# **Variable**

**Date: 12th Feb 2024**

Variables are nothing but piece of memory use to store information.

One variable can store 1 information at a time.

Variables also used in information reusability.

**To utilise variables in java programming language we need to follow below steps:**

1. Variable declaration (Allocating/Reserving memory)

2. Variable Initialization(Assigning or Inserting value)

3. Variable Usage

//dataType

// String --> multiple characters --> shruti, Pratik, IT, computer, abc@1234, A+

// int --> numeric + non-decimal --> 10, 20, 10000, 5, -1, -11

// float --> numeric + decimal --> 2.5, 20000.5, 55.6, 61.2

// char --> single character --> A, B, a, x

## **Example1**

**package** Variable;

**public** **class** demo2

{

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

{

//Step 1: variable declaration

String studentName; //dataType variableName;

String studentDept;

//Step2: variable initialization

studentName="laxmi"; //variableName="variable Value";

studentDept="Computer Science";

//Step3: variable usage

System.***out***.println(studentName); //variableName

System.***out***.println(studentName); //info reusability

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

}

}

## **Example2:**

# **Datatype**

Date: 14th Feb 2024

Datatype are used to represent type of data or information which we are going to use in our java

program.

In java programming it is mandatory to declare datatype before declaration of variable.

In java datatypes are classified into two types:

1. Primitive datatype.

2. non-Primitive datatype.

## **1.Primitive datatype:**

There are 8 type of primitive datatypes(byte,short,int,long,float,double,char,boolean).

All the primitive datatypes are keywords.

\* **Memory size of primitive datatype is fix.**

The types of primitive datatype are:

**Note**:- keyword starts with lower case

Primitive datatype starts with lower case

syntax: datatype VariableName;

**1.(Numeric + Non-decimal):- Ex: 80,85,10,5, 10000 ..etc**

Data Type Size range

1. byte 1 byte -128 to 127

2. short 2 bytes

3. int(imp) 4 bytes

4. long 8 bytes l

**1GB=1024MB**

**1MB=1024KB**

**1KB=1024Byte**

**1Byte= 8bit**

**2. (Numeric + decimal):- Ex: 22.5,22.8,6.4....**

5. float(imp) 4 byte f

6. double 8 byte d

**3. Single Character: - Ex: A,B,X,Z.**

7. char 2 byte

**4. Conditional:- Ex: true,false.**

8. boolean 1 bit

## **2. Non-Primitive datatype:**

There are 4 types of non primitive datatypes .

All the Non primitive datatypes are identifiers.

\*Memory size of non primitive datatype is not defined or not fix.

Note: Identifier starts with capital letter.

Non-primitive datatype starts with capital letter.

e.g. Strings, Arrays, Classes, Interface, etc.

# **Methods**

A method is a block of code which only runs when it is called.

Methods are used to perform certain actions, and they are also known as functions.

You can pass data, known as parameters, into a method.

**Why use methods?** To reuse code: define the code once, and use it many times.

**1. main method (Pre-defined)**

In any Java program, the main() method is the starting point from where compiler starts program execution.

So, the compiler needs to call the main() method.

without main method we can't run any java program.

**2. Regular method (user defined)**

**1. static regular method**

1. static regular method call from same class --> methodname();

2. static method call from diffrent/another class -->className.methodname();

**2. non- static regular method**

3. non-static method call from same class --> 1. create object of same class 2. objectname.methodname();

4. non-static method call from diffrent/another class --> 1. create object of diff class 2. diffClassObjectName.methodname();

**Note**:

At the time of program execution main method is going to get executed automatically, whereas regular methods are not going to get executed automatically.

At the time of program execution priority is scheduled for main method only.

To call a regular method we need to make call method call from main method, until unless if the method call is not made regular method will not get executed.

regular methods can be called multiple times.

