it will not support

Identifies :- Names given to components / elements of a program.

⇒ Identifiers will not allow spaces.

cls ⇒ clear screen.

D:\core Java> javac AA.java.

⇒ Identifies cannot start with digit. But after one character it will allows n number of digits

⇒ Variables :-

If we want to execute java program we can go for JRE

⇒ If we want to develop an application we go for JD

variable in JAVA :- It is a peace of memory.

variable is a name given to a memory.

⇒ variable can have data, value in the variable

⇒ whenever we want to store some value, or data. we go for variable

⇒ every variable should have datatype compulsory

Data Type :- Data type will specify the type of data has to store in the memory.

⇒ Java some times called as strongly typed language.

1. premitive
2. Reference

premitive Datatypes :-

8 primitive Datatypes (Numeric)

6 digits (char boolean)

2.

① **Byte** :- (1byte/8bits)

It can have numerico values

```
class Demo{
public static void main(String[]args){
    byte b = +127;   (127)  (-128) error
                     lowest range
    System.out.println(b);
}
}
```

127 + 128 / 255

(-128 to 127)

2) short :- Numeric datatype (2 bytes)

{

byte b = -127;

short s = -37555;

}

32768

short s = +32768;

short s = -32767

(-32767 to 32768)

3) int :- 4 bytes. (32 bits

~~(65~~ (-65536 to 65535)

4) long :- 8 bytes (64 bits)

(-9 billion to +9 billions)

5) double (8 bytes / 64 bits)

6) float (4 bytes)

{

float f = 10.123456789f;

double d = 10.123456789;

}

7) char :- (2 byte)

{ char c = 'A';

char c1 = 'AB';

}

It can have only one character.

// Delaring a boolean datatype variable trying to store a boolean type.

Boolean :-

It can accept either true or false.

True - 1

False - 0

{

boolean b = true;

} system.out.println(b);

Declaration and Initialization :- 20/08/18

declare
Initialization

```
Datatype var;  // Delaration part
var = value;  // Initialization part.
System.out.println(var);  // utilization.
```

Operators :-

⇒ An operator is a special symbol or a keyword. that is used to designate a mathematical operation or some other type of operation)

⇒ These operations can be performed on one or more than one values. called as operands.

Arithmetic operators :-

+ } additive arithmatic operators
-

(1) + (-1)
1 - 1
0

* } Multiplication arithmetic operators
/ } Multiplicative.
% }

4 % 2

$\frac{4}{2} = \frac{4}{1} * \frac{1}{2}$.

Compound operators (compound assignment operators)

Combination of arithmatic and assignment operater.

```
i = 10;
j = 20;
j += i;  => 30 => j
j -= i;  => 20 => j
j *= i;  => 200 => j
j /= i;  => 20 => j
j %= i;  => 0 = j
```

Increment operator :- (unitary operators) Decrement operator :-

(++) - Increment

(--) - Decrement

Operators :-

i) Arithmetic
ii) Assignment
iii) Increment and Decrement

++a pre increment.

int f = 10;

++ f; // it will increment by one unit.

(i) Increment.
(ii) Substitute.
(iii) Utilize.

post Increment.

we will be using the ++ after the operand.

int f = 10;

f ++ f

(i) substitute. f = f++ = 10
(ii) utilization. f = 10
(iii) Increment. f = 11

⇒ we can give the operator Before the operand or after the operand.

→ Before means pre Increment.

→ After means post Increment.

4) Relational operators :— comparision operations (operators)

By comparing we are executing relational operators

i) == ⇒ returns true when both signs of the equations equals then true otherwise false

ii) != ⇒ not equal. returns true when both the equations are equal.

iii) <

iv) >

v) >=

vi) <=

i) < ⇒ returns true if the left side of the equation is less than & right side of the equation

ii) > ⇒ returns true if the left side of the equation is greater than right side of the equation

iii) >= ⇒ returns true if the left side of the equations is greater than or equal to the right side of the equation

iv) <= ⇒ returns true if the left side of the equation is less than (or) or equal to the right side of the equation

⇒ It will returns Boolean output like true or false.

(5) Logical operators :-

when ever we want to check two or more conditions we go for logical operators

→ logical operators are using with Relational operators

⇒ It will return either true or false.

1) Conditional AND (Logical AND)
2) conditional OR (conditional OR)

1) NOT ! (unary operator)
2) && (AND)
3) OR (||)

Here we are using double two operators

NOT ! :-

It is a unary operator it will return false if the right hand side of the operator is True.

AND :- And is not a unary operator

⇒ If the one condition is false it wont go for second condition it will return false.

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR :-

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

⇒ If the first condition is true then it will not go for second condition it will return True. otherwise false.

Bitwise operator :- [single AND operator]

& AND Bit by Bit ie. it will execute and it will return boolean.

Operation					
&	0	0	1	0	
	0	0	1	1	
	0	0	1	0	2

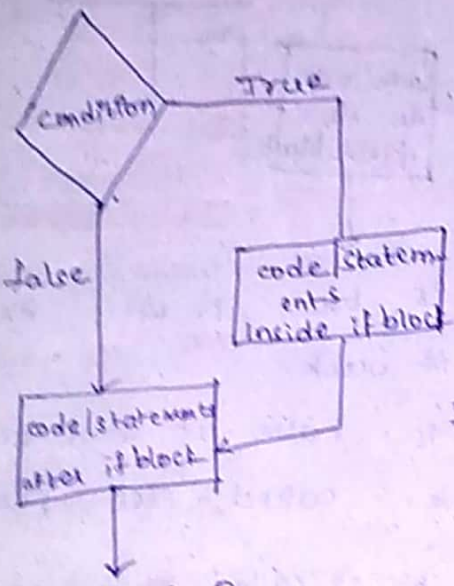
Operation					
	0	0	1	0	
|	0	0	1	1	
	0	0	1	1	3

Control Statements

If we want to execute certain lines of code based on certain conditions then we go for control statement.

