JAVA PROGRAMMING NOTES

**BASIC FEATURES OF JAVA**

1. Simple
2. Object-oriented
3. Portable
4. Platform independent
5. Robust
6. Distributed

**Platform dependent and platform independent.**

As we all know that the machine understands only the language of 0s and 1s so we need to compile the code in that language before processing. Once it is compliled it produces a .exe or executable file which can only run on the platform ( windows / Mac) on which it was compiled.

**Why is JAVA called Platform independent Language (WORA) ie Write Once and Run Anywhere.**

In JAVA the code is written in a HLL high Level Language, now once this code is given to Java Compiler ,

This JavaCompiler generates a byte code. A byte code is neither a machine understandable nor a human understandable code. This is an intermideate code which can be sent to different platforms wherin other platforms can execute it. Now after passing from the byte code is sent to the **JVM**

**JAVA VIRTUAL MACHINE.** This JVM futher converts the HLL to MLL which is understandable by the computer.

**CLASSES AND OBJECTS**

Whenever we deal with Java we come across classes and objects. A class is a blueprint of the program.

And an object is any real word entity.

So every object has certain properties and has a behavior.

Lets take example of a student.

A student has

Properties : name, age ,gender ……….

Behavior : sleep(), eat(),study() ………..

So in program you can write as:

Class student{

String name;

Int age;

String gender;

Void sleep(){

//Code

}

Void eat(){

//Code

}

Void study(){

//Code

}

//Now making object for the class.

Student s=new student();

//now to access methods of the class.

s.sleep();

s.eat();

s.study();

}

**Main method in JAVA:**

The main method is nothing but the start point of a program. The JVM will execute only that code where there exists a main method.

A normal method syntax is as :

1. Method name
2. Method Parametre
3. Code inside the method
4. Method return type (int/void).

For ex :

Int Sum(int n1,int n2){

Int sum;

Sum=n1+n2;

Return sum; //here return type is integer so we use int in the method. We use Void() in case of no return type.

}

Now Main method:

Syntax:

**Class A{**

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

**}**

**}**

**Here**

**Public: Defines that the method is accessible to all the classes.**

**Static: JVM can access this method without making any object for the class.**

**Void: Is just the return type.**

**Main: name of the method.**

**String[] Args: Extra parametres that we can pass to the the main method.**

**System.out.println: To print**

**Statically Types vs Dynamically typed :**

Statically typed : Here the memory of the variable is defined at the time of compilation.

Eg: c,c++,JAVA

Dynamically Typed : Here the data type is already defined no need to specify the type of the variable.

Eg: Python, PHP.

**Variables in Java**

Variables are just used to store the values . Once it is made a memory is assigned to the variable.

Syntax:

Data\_type variable\_name = value.

Eg:

**Int num1=10**

**String name=”Ujjwal”**

There are certain rules in variable naming:

1. We cannot use reserved word as our variable. Reserved words like : void , int etc
2. Cannot start our variable with a number . eg: 1number. Not allowed.

Camelcase convention : noOfStudents.

Extra characters : $ ,\_ can also be used.

DataTypes:

There are two types of data types in Java:

Primitive Data Type: Boolean, char, byte, short ,int , long ,float and double.

Non Primitive Data Type: classes, string , interfaces and arrays.

For integer : Short , Byte , int and long.

Float-point number : float and double

Characters : char

Boolean : Boolean

Non-primitive : String , classes, array and Etc.

**Type-Casting in Java**

Changing data from one format to another form.

Implicit or explicit type.

*class* Typecasting {

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

*// implicit type conversion*

*// int n1=12;*

*// double b;*

*// b=n1;*

*// System.out.println(b);*

*// explicit type conversion*

        double n2 = 14.4;

        int a;

        a = (int) n2;

        System.out.println(a);

    }

}

**Identifiers in Java**

Identifiers is a name given to a package, class ,interface method or variable.All identifiers must have different names.

Rules:

Rule 1: Should begin with A to Z or a to z , $ and \_ must be unique.

Rule 2: after first character/letter identifers can have any combination of charqachters.

Rule 3: A keyword cannot be used as an identifier

Rule 4: Identifiers are case sensitive.

Rule 5: Whitespaces are not permitted .