**5. method without/zero parameter**

**6. method with parameter.**

**7. method with return type**

## **Example1: static regular method call from Same class**

**package** Methods;

**public** **class** Sample1

{

//1: static regular method call from same class

//main method

**public** **static** **void** main(String[] args) //pre-defined

{

*m1*(); //methodName();

*m2*();

*m2*();

}

//static ->regular method

**public** **static** **void** m1() //user defined

{

System.***out***.println("running static regular method: m1");

}

//static -> regular method

**public** **static** **void** m2()

{

System.***out***.println("running static regular method: m2");

}

}

## **Example2: static regular method call from diff class**

**package** Methods;

**public** **class** Sample2

{

//2: static regular method call from diff class

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

{

Sample3.*m3*(); //diffClassName.methodname();

Sample3.*m4*();

Sample3.*m4*();

}

**package** Methods;

**public** **class** Sample3

{

**public** **static** **void** m3()

{

System.***out***.println("running static regular method m3 from diff class");

}

**public** **static** **void** m4()

{

System.***out***.println("running static regular method m4 from diff class");

}

}

## **Example3: non-static regular method from same class**

**package** Methods;

**public** **class** Sample4

{

//3: non-static regular method from same class

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

{

//1: Create object of current/same class

//2: method call --> objectName.methodName();

// className objectName=new className(); //object creation

//objectName.methodName();

Sample4 s4 = **new** Sample4();

s4.m5();

s4.m6();

s4.m6(); //method reuse

//1: Sample4 --> className -> datatype / objectType

//2: s4 --> objectName -> use to identify/refer object

//3: new --> keyword --> use to create blank/empty object

//4: Sample4() --> className() --> Constructor --> use to copy/load all the members of class into object

}

//non-static regular method

**public** **void** m5()

{

System.***out***.println("running non-static regular method: m5");

}

//non-static regular method

**public** **void** m6()

{

System.***out***.println("running non-static regular method: m6");

}

}

## **Example4: non-static regular method call from diff class**

**package** Methods;

**public** **class** Sample5

{

//4: non-static regular method call from diff class

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

{

//1: Create object of diff class

//2: method call -> diffClassObjectName.methodName();

Sample6 s6=**new** Sample6();

s6.m7();

s6.m8();

s6.m8(); //method reuse

}

}

**package** Methods;

**public** **class** Sample6

{

**public** **void** m7()

{

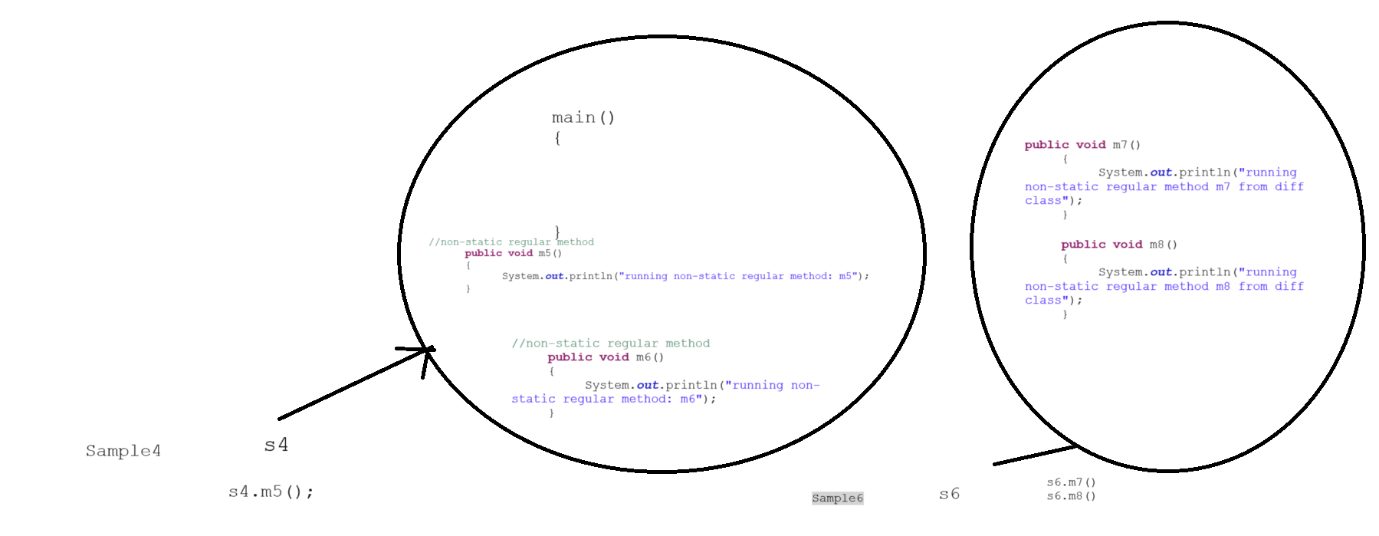
System.***out***.println("running non-static regular method m7 from diff class");