→ With the help of control flow statements we are controlling the flow of statements.

If :- control Flow statements

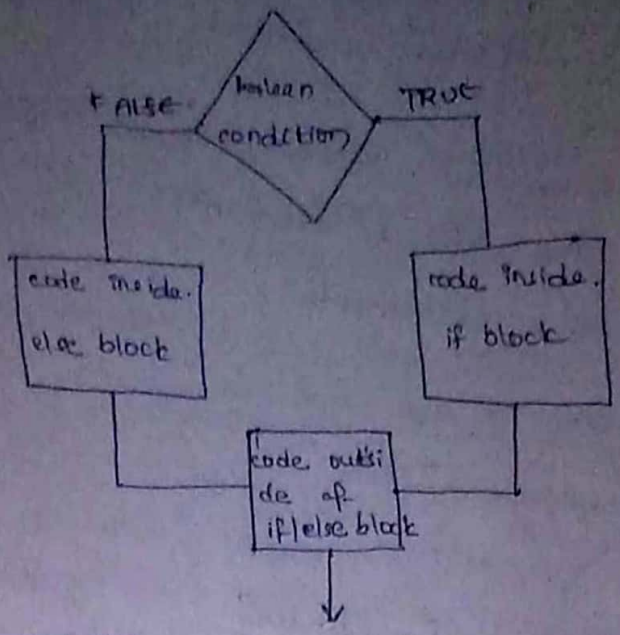


If the condition is true this will give the result

```
if(boolean condition){
    statements---
    statements---
}
public class Ifstatement{
psvm(String[] args){
int i=10;
int j=20;
if(i<j){ ⇒ This condition is true
s.o.pln("Inside of block");
}
sy.o.pln("outside of block");
}
}
```

If we want to stasfy one Boolean condition we are

* Inside of the If else If-else



⇒ If condition is true it will execute the inside of the if block

* If the condition is False it will execute the outside of the block. called else part.

⇒ It is also some times called as else if Ladder.

⇒ if-elseif ladder :-

⇒ we go for elseif when we have more than one boolean condition to be satisfy.

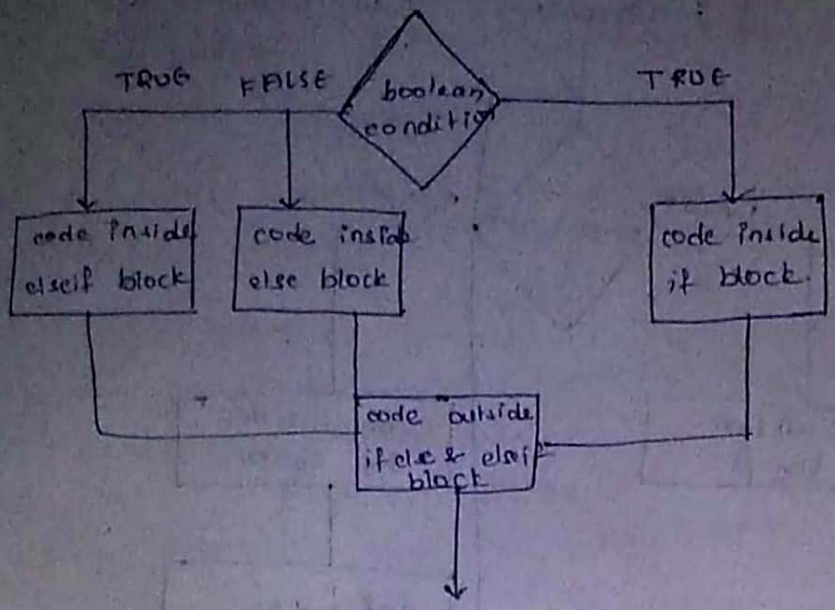
⇒ when ever we have multiple conditions we go for elseif condition.

1 if

or else

elseif block we can have n number of times

⇒ we can write n number of block.

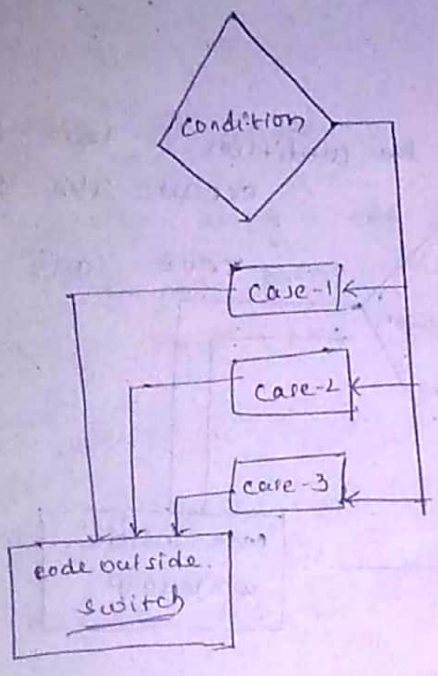


switch

⇒ switch case is used to check the equal condition.

If we want to check equal condition {>=, <=} wont work of

```
{
```




```
break;
}
{
}.
```

⇒ If we have perform same operation. using . break is default.

