# Section 1- Basic Java

1. Variables & Data types
2. Methods
3. Constructor
4. Control Statements
5. Loops
6. Types of variables
7. Keyword and identifiers

# Variables & Data Types Variables:-

* + 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 utilize variables in java programing language we need to follow below steps:

1. Variable declaration (Allocating/Reserving memory)
2. Variable Initialization (Assigning or inserting value)
3. Usage

**Note**: -

* + According to all programming language dealing with information directly is not a good practice to overcome this variables are introduced.

# Data Types:-

* + Data type 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 data type before declaration of variable.
  + In java data types are classified into two types :

1. Primitive data type.
2. Non-primitive data type.

# Primitive data type:-

* + There are 8 type of primitive data types.
  + All the primitive data types are keywords.
  + Memory size of primitive data type are fix.
  + The types of primitive data type are:

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

# syntax: datatype variablename;

**1. (Numeric + Non-decimal):-**

Ex: 80,85,10,..etc

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Size** | **Description** |
| 1. byte | 1 byte | Stores whole numbers from -128 to 127 |
| 2. short | 2 bytes | Stores whole numbers from -32,768 to 32,767 |
| 3. int | 4 bytes | Stores whole numbers from -2,147,483,648 to |

2,147,483,647

1. long 8 bytes Stores whole numbers from -9,223,372,036,854,775,808 to

9,223,372,036,854,775,807

# 2. (Numeric + decimal):-

Ex: 22.5,22.8,6.4....

1. float 4 byte
2. double 8 byte

# 3. Character:-

Ex: A,B,X,Z.

1. char 2 byte

# 4. Conditional:-

Ex: true,false.

1. boolean 1 bit

# Non-primitive datatype:-

* + There are 2 types of non primitive datatypes .
  + Non primitive datatypes are identifiers.
  + Memory size of non primitive datatype is not defined.

**Note: -** Identifier starts with capital letter.

e.g. String, class

# Basic Programs of java

1. **Write good morning program in java package** Ahmedpur;

**public class** Malegaon

{

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

{

System.***out***.println("good morning"); System.***out***.println("vaishu");

}

}

# Consol window output-

good morning vaishu

# Write student information program in java package Variable\_DT;

**public class** Student\_info

{

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

{

//variable declaration

String StudentFirstName; String StudentFatherName; String StudentLastName;

**int** StudentAge;

**int** StudentClass; **char** StudentGrade; **float** StudentPer;

// variable initialization StudentFirstName= "Ranjeet"; StudentLastName= "kendre"; StudentAge= 10;

StudentClass= 4; StudentGrade= 'A'; StudentPer= 95.3f;

//usage

System.***out***.println("StudentFirstName=" + StudentFirstName); System.***out***.println("SttudentLastName=" + StudentLastName); System.***out***.println("StudentAge=" + StudentAge + "Year"); System.***out***.println("StudentClass=" + StudentClass + "th"); System.***out***.println("StudentGrade=" + StudentGrade ); System.***out***.println("StudentPer=" + StudentPer + "%");

}

}

# Consol window output-

StudentFirstName=Ranjeet SttudentLastName=Kendre StudentAge=10Year StudentClass=4th StudentGrade=A StudentPer=95.3%

# Methods:-

* + A method is a block of code which only runs when it is called.
  + You can pass data, known as parameters, into a method.
  + Methods are used to perform certain actions, and they are also known as functions.
  + Why use methods? To reuse code: define the code once, and use it many times.

# Main method

* + 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.

# Regular method

1. **static regular method**
   1. static method call from same class
   2. static method call from diffrent/another class

# non- static regular method

1. non-static method call from same class
2. non-static method call from diffrent/another class

# 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.

1. method without/zero parameter
2. method with parameter.
3. method with return type

# Write a program on static method call from same class package Methods;

**public class** static\_method\_call\_from\_same\_class

{

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

{

System.***out***.println("main method started");

//methodname();

*m1*();

System.***out***.println("main method ended");

}

# public static void m1()

{

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

}

}

# Consol window output-

main method started

running static regular method main method ended

# Write a program on static method call from different class package Methods;

**public class** static\_method\_call\_from\_different\_class

{

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

{

System.***out***.println("main method started");

//classname.methodname();

sample1.*m2*();

sample1.*m3*();

System.***out***.println("main method ended");

}

}

* 1. **different class package** Methods;

**public class** sample1

{

# public static void m2()

{

System.***out***.println("static regular method started");

}

# public static void m3()

{

System.***out***.println("static regular method ended");