}

**public** **void** m8()

{

System.***out***.println("running non-static regular method m8 from diff class");

}****

}

## **Example5: Without/zero parameter method**

**Date: 23rd Feb 2024**

**package** Methods;

**public** **class** Sample7

{

//Example5: Without/zero parameter method

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

{

//1: static regular method call from same class

*m1*();

//2: static regular method call from diff class

Sample8.*m2*();

//3: non-static regular method call from same class

Sample7 s7=**new** Sample7();

s7.m3();

//4: non-static regular method call from diff class

Sample8 s8=**new** Sample8();

s8.m4();

}

**public** **static** **void** m1() //without/zero parameter method

{

System.***out***.println("static method m1 from same class");

}

**public** **void** m3() //without/zero parameter method

{

System.***out***.println("non-static method m3 from same class");

}

}

**package** Methods;

**public** **class** Sample8

{

**public** **static** **void** m2() //without/zero parameter method

{

System.***out***.println("static method m2 from diff class");

}

**public** **void** m4() //without/zero parameter method

{

System.***out***.println("non-static method m4 from diff class");

}

}

## **Example6: method with parameter**

### Example 6.1 : methods with all type of parameters

public class **Sample9**

{

//Example6: method with parameter

public static void main(String[] args)

{

add(10, 20); //30 variable initialization

add(50,60); //110

System.out.println("------");

Sample9 s9=new Sample9();

s9.sub(100, 50); //50

s9.sub(70, 110); //-40

System.out.println("---------------------------");

Sample10.mult(5, 6); //30

Sample10.mult(4, 3); //12

System.out.println("-------");

Sample10 s10=new Sample10();

s10.findCube(5); //125

s10.findCube(6); //216

}

**//method with 2 int(int, int) parameter**

public static void add(int num1, int num2) //variable declaration

{

System.out.println(num1+num2); //50+60 = 110

}

**//method with 2 int(int, int) parameter**

public void sub(int num1, int num2) //num1=80, num2=50

{

System.out.println(num1-num2); //80-50=30

}

}

public class Sample10

{

**//method with 2 int parameter**

public static void mult(int num1, int num2) //num1=5, num2=6

{

System.out.println(num1\*num2); //5\*6 = 30

}

**//method with int parameter**

public void findCube(int num) //num=5

{

System.out.println(num\*num\*num); //5\*5\*5 = 125

}

}

### Example 6.2 : methods with all type of parameters

**package** Methods;

**public** **class** **Sample11**

{

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

{

*studentName*("Rahul"); //variable initialization

*studentName*("Payal");

System.***out***.println("----------");

*studentFullName*("abc1", "xyz1");

*studentFullName*("abc2", "xyz2");

System.***out***.println("-------------");

*studentInfo*("Amol", 101, 51.5f, 'B');

*studentInfo*("Rahul", 102, 71.5f, 'A');

}

//method with String parameter

**public** **static** **void** studentName(String name) //variable declaration

{

System.***out***.println(name); //Payal //variable usage

}

//method with 2 String(String, String) parameter

**public** **static** **void** studentFullName(String FN,String LN)

{

System.***out***.println(FN +" "+ LN);

}

//method with String, int,float,char parameter

**public** **static** **void** studentInfo(String name,**int** rollNum, **float** per, **char** grade)

{

System.***out***.println("Student Name: "+name);

System.***out***.println("Student Roll Num: "+rollNum);

System.***out***.println("Student Percentage: "+per);

System.***out***.println("Student Grade: "+grade);

}

}

# **Control statements:**

1. if

2. if else

3. else if

4. nested if

5. switch

## **1: if statement**

if(condition)

{

//if body

}

## **2: if else statement**

if(condition)

{

//if body

}

else

{

//else body

}

## **3: else if**

### Example3.1:

If(condition1)

{

}

else if(condition2)

{

}

else if(condition3)

{

}

Or

### Example3.1:

If(condition1)

{

}

else if(condition2)