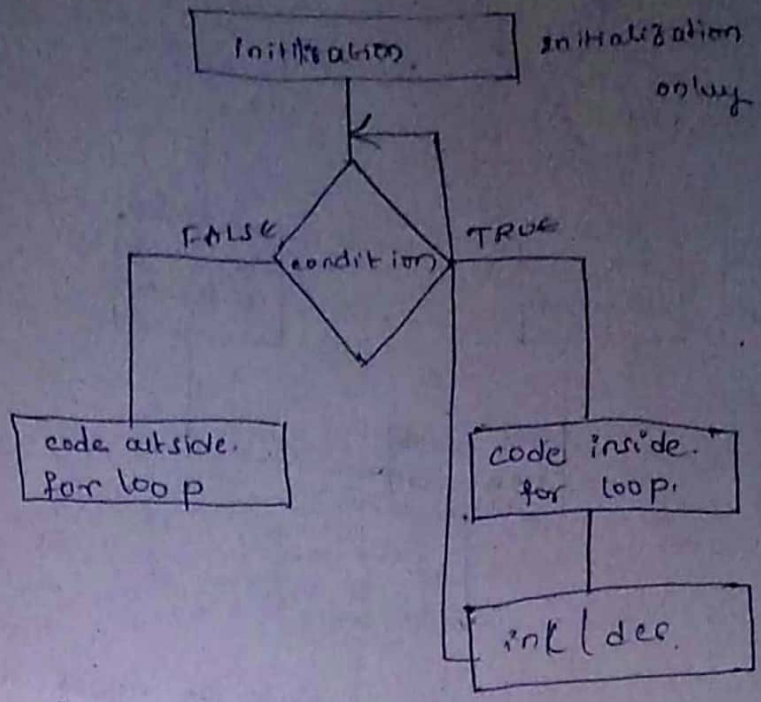
Loops or Looping statements :-

⇒ Starting ending points all the same

⇒ Technical meaning :- Instead of repeating the code again and again we are using looping condition.

syntax :- for (initialization; test condition; increment/decrement)

```
{
  statents---;
}  statements---;
```

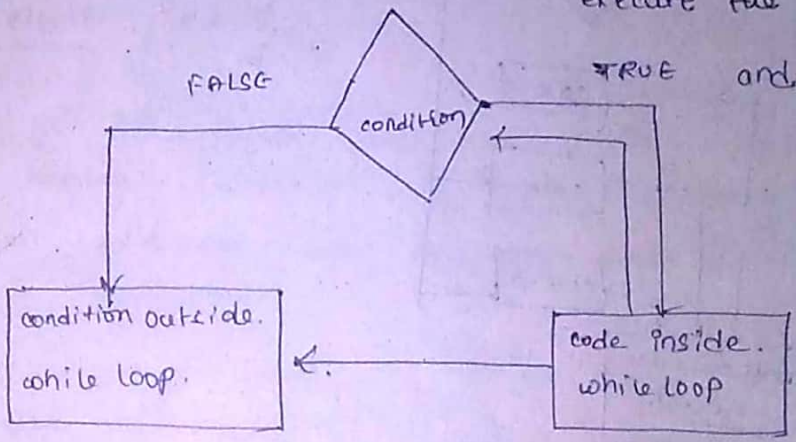


Initialization will happen only once.

Iii) while :-

```
while (condition) {
    statements ---.
    statements ---.
}
```

If the condition is true it will execute the ~~for~~ while loop. and exe.



⇒ Initialization should be done in outside the whileloop If we want to Inc/Dec we have to write inside the loop

```
int i=1;
while (i<100) {
    s.o.p in(i);
    i++;
}
i=?
```

iii) do-while

certain kind of program is executed after that the condition is checked.

⇒ It will displays at least one output before condition checks.

Syntax:- do{

statements

statements

} while (condition)

If we want to check the condition is true or false.

we go for do-while.

for each loop:- advanced loop

arrays
collection
iterations

we are using for these

write a program using CF statements and looping statements where the users can print his name onces if the days of the week, (1) weekend 5 times
week 1 time.

(iv) for-each: (Nested for loop)

```
for (int i=1; i<=4; i++)
{
    for (int j=1; j<=i; j++)
    {
        s.o.p("*");
    }
    s.o.println();
}
```

i=1; 1<=4 ✓
j=1; 1<=1
j=2; 2<=1 ×
i=2; 2<=4
j=1; 1<=2 ✓
j=2; 2<=2

int k=4
k=3
k=2

```
*
* *
```

i=1; 1<=4 ✓
j=1; 1<=4 ✓
j=2; 2<=4 ✓
j=3; 3<=4 ✓

```
int k=4;
for(int i=1; i<=4; i++)
{
  for (int j=1; j<=k; j++)
  {
    System.out.print("*");
  }
  System.out.println();
  k--;
}
```

```
* * * *
* * *
* *
*
```

i=1; i<=4;
j=1; 1<=4 ✓
j=2; 2<=4 ✓
j=3; 3<=4 ✓
j=4; 4<=4 ✓
j=5; 5<=4 ✗

k=4

k=2
i=3; 3<=4 ✓
j=1; 1<=2 ✓
j=2; 2<=2 ✓
j=3; 3<=2 ✗

k=3
i=2; 2<=4 ✓
j=1; 1<=3 ✓
j=2; 2<=3 ✓
j=3; 3<=3 ✓
j=4; 4<=4 ✗

```
* * * *
* * *
* *
*
```

k=1
i=4; 4<=4 ✓
j=1; 1<=1 ✓
j=2; 2<=1 ✗

using for loop print

Assignment

```
1
2
3
4
```

```
1
2  5
3  6  8
4  7  9  10
```

(ii)
```
A a
B b  C c
D d  E e  F f
```

Assignment

```
1
2  5
3  6  8
4  7  9  10
```

Assignment

```
    *
  * * *
* * * * *
  * * *
    *
```

(iv)

A

B C D

E F G H I

1 2 3

4

i=4

i=1 ; i<=4 ✓

j=1 ; 1<=1 ✓

j=2 ; 2<=1 ✗

```
*
* *
* * *
* * * *
```

```
int k=1;
for (int i=1; i<=4; i++)
{
    for (int j=1; j<=k; j++)
    {
        s.o.p("*");
    }
    s.o.p ln();
    k++;
}
```

k=1

i=1 ; i<=4 ✓

j=1 ; j<=1 ✓

j=2 ; 2<=1 ✗

k=2

i=2 ; i<=4 ✓

j=1 ; 1<=2 ✓

j=2 ; 2<=2 ✓

j=3 ; 3<=2 ✗

k=3

i=3 ; 3<=4 ✓

j=1 ; 1<=3 ✓

j=2 ; 2<=3 ✓

j=3 ; 3<=3 ✓

j=4 ; 4<=3 ✗

k=4

i=4 ; 4<=4 ✓

j=1 ; 1<=4 ✓

j=2 ; 2<=4 ✓

j=3 ; 3<=4 ✓

j=4 ; 4<=4 ✓

j=5 ; 5<=4 ✗

Methods

dec{   Dec   Access specifier   Access modifiers   Return type   method name (arglist);

dif.

