**Simple java code:**

class FirstExample

{

public static void main(String[] Args)

{

System.out.println("Q-Spider BTM.");

}

}

System is pre defined Class file.

Output object reference

println() is a method

JVM calls the main method to start the execution.

Class file name has to be same has the java fine name.

Date: 01-Feb-2022

**Q1. Write a java Program to perform addition, multiplication, division subsctraction in a single method? Print the result of each Operation.**

**Date: 01-Feb-2022**

**Local variable:** Local variable is a variable declared inside a block. Local variable should always initialized. If a variable declared as local without initialization by default the variable doesn’t contain any default values.

For ex:

class A

{

public static void main (String [] z)

{

Int x; //x is a local variable

System.out.println(“Value of x is “ + x); //Compile time error will come.

}

}

When we are trying to print the value of x, compile time error is getting because in the above program x is a local variable and was not initialized.

**Program to identify what the local value.**

class LocValue

{

public static void main(String[] Args)

{

int x=5; ***//Local to main***

System.out.println("main value" + x);

local();

}

public static void local()

{

int x= 10; ***//Local to local method***

System.out.println("main valu." + x);

}

}

**Write a java program to add 2 floating point numbers using method. The values should be passed form the main method. (Actual argument to formal argument)**

class Add

{

public static void main(String [] A)

{

float x= 10.50f;

float y= 5.50f;

Sum(x,y); ***//Actual argument***

}

public static void Sum(float A, float B) ***//Formal argument***

{

float R = A+B;

System.out.println("Result after addition = " +R);

}

}

**03- Feb-2022**

***Caller method & Called method:***

A method calling another method is known as **Caller method.**

The method being called by some other method is known as **Called method**

**EXAMPLE:**

class A

{

public static void main(String[] Args) ***//Caller method***

{

Add(100,200);

}

public static void Add(int x, int y) ***//Called method***

{

int C = x+y;

System.out.println(C);

}

}

In the above example, main() calls Add(). When the method call statements is getting executed, control goes to called method i.e Add() definition.

**METHOD with RETURN TYPE:**

If a method wants to return a value to the caller method, return type should be used instead of void. To return a value from the called method to a caller method ‘return’ [syntax- return variable] keyword can be used

**EXAMPLE-**

class A

{

public static void main(String[] Args) ***//Caller method***

{

int return\_value = Add(100,200); */\*the returned value stored in return\_value\*/*

}

public static int Add(int x, int y) *//Called method*

{

int C = x+y;

System.out.println(C);

return C*; /\*NOTE: Returning C value to C so that the same C value can be used in next program.\* Reusing the value of C further in diff method\*/*

}

}

**04- Feb-2022**

**Actual and Formal Arguments**

* **Actual arguments** are the values or the variables used in method call statement.
* **Formal arguments** are the variables or the parameters used in the method definition section.

Example:

class Sample

{

public static void main(String[] Args) ***//Caller method***

{

int a= 25;

int b= 2;

int c = Div(100,200); ***//Actual arguments***

System.out.println(c);

}

public static int Div(int x, int y) ***//Called method & Formal arguments***

{

int v = x/y;

System.out.println(v);

return v; ***//Method process terminates after return***

}

}

In the above program, 25 and 2 are passed by using the variable a and b respectively. In the method definition inside method header, the passed values are received inside the variables x and y respectively. Formal argument are separated by comma and each argument should be declared with datatype.

**Write a java program to perform the modular (%) operation on two integer values. The value should be passed from the caller method. The result value should be returned to caller method.**

class Mod

{

public static void main(String[] Arg)

{

int a = 25;

int b = 3;

int C= Modular(a,b);

System.out.println("Returned value "+ C);

}

public static int Modular(int x, int y)

{

int z=x%y;

System.out.println("value of Z= "+ z);

return z; //*can be written as return(z)*

}

}

Note: Execution happens in Stack Area frame by frame.

**Write a java program to calculate total marks of three subject each mark is out of 100. After that calculate percentage by designing one more method.**

class Marks

{

public static void main(String[] Args)

{

float Mat = 96f;

float Phy = 86f;

float Chem = 76f;

System.out.println(" Result \n");

float Sum= Total(Mat, Phy, Chem);

float Per= Percentage(Sum);

}

public static float Total(float M, float P, float C)

{

float sum = M+P+C;

System.out.println("Total mark is= " + sum );

return sum;

}

public static float Percentage(float S)

{

float Avg= S/300f\*100f;

System.out.println("\nPercentage mark is= " +Avg);

return Avg;

}

}

**05- Feb-2022**

*TYPE CASTING: ----*

* Type casting is the process of converting one type to another.
* It’s of 2 type
* Primitive
* Non-Primitive

**Primitive Typecasting:-** Processes of converting one of primitive to another primitive type that is compatible in nature. It can be of 2 types

* Widening
* Narrowing

**Widening:** It is the process of converting smaller datatype to larger/bigger datatype.

byte p = 5;

float q = p;

System.out.println(p); //55

System.out.println(q); //55.0

int p=100;

float q = p;

System.out.println(p); //100

System.out.println(q); //100.0

**Narrowing:** It’s the process of converting bigger datatype to smaller datatype. It should be performed explicitly by the programmer by using the typecast operator ().

Example:

float x = 10.5f;

int y= x;

Compile time error occur above. Use typecast operator to convert the float value inside **x** variable to an integer type variable **y.** Even if the type cast operator is used data loss will still occur. In the below example, **.5** loss.

float x = 10.5f;

int y= (int)x;

System.out.println(x); //10.5

System.out.println(y); //10

**Write a java program to implement primitive typecasting.**

class TypeCast

{

public static void main(String[] Args)

{

L2S();

S2L();

}

public static void L2S() *//Narrowing*

{

float a = 10.5f;

int b= (int)a; *//Explicit typecasting*

System.out.println("Typed value =" + a);

System.out.println("Type casting from Float to Integer ="+ b);

}

public static void S2L() *//Widening*

{

int x = 100;

float y = x; *// Implicit typecasting*

System.out.println("Value of x = " + x);

System.out.println("Value of y = "+ y);

}

}

**05- Feb-2022**

***Decision making statements:*** In a Java program code, there may be situations to execute only a set of instructions and skip another set of instructions. For that, we have decision making statements in Java.

We have 5 types of decision making statements

1. if
2. if else
3. Nested if
4. else if ladder
5. switch

***If condition:*** In this decision making statement, we can execute a set of instructions only if the conditions satisfies. That means we are putting the condition with if(condition) and there is a block of statements respective to if block.

**Syntax**

If (condition)

{

Statements to execute if conditions satisfies;

}

**Flow Chart:**

Condition

True

False

If block

Out side if block

In the above program if the condition is false the statement inside if block will not execute.

***if – else Condition:*** In this decision making statement the statements inside the if block is getting executed only if the condition is true. Otherwise the statements inside else block is executed.

**Syntax**

If (condition)

{

Statements to execute if conditions satisfies;

}

Else

{

Statements to execute if the condition fails;

}

**Flow Chart:**

If(Condition)

False

else block

True

If block

Out side if/else block

**Write a program to demonstrate if and else conditional statement.**

class Vote

{

public static void main(String[] Arg)

{

int age = 30;

if(age < 18)

{

System.out.println("Not eligible for voting!!"); *//True block for if*

}

else

{

System.out.println("You are eligible for voting"); *//Else block if condition fails*

}

System.out.println(“End of program”);***/\*Executable section out site if and else block\*/***

}

}