{

}

else

{

}

## **4: nested if**

### 4.1

If(condition1) //outer if

{

If(condition2) //inner if or nested if

{

}

}

Or

### 4.2

If(condition1) //outer if

{

If(condition2) //inner if or nested if

{

}

else

{

}

}

else

{

}

## **5: switch**

### 5.1

**public** **class** example5\_switch3

{

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

{

**switch**(9)

{

**case** 1: System.***out***.println("Today is Mon");

**break**;

**case** 2: System.***out***.println("Today is Tue");

**break**;

**case** 3: System.***out***.println("Today is Wed");

**break**;

**case** 4: System.***out***.println("Today is Thr");

**break**;

**case** 5: System.***out***.println("Today is Fri");

**break**;

**case** 6: System.***out***.println("Today is Sat");

**break**;

**case** 7: System.***out***.println("Today is Sun");

**break**;

**default**:

System.***out***.println("wrong case number");

}

}

}

### 5.2

**public** **class** example5\_switch1

{

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

{

String inp="MT";

**switch**(inp)

{

**case** "CD": System.***out***.println("running CD code");

**break**;

**case** "CW": System.***out***.println("running CW code");

**break**;

**case** "MS": System.***out***.println("running MS code");

**break**;

**case** "BI": System.***out***.println("running BI code");

**break**;

**case** "MT": System.***out***.println("running MT code");

**break**;

**default**: System.***out***.println("invalid input");

}

}

}

# **Loops:**

In programming languages, loops are used to execute a set of instructions/functions repeatedly when some conditions become true.

There are 4 types of loops in Java.

1. for loop

2. while loop

3. do while

4. for each 🡪array/collection--> selenium

## **1: for loop**

for (**initialization**; **Condition**; **Incr/decr**)

{

// code block to be executed

}

**Statement1: initialization**

is executed (one time) before the execution of the code block.

**Statement 2: Condition**

defines the condition for executing the code block.

**Statement 3: Incr/decr**

is executed (every time) after the code block has been executed.

## **2: while loop**

**Initialization**

While(**condition**)

{

While loop body

**Incr/decr**

}

## **3: do while**

**initialization**

do

{

//Do while loop body

**Incr/decr**

}

While(**condition**);

**Diff between while loop & do while:**

While loop checks the condition before executing the statement,

do-while loop executes the statement at least once before checking the condition

## **4: foreach**

pending

# Types of Variables

**Date: 8th Feb 2024**

**1. local variable:**

The variable which is declared inside the method/block/constructor is called local variable.

scope of local variable remains only within the method/block/constructor.

local variable is also called temporary variable.

**2. global variable:**

the variable which is declared outside the method/block/constructor is called global variable.

scope of global variable remains thought the class.

global variable is also called permanents variable.

**3. static/class variable:**

1. static variable call from same class -->variableName

2. static variable call from diff class--> className.variableName

**Note**: we can access static global variable in both static & non-static method

**4. non-static/instance variable: (instance-object)**

3. non-static variable call from same class

4. non-static variable call from diff class

## **Example1: of Local & Global Variable**

**package** TypesOfVariable;

**public** **class** **sample1**

{

**int** c=30; //global variable

**public** **void** m1()

{

**int** a=10; //local variable

System.***out***.println(a); //10

System.***out***.println(c); //30

}

**public** **void** m2()

{

**int** b=20; //local variable

System.***out***.println(b); //20

System.***out***.println(c); //30

}

}

**package** TypesOfVariable;

**public** **class** **Sample2**

{

//2: Static global variable

**static** **int** *num*=10; //static - global variable

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

{

//static global variable call from same class

System.***out***.println(*num*); //variableName

//static global variable call from diff class

System.***out***.println(Sample3.*num1*); //diffclassName.variableName

}

**public** **static** **void** m1()

{

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

}

**public** **void** m2()

{

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

}

}

**Date: 11th Feb 2024**

## Example2: Static global variable

**package** TypesOfVariable;

**public** **class** **Sample2**

{

//2: Static global variable

**static** **int** *num*=10; //static - global variable

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

{

//static global variable call from same class

System.***out***.println(*num*); //variableName

//static global variable call from diff class

System.***out***.println(Sample3.*num1*); //diffclassName.variableName

}

**public** **static** **void** m1()

{

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

}

**public** **void** m2()

{

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

}

}

**package** TypesOfVariable;

**public** **class** **Sample3**

{

**static** **int** *num1*=20; //static global variable from diff class or class variable

}

## Example 3: Non-static global(instance) variable

**package** TypesOfVariable;

**public** **class** Sample4

{

//Non-static global(instance) variable

**int** num2=30; //non-static global(instance) variable

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

{

//3: non-static global variable call from same class

Sample4 s4=**new** Sample4(); //1: create object of same class

System.***out***.println(s4.num2); //2: variable call -> objectName.variableName

//4: non-static global variable call from diff class

Sample5 s5=**new** Sample5(); //1: create object of diff class

System.***out***.println(s5.num3); //2: variable call -> diffclassObjectName.variableName