}.

DEFI

statements...

statements...

} return statement;

It just tell us the visibility of any method.

If we are using access. specifier with a variable

Access specifier specify a visibity of any code component.

Access modifiers :-

⇒ Access modifier speicifier that method, a perticular class belongs to perticular object or component.

class

Return type :-

Return type can be any of primitive Datatypes. Or it can wide or it can be any of the Reference types

⇒ A Method can perform perticular task

⇒ Method delaration is compusory.

⇒ Method is a named block of code.

* when we are give return type other than wide we should be mention return statement in the inside of the body in the last Line

⇒ Method will always start ~~to~~ a small alphabet

Java coding convention

⇒ Method name will always start with small.

* After method ()

⇒ A method may or may not return any value.

⇒ A Method will only execute if it is been called by passing the required arguments.

⇒ A method which is being called is known as called method

⇒ A Method which is calling another method is known as calling method

⇒ when ever we need to call method we need proper arguments . it can be null, datatype, reference

⇒ We can create Method inside the class

⇒ If we try to create within or inside the method it will cause problem.

New

Java project

or

others (window)

Java project

coin

Make a folder in c drive & c dave.

open desktop.

project name ⇒ UST Global.

finish

open perspective

Java.

pattern Daimond

Integers

Alphabets

public class Alphabets {

public static void main (String [] args) {

int a=65, b=97;

to.

COM.dev

control + N

com.dev.methods

Method Example

```
public class Integers4 {
public static void main(String [] args) {
 int rows = 4;
for(int i=1; i<=rows; i++)
{
  int num = i;
  for(int j=1; j<=i; j++)
  {
   System.out.print(num+" ");
   num= num+rows-j;
  }
   System.out.println();
  }
 }
}
```

```
1
2 5
3 6 8
4 7 9
```

i=1 ; 1<=4 ✓

j=1 ; 1<=1 ✓

j=2 ; 2<=1 ✗

1+4-2

2.

i=2 ; 2<=4 ✓

j=1 ; 1<=2 ✗

i=1 ; i<=4 ✓

n=1

j=1 ; j<=1 ; j++

1+" "

n = 1+4-1=4

i=2 ; 2<=4

n=2

j=i ; i<=2

2+" "

n= 2+4-1=5

i=2 ; 2<=2

5+" "

n=7 5+4-2

3<=4

n=3

n= 9+4-3= 10

j=1 ; 1<=3

3+" "

n=3+4-6=6

i=2 ; 2<=3

6+" "

i=3 3<=3 ✓

9+" "

n= 9+4-3=10

i=4 4<=4

n=4

i=1 1<=4

4+" "

n=4+4-1=7

i=2 2<=4

7+" "

3<=4

7 9 10

+4-2

```
pabuc class Integers {
public static void main (string [] args) {
 int rows = 4;
for(int i=1; i<=rows; i++)
{
  int num = i;
  for (int j=1; j<=i; j++)
  {
   System.out.print(num+" ");
   num= num + rows - j;
  }
   System.out.println();
  }
 }
}
```

1
2 5
3 6 8
4 7 9

i=1 ; 1<=4 ✓

j=1 ; 1<=1 ✓

j=2 ; 2<=1 X

1+4-2

2.

i=2 ; 2<=4 ✓

j=1 ; 1<=2 X

i=1 ; i<=4 ✓

n=1

j=1 ; j<=1 ; j++

1+" "

n = 1+4-1=4

i=2 ; 2<=4

n=2

j=i ; i<=2

2+" "

n= 2+4-1=5

i=2 ; 2<=2

5+" "

n=7 5+4-2

3<=4

n=3

n= 9+4-3=10

j=1 ; 1<=3

3+" "

n=3+4-6=5

i=2 ; 2<=3

6+" "

i=3 3<=3 ✓

9+" "

n= 9+4-3=10

i=4 4<=4

n=4

i=1 1<=4

4+" "

n=4+4-1=7

7 9 10

i=2

2<=4

7+" "

3<=4

=7+4-2

n=5

i=1; i<=5;

sp=n-i 4 3 2 1 0

st=1 3 5 7 9

— * —

i=1; i<=6;

j=1; j<=3; j++

k=1; k<=1;

i=1; i<=5;

```
int sp=4;
int st=1;
for (int i=1; i<=5; i++)
{
    for (int j=1; j<=sp; j++)
    {
        System.out.print(" ");
    }
    for (int k=1; k<=st; k++)
    {
        s.o.p("*");
    }
    s.o.pln();
    if (i<=2)
    {
        sp--;
        st+=2;
    }
    else
    {
        sp++;
        st-=2;
    }
```

control + A

control + I

It makes proper alignment

project explorer

Run as Java application.

Arrays

Array is an group of homogeneous data that has some index with and a fixed size

Index of an array will always start from zero.

⇒ Declaration of array.

creating arrays.

datatype [] arrayname;

datatype[] arrayname;

datatype arrayname[];

creation

intArr = new int(size)

Initialization

intArr[index] = value;

length is a variable in array that gives size of an array.

char[] chr = new char[10];

char ch;

char[] ch; //size-1 = index

String :-

char c = 'A';

char[] c = new char[15];

String str = " ";

⇒ String is a sequence of characters.

⇒ If we want to change a value for the variable we have to use Immutable.

String is a user defined code.