Legal identifiers : rank , $name , \_rate ,\_\_2\_\_mark

Illegal identifiers : 102pqr, -name

**Operators in JAVA**

1. Arithmetic operators
2. Relational operators
3. Logical
4. Assignment
5. Unary
6. Bitwise

**Incrementation and decrementation:**

int a =5

a=a+1;

6

a=a-1

4

Other way of writing : a++ , a—

Prefix= ++a or –a

Postfix- a++ or a--

**Assignment Operator**

The value on the right side is assigned to the left side.

For eg:

Int a=12;

So 12 is assigned to variable a.

Lets take an example:

Int age;

Int cost;

Int price;

We can write the same variables in just a single line just by writing:

Int age,cost,price;

Taking another example of marks:

Int m1;

Int m2;

Int m3;

We can write like int m1,m2,m3,m4;

Another assignment technique is:

If values of many variables are same so we can write it as:

Int m4=m3=m2=m1=10;

Another way of assigning the values is:

If int a=10;

a+=5;

so, new assigned value of a will be 15.

Or int a =10;

a-=5;

so, new assigned value of a will be 5;

same for other operators also.

**Arithmetic Operators**

**== , < ,>,<=,>=,!=**

        int n1,n2,n3;

        n1=10;

        n2=12;

        n3=13;

      System.out.println(n1==n2);

      System.out.println(n1<n2);

      System.out.println(n1>n2);

      System.out.println(n1<=n2);

      System.out.println(n1>=n2);

      System.out.println(n1!=n2);

**true**

**false**

**true**

**false**

**true**

**Logical Operators:**

**1)And(&&)**

**2) Or (||)**

**3) Not(!)**

In AND even if one condition becomes false the resultant condition will be false.

In OR even If one condition is true out of the remaining condition the resultant condition will be true.

**IF-ELSE CONDITION AND LADDER**

Conditionals if – else:

If(condition)

{

}

Else{

// code

}

Eg:

Int age=18;

If(age<18){

//code

}

Else{

//code

}

We can make a nested if else loop that means if else inside an if statement.

*public* *class* ifelse {

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

        int age=33;

*// if(age<18){*

*//     System.out.println("You cannot vote ");*

*// }*

*// else{*

*//     System.out.println("You can vote now");*

*// }*

        if(age>=18 && age<=30){

            System.out.println("You can vote now");

        }

        else if(age>31 && age<=50){

            System.out.println("You are becoming old");

        }

        else if(age>51 && age<80){

            System.out.println("Bruh! you are old as fuck");

        }

        else{

            System.out.println("Lol! you are a kid still gotta live more");

        }

    }

}

**Ternary operator**

It is just like an if else statement just the syntax is different.

Int a,b,c;

a=12;

b=13;

c=14;

(a<b)? “a is lesser”:”b is lesser”;

The value before the colon is in the if block and the value after the colon is the else block.

*public* *class* ternaryOper {

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

        int a,b,c;

        a=12;

        b=13;

        c=14;

        String res=(a<b && a<c)? (b<c && b<a)?  "the smallest element is a":"the smallest element is c":"the smallest element is b";

*// String result=(a<b)? "a is lesser value":"b is the lesser value";*

*// System.out.println(result);*

        if(a<b && a<c){

            System.out.println("the smallest element is a");

        }

        else if(b<c && b<a){

            System.out.println("b is lesser");

            }

        else{

            System.out.println("c is the smallest");

        }

        }

    }

**Switch Case**

**Syntax:**

**Switch(expression)**

**{**

**Case1:**

**Break;**

**Case 2:**

**Break;**

**Default:**

**}}**

import *java.util.\**;;

*public* *class* Switchcase {

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

        int choice;

        int a,b,sum,dif,mult;

        a=10;

        b=4;

        Scanner sc=new Scanner(System.in);

        System.out.println("Enter your choice ");

        choice=sc.nextInt();

            switch(choice){

                case 1:

                sum=a+b;

                System.out.println(sum);

                case 2:

                dif=a-b;

                System.out.println(dif);

                case 3:

                mult=a\*b;

                System.out.println(mult);

                default:

                System.out.println("invalid values");

            }

        }

    }

**Loops in JAVA**

Loops: A loop is used to repeat a set of task for the programmers.

Different types of Loops:

For , while , do while loops

For eg: we wanna print a “\*” from 1 to 10 times .

So we say:

For(int i=0;i<=10;i++)

{

System.out.println(“\*”);

}

*public* *class* Loops {

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

        for (int i=0;i<=10;i++){

            System.out.println("I love myself");

        }

    }

}

I love myself

I love myself

I love myself

I love myself

I love myself

I love myself

I love myself

I love myself

I love myself

I love myself

I love myself