}

**public** **static** **void** m3()

{

Sample4 s44=**new** Sample4();

System.***out***.println(s44.num2);

}

**public** **void** m4()

{

System.***out***.println(num2);

}

}

**package** TypesOfVariable;

**public** **class** Sample5

{

**int** num3=40; //non-static global variable from diff class

}

# **Class**

A class is definition block where we can declare

Methods

Variables

Constructor

Block

A class is a template used to create objects and to define object data types and methods

Java provides a reserved keyword **class** to define a class. The keyword must be followed by the class name. Inside the class, we declare methods, variables, Constructor & block

Class className

{

//Class body

}

# **Constructor:**

**Date**: 13th March 24

**Constructor:**

A constructor in Java is a special member of class that is used to initialize global objects/variables.

The constructor is called when an object of a class is created.

**Types of Constructors**

1. Default Constructor

2. User defined Constructor

## **1. Default Constructor**

If Constructor is not declared in java class, then at the time of compilation compiler will provide Constructor for the class

If programmer has declared the constructor in the class, then compiler will not provide default Constructor.

The Constructor provided by compiler at the time of compilation is known as Default Constructor

**Date:** 14th March 23

## 2. **User defined Constructor**

If programmer is declaring constructor in java class, then it is considered to be as User defined constructor.

**package** Constructor;

**public** **class** Sample3

{

//2: User defined constructor

//step1: variable declaration

**int** num1; //num1=10

**int** num2; //num2=20

//step2: variable initialization

//user defined constructor --> provided by user

//use1: to initialize global variable

//use2: copy/load all the members of class into object

Sample3()

{

num1=10;

num2=20;

}

//step3: variable usage

**public** **void** add()

{

System.***out***.println(num1+num2);

}

**public** **void** mul()

{

System.***out***.println(num1\*num2);

}

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

{

//3: user defined constructor call from same class

Sample3 s3=**new** Sample3();

s3.add(); //30

s3.mul(); //200

System.***out***.println("------------------------");

//4: user defined constructor call from same class

Sample4 s4=**new** Sample4();

s4.sub(); // 10

}

}

**package** Constructor;

**public** **class** Sample4

{

//step1: variable declaration

**int** num3; //70

**int** num4; //60

//step2: variable initialization

//user defined constructor

//use1: to initialize global variable

//use2: copy/load all the members of class into object

Sample4()

{

num3=70;

num4=60;

}

//step3: variable usage

**public** **void** sub()

{

System.***out***.println(num3-num4);

}

}

### Example1: default constructor

**package** Constructor;

**public** **class** Sample1

{

//Example1: default constructor

//default constructor -> provided by compiler at the time of compilation

//use: to copy/load all the members of class into object

// Sample1()

// {

//

// }

**public** **void** m1()

{

System.***out***.println("running method m1");

}

**public** **void** m2()

{

System.***out***.println("running method m2");

}

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

{

//1: default constructor call from same class

Sample1 s1=**new** Sample1();

s1.m1();

s1.m2();

//1: sample1-> className -> datatype

//2: s1 -> objectName -> to refer/identify object

//3: new -> keyword -> use to create blank/empty object

//4: Sample1() -> className() -> constructor call -> to copy/load all the members of class into object

System.***out***.println("------------------");

//2: default constructor call from diff class

Sample2 s2=**new** Sample2();

s2.m3();

}

}

**package** Constructor;

**public** **class** Sample2

{

//default constructor -> provided by compiler

//use: to copy/load all the members of class into object

// Sample2()

// {

//

// }

**public** **void** m3()

{

System.***out***.println("running method m3 from diff class");

}

}

# **Logical Program**

## **1: EvenOdd**

**public** **class** example1\_EvenOdd

{

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

{

**int** num=99;

// System.out.println(num%2); //51%2 =1

// 4%2=0 == 0

**if**(num%2 == 0)

{

System.***out***.println("Even num");

}

**else**

{

System.***out***.println("Odd Num");

}

}

}