String str = "Hello";

↳ reference

⇒ the datatypes maybe programmer defined. or

⇒ user defined or allread in the API, we are just using the reference variable for that to store the string data.

⇒ sequence of characters.

inside the string we are using character Array

⇒ If we want to create new string of a variable.

String str = new String("Hello");

creating a new variable using new keyword.

without creating new keyword.

String str = "Hello"; ⇒ String str; str = "Hello

⇒ Instead of storing inside a character.

we are storing in the String

⇒ S.charAt is a method. which returns value

character of the length. com.dev.Strings

creating strings.

It should be String

Not null

String Methods:-

First Method is the beginning index.

2nd Method is the ending index.

```
String a = str.substring(3);
S.O.pln("output for substring():"+a);
String q = str.substring(3,1);
S.O.pln("output for substring(int,int):"+q);
```

⇒ when ever we are creating a new Method we have to create variable.

Reference type:- 06/08/19

⇒ A Reference type is a type that is based on class rather than a primitive datatype.

⇒ The Reference type can be based on classes in Java or classes defined by programmer or developer. / predefined

⇒ new keyword = creating new object

String str = " ";

String str1 = new String(" ");

new object of string may be created.

⇒ It will store address of the object that has been created

⇒ It is the physical address that we have created.

* The string stores the particular physical address of the particular variable.

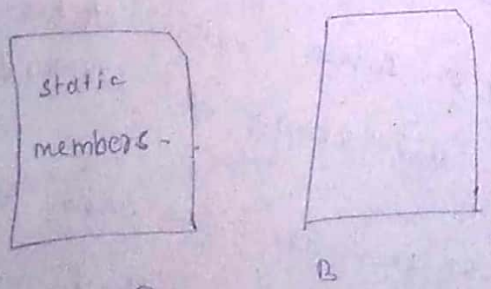
char a = new char[5];

Static and Nonstatic:-

Static:- Any member of the class that has been declared with keyword static

it is a static member.

⇒



A B

⇒ If we want to use static methods in any other class no need to create object - for the class which is having static members, directly we can use by using the . operator

class name

⇒ If we are using non-static members in another class we have to create an new object.
in the another class.

⇒ members which does not have static called as non static.

→ static member it
If a member is static it is belongs to class
If a member is non-static belongs object of that particular class

⇒ If we want use nonstatic members in the same class we have to create new object

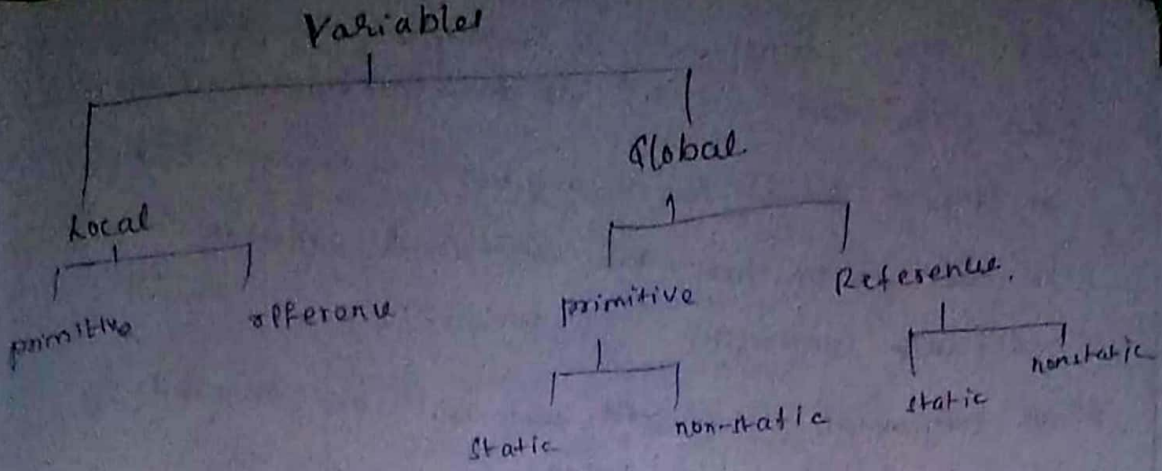
⇒ If we want to use static members in the same class no need to create new object we can call directly by the method name.

⇒ Inside any method whenever we are tapping to create static it is not possible

⇒ we can create same data and same name variable.
Inside the class and outside the class

⇒ If we want create same name data types they are inside the scope of method.

⇒ Method can be we can use the area where it has been declared in the method not in any where.



⇒ It does not have return type.

⇒ the name of the method is same as the class name.

⇒ If we create a class and compile the compiler will create their own method. with the above feature.

public class test{

}

Constructor :-

⇒ Constructor is a special type of ~~not~~ method.

constructor should name should be class name.

⇒ It does not have any return type.

Constructor is similar to method not actually a method

⇒ when we are trying to call a method we trying to invoke method.

⇒ with the help of constructor we can create objects

⇒ A constructor can have zero arguments or n number of arguments as well as methods

⇒ The constructor every class should and must have constructor.

⇒ If we are not creating constructor the compiler will create constructor.

⇒ It is called default constructor.

⇒ If the constructor is taking any input data. then we can call it as parameterized constructor

it doesn't have any argument, statements, body in the constructor, the constructor called as default constructor

⇒ no argument constructor will be created by developer (it contain body)

⇒ default constructor is created by compiler. In body (nothing will be written in body part)

⇒ The no argument constructor will have body inside the constructor.

→ name of the constructor are same i/p is different

constructor overloading.

Constructor overloading :-

⇒ we are trying to create same constructor with different arguments

⇒ 1. no-arg.

⇒ int

⇒ string.

→ either no. of input arg.

⇒ either type of data

⇒ ~~no~~ no argument

⇒ There is no order of ~~datatype~~ input argue

default
parameterized
no-arg/argument
} 3 type of constructor

Access specifiers :-

1. public : controls the visibility of the member
2. protected :
3. default : also known as package level
4. private.

⇒ when a member is declared with public specifier it can be used by any other class

⇒ when a member is declared with protected it cannot be accessed by any package.

Default (or) package level :-

we cannot use methods in some other package.

private :- we can use particular member with in the class.

Association :- Relation between two classes

⇒ Association in JAVA is relationship between two different classes.

can be of different types

1) one to one.
2) one to many.
3) many to many.
4) many to one

In JAVA we can have 2 different types of relationships

1) has a
2) is a

has a :-

i) aggregation

ii) composition.

has a:-

1. aggregation
2. composition

aggregation

(a) To classes will be related to each other not closely

⇒ exists of one class will not depends on the another class

Ex:- Students
Trainers
this classroom

2) composition:- is also a kind of a has a relationship existe of one class will depends on the existence on another class

is a relation:- is also known as Inheritance.

Inheritance:- ** Imp

Acquiring properties from some one is called inheritance

⇒ The process by which one class acquires the properties and functionalities of another class is called as Inheritance.

⇒ we can inheritance either functionalities or properties.

⇒ The aim of inheritance is to provide reusability of code and so that

⇒ whose a class whose properties or functionalities are being inherited by some other class is known as parent class or super class or base class.

⇒ The class that inherits properties (a) or functionalities from another class is known as child class or subclass or derived class

⇒ we are using inheritance to reduce the code

⇒ To reduce lines of code.

⇒ For Inheritance in JAVA we have to make use of extends keyword.

⇒ We can access the properties of both superclass and subclass. using object of subclass.

⇒ Final class can not be inherited; Final members of a superclass can be inherited. but can not be changed.

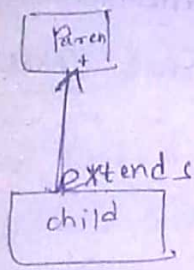
⇒ private members and constructors of superclass cannot be inherited.

⇒ Types of Inheritance.

1. single Inheritance :-

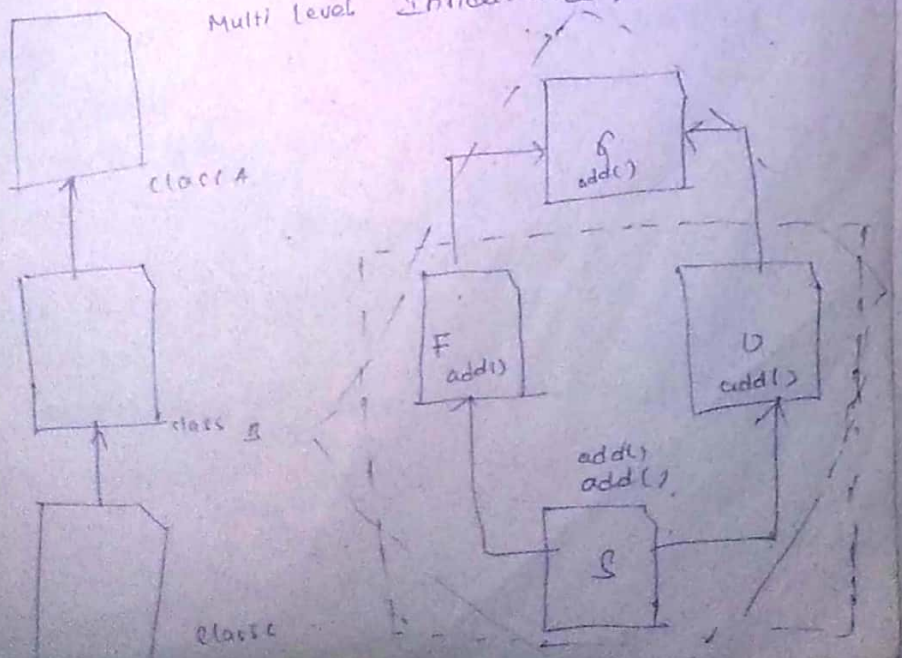
⇒ only one class acquiring properties from the another class then it is called as single Inheritance.

Ex :-



Multilevel Inheritance - some what like single Inheritance.

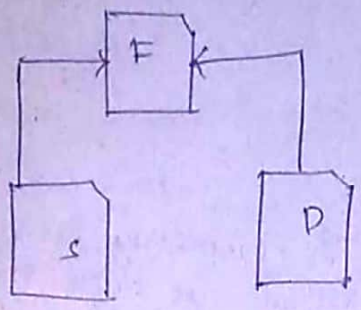
Diamond problem



⇒ Multiple Inheritance is not possible in Java. Ref of diamond p
supported.
* The compiles gets confused where ? need to get the acquire from either

Hierarchial Inheritance-

when even two more Classes acquiring the properties of single class is known as Hierarchial Inheritance.



Hybrid Inheritance is a combination of any type of Inheritance either Multilevel Inheritance.

superclass

The super keyword refers to an objects of Immeadicate parent or superclass

=> To access the datamember of a parent class when both child and parent class have the data member with the same Name

=> If sub class constructor does not contain super keyword then java compile implicitly adds super keyword in the constructor.

=> java compiler implicitly adds super keyword when the super class contains no-arg constructor otherwise we have to pass the super() explicitly in subclass

Method overriding :-

=> Defining a Method in a child class which has already been declared in parent class is known as Method overridding.

=> Method overriding is done to provide implementation specific to a child class.

Advantages of Method overriding :-

=> The advantage of Method overriding is that you can provide implementation to the child class method without changing the code present in the parentclass.

Final members can be overridden.

=> we cannot overridden private methods that are present in superclass.

Method Overloading:-

=> write a method in single class.

method overloading is feature in java

=> that allowes to have same methods (same thing) in a class more than once.

provided the arguments ~~this~~ list differ

1. No. of characters
2. Order of characters
3. Data types of characters

=> more than one method also can be specified in a single class.

=> same name with two same method in a single class.

=> that allows same ~~that~~ methods with the same name in a single class.

write a program that performs basic arithmetic operations +, -, *, /, having different argument list.

polymorphism :-

+ single entity is working differently for different classes

⇒ print(Name) method consists of all the classes are the working is different

⇒ single entity acts differently for different objects.

- polymorphism : late / Binding.

+ Method overriding is a type of polymorphism

+ Method overriding is an example Runtime polymorphism

⇒ In Method overriding Binding is done at Runtime.

⇒ compile time polymorphism :- (Method overloading)

| late Binding
Early Binding.

add (int ?, int j)

add (int ?, int j, int k)

add (int ?, int j, int k, int l)

particular method which has two i/p arguments it tries to binding then and there.

⇒ when we call method which has two arguments at that time that add method will be executing. the functioning of all the methods is same but with different arguments

⇒ Binding is done at compile time. compiletime.

Abstraction :- providing functionality and hidding the details is known as abstraction

Abstraction is about not need of explain the main method. It explains about the method what we have invoked.

⇒ hide all the implementation and gives the result. what we invoked.

Abstract class

⇒ Any class that has been declared with abstract keyword is called as Abstract class

⇒ Any method that has been declared with an abstract keyword that is known as abstract Method.

⇒ Abstract method does not have a body.

⇒ abstract method do not have definition.

⇒ abstract class is can have both abstract methods as well as concrete methods.

⇒ Any class having an abstract method should be declared as abstract. but Viceversa not

⇒ It is not mandatory to have a class but have concreate method

⇒ If a class have been declared as a abstract class than that class should be extended by any child class or subclass.

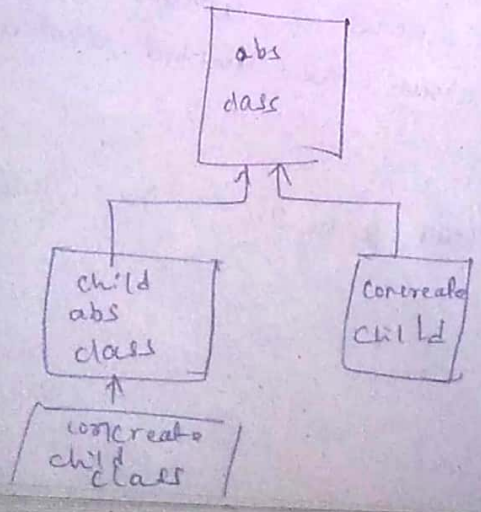
⇒ we can not create objects for abstractclass

⇒ we can create constructor for abstractclass.

⇒ If have to override all the abstract method.

⇒ we can achieve 0 to 100 per using abstract class

⇒



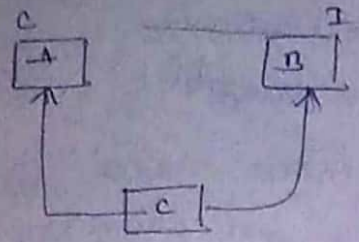
Interface

=> It is just like an abstract class

=> where we can declare different Methods

=> A class extends two class

=> A class Implement the Interface. <<B>>



<<B>>

static

default

=> ~~when~~ a ~~we~~ ~~create~~ a.

=> constructors are available for abstract class, But doesnot in Interfaces.

-> we can create concrete method inside interface by using static, default other than this it will throws error.

Abstract	Interface.
1) Abstract class can have abstract and non-abstract methods	1) Interface can have only abstract methods since Java 8, it can have default and static methods also
2) Abstract class doesnot support multiple inheritance.	2) Interface supports multiple inheritance
3) Abstract class can have final, non-final, static and non-static variable	3) Interface has only static and final variables
4) Abstract class can provide the implement of interface	4) Interface can't provide the implementation of abstract class.
5) The abstract keyword is used to declare abstract class	5) The Interface key is used to declare interface.
6) An Abstract class can extend another Java class and implement multiple Java interface	6) An interface can extend another Java Interface only.

⇒ A class can extends two Interfaces but a single class

⇒ But In a single class another class will not be allowed.

** what we can we do data members in java?

all variables declared inside interface are implicitly public static final. variables (constants). All methods declared inside Java Interfaces are implicitly public and abstract, even if you dont use public or abstract keyword. Interface can extend one or more other interface

1) Marker
2) Functional Interface 3) Typical

It has only one single abstract method.

To @ Functional Interfaces

To make use of FunctionalInterfaces, we are using @Fun annotation.

⇒ If we want add another method in a single class by the @ others we are using.

```
public interface obj{
    void display();
    void show();
}
```

1) No method for Function Interface
2) one abstract Method

clone is a method present in a class

a) Functional Interface & An Interface that contains one abstract method is known as Function Interface.

Remote Interface

predefined marker interface present inside in the java.

b) Marker Interface; An Interface that contain no methods inside in it is known as Marker Interface. It is also known as Serializable.

c) Normal & An Interface that contains n. no. of methods and concrete methods known as Normal.

➡ Encapsulation?-

Hiding the data members.

If we trying to provide data to the public members

⇒ getters method help us to access the data from the ~~other~~ private from the other class

⇒ return type of setters is void

⇒ hide the data

⇒ we can make variables of data only to read.

∴ Getters

⇒ If we want show the values then we go for setters method.

* Encapsulation is a mechanism with which we wrap up the data members and functional members into a single object (in a single unit)

⇒ If we can read said to be getters Method

⇒ If we can write said to be setters Method.

⇒ Packages:

A java package is a group of similar types of classes, interfaces and sub-packages

⇒ package in java can be categorized into two form,

1) built-in package and

2) user defined package.

There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc

Advantage of Java package:-

⇒ Java package is used to categorize the classes and Interface.

Final (keyword):-

It is can be called with variables, Methods, classes.

⇒ If we are delaring a variable with a keyword final only the value of that variable perticalled change

object class:-

⇒ In JAVA each and every class directly or in directly inherites the properties of object class. that is (In essence of) i.e., object class is the supa most class in JAVA.

⇒ each and every class either a predefined class or a user defined class is a child class of object

Object class :- is the superclass present in Java.
Object defines its properties
class

Methods of Object :-

objects.getClass()

hashcode :- return type of hashcode is int

Java Lambda Expressions:-

***

⇒ The lamda expression is used to provide the implementation of a functional Interface.

⇒ In case of lamda expression, we don't need to define the method again for providing the implementation.

Functional Interfaces-

⇒ Lamda expression provides implementation of functional interface. An Interface which has only one abstract method is known as functional interface.

⇒ Java provides an annotation @FunctionalInterface, as functional Interface

why use Lamda Expressions-

⇒ To provide the Implementation of the Functional Interface.

⇒ less coding.

Threads:-

A Thread is a single sequence of executable code within a larger program. All the programs so far have used just one thread - the main thread that starts automatically when you run the program - but java lets you create programs that start additional threads to perform specific tasks

Eg:- web browsers can download files while letting you view web pages. when you download a file

Thread creation:-

In Java threads can be created in two ways:

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

Thread creation by extending the thread class

We create a class that extends the java.lang.thread class. This class overrides the run() method available in the Thread class.

Thread creation by implementing the Runnable Interface

We create a new class which implements java.lang.Runnable. Interface and override run() method. Then we instantiate a thread object and call start() method on this object

Every thread will have three important properties:

1. Thread Name.
2. Thread Id
3. Thread ~~propert~~ priority.

Thread Name:- Thread Names can be created by programmer in order to identify the threads.

Thread Id:- It is unique number (id) which is created and assigned by the thread scheduler to every single thread in order to identify them uniquely

Thread priority:- It is used by the thread scheduler to decide the order of execution of the given threads.

The priority of the thread is an integer value ranging between 1-10. We can set the priority of

11/09/19

Thread life cycle:-

→ The start method creates the system resources, necessary to run the thread, schedule the thread to run, and call the thread's run method.

→ A thread becomes "Not Runnable" when one of these events occurs:

a. If sleep method is invoked.

b. The thread calls the wait Method.

⇒ A thread dies naturally when the run method exits

⇒ A thread can be in one of the following states:

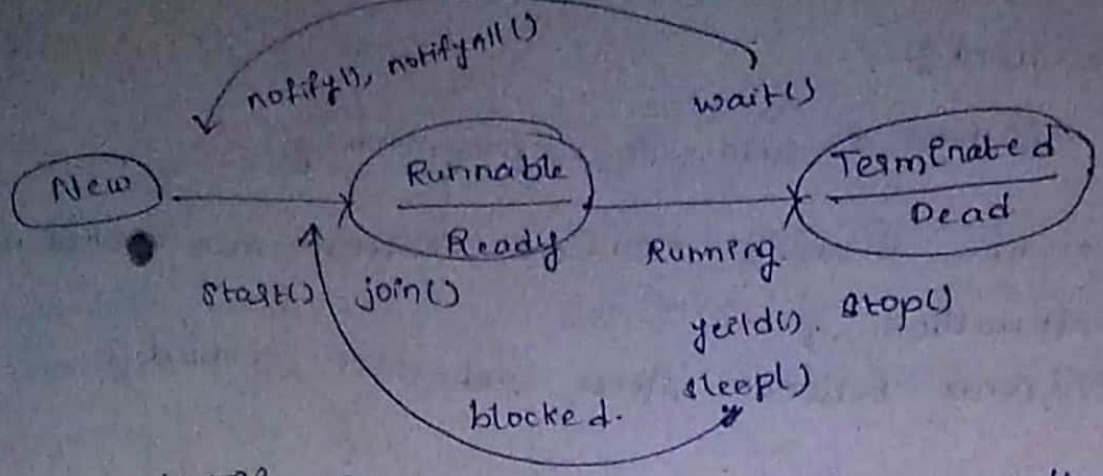
a) NEW - A thread that has not yet started in this state

b) RUNNABLE - A thread executing in the Java Virtual machine is in this state. {start()}

c) BLOCKED - A thread that is blocked waiting for a monitor lock is in this state. {yield(), sleep(), join()}.

d) WAITING - A thread that is waiting indefinitely for another thread to perform a particular action is in this state {wait(), notify(), notifyAll()}.

e) Terminated (stop) - A thread that has exited in this state {stop}.



Race condition:- Race condition occurs in a multi-threaded environment when more than one thread try to access a shared resource at the same time. Note, it is safe in multiple threads are trying to read a shared resource, as long as they are not trying to change it. since multiple threads try to race each other to finish executing a ~~the~~ method thus the name race condition.

Synchronization:-

⇒ In many cases concurrently running threads share data and two threads try to do operations on the same variables at the same time. This often results in corrept data as two threads try to operate on the same data

⇒ A popular solution is to provide some kind of lock primitive, only one thread can acquire a particular lock at any particular time. This can be achieved by using a keyword "synchronized"

⇒ By using the synchronize only one thread can access the method at a time and a second call will be blocked until the first call returns or wait() is called inside the synchronized method

Deadlock :-

Deadlock in Java is a programming

1) Difference between Run() and start() run method & start method :-

2) Difference between sleep and wait method :-

3)

start()	run()
1) when program calls start() method a new thread is created and code inside run() method is executed in Thread while if you call run() method directly no new new Thread is created and code inside run() will execute on current thread	
=> Most of the time calling run() is bug or programming mistake because caller has intent of calling start() to create new thread and this error can be detected by many static code coverage tools like findbugs	
=> start vs run in java thread is that you can not call start() method twice on thread object once started, second call of start() will throw IllegalStateException in Java while you can call run() method twice.	

Regex :- Regex

Regular expression :- The regular expression to data defines a pattern for a string. Regular expression can be used to search, edit or manipulate text. A regular expression is not language specific but many differs slightly for each stage.