}

}

# Consol window output-

main method started

static regular method started static regular method ended main method ended

# Write a program on non-static method call from same class

**package** Methods;

**public class** sample2

{

//non-static method call from same class

//sample2-->classname-->datatype

//s4-->objectname/variablename

//new-->operator/keyword-->to create blank object/instance

//sample2()-->classname()-->constructor-->to copy or load non static non static member in to object

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

{

System.***out***.println("main method started");

//classname objectname/variablename=new classname();

//objectname.methodname();

sample2 s4= **new** sample2(); s4.m4();

s4.m5();

System.***out***.println("main method ended");

}

**public void** m4()

{

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

}

**public void** m5()

{

System.***out***.println("running non static regular method ended");

}

}

# Consol window output-

main method started

running non static regular method running non static regular method ended main method ended

# Write a program on non-static method call from different class package Methods;

**public class** sample3

{

//non static method call from different class

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

{

System.***out***.println("main method started");

//classname objectname/variablename = new classname();

//objectname.methodname();

sample4 s5 = **new** sample4(); s5.m6();

s5.m7();

System.***out***.println("main method ended");

}

}

* 1. **diffrent class package** Methods;

**public class** sample4

{

**public void** m6()

{

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

}

**public void** m7()

{

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

}

}

# Consol window output-

main method started

running non static regular method from different method:m6 running non static regular method from different class:m7 main method ended

# Write a program on method without/zero parameter package Methods;

**public class** sample5

{

//method without/zero parameter

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

{

System.***out***.println("main method started");

//methodname();

*m1*();

System.***out***.println("main method ended");

}

# public static void m1()

{

System.***out***.println("runnung starting method started"); System.***out***.println("running starting method ended");

}

}

# Consol window output-

main method started

runnung starting method started running starting method ended main method ended

# Write a program on method with parameter package Methods;

**public class** sample6

{

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

{

//classname objectname/variablename = new classname();

//objectname.methodname();

sample7 pc = **new** sample7(); pc.guestAcc();

pc.adminAcc("abc");

}

}

* 1. **Different class package** Methods;

**public class** sample7

{

**public void** guestAcc()

{

System.***out***.println("login succesful");

}

**public void** adminAcc(String password)

{

System.***out***.println(password); System.***out***.println("login succesful");

}

}

# Control Statements-

* 1. **if**
  2. **if else**
  3. **else if**
  4. **nested if**
  5. **switch**

1. **if**

**package** Control\_Statement;

# public class If

{

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

{

**int** marks=35;

//if(conditon){ }

**if** (marks>=35)

{

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

}

}

}

# Consol window outout-

Pass

# if else

**package** Control\_Statement;

**public class** if\_else

{

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

{

**int** marks= 50;

// **if** (condition) { }

// **else** { }

**if** (marks>=51)

{

}

# else

{

}

}

}

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

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

# Consol window output-

Fail

# else if

**package** Control\_Statement;

**public class** else\_if

{

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

{

**int** marks=33;

**if** (marks>64)

{

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

}

**else if** (marks>59)

{

System.***out***.println("1st class");

}

**else if** (marks>49)

{

System.***out***.println("2nd class");

}

**else if** (marks>34)

{

}

# else

{

}

}

}

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

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

// if (condition) { }

// else if (condition) {}

// else { }

# Consol window output-

Fail

# nested if-

**package** Control\_Statement;

**public class** nested\_if

{

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

{

String UN=("abc"); String PWD= ("xyz");

**if** (UN=="abc")

{

System.***out***.println("correct UN");

**if** (PWD=="xyz1")

{

}

# else

{

}

# else

{

}

System.***out***.println("correct PWD"); System.***out***.println("login successful");

System.***out***.println("wrong PWD"); System.***out***.println("login failed");

System.***out***.println("wrong UN"); System.***out***.println("login failed");

}

}

}

# loops

* 1. **for loop**
  2. **while**
  3. **do while**

**1. for loop**

1. **Without for loop package** Loops;

**public class** withoutfor\_loop

{

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

{

System.***out***.println(1); System.***out***.println(2); System.***out***.println(3); System.***out***.println(4); System.***out***.println(5); System.***out***.println(6); System.***out***.println(7); System.***out***.println(8);

}

}

# Consol window output-

1

2

3

4

5

6

7

8

# Print the number from 0 to 100 package Loops;

**public class** for\_loop

{

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

{

**for** (**int** i = 0; i <= 100; i++)

{

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

}

}

}

# Syntax for for loop-

**for** (**int** i = 0; i < args.length; i++)

{

}

# Print the number from 100 to 0 package Loops;

**public class** for\_loop3\_print\_nb\_from\_100\_to\_1

{

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

{

// i=100; i>=1; i=i-1

**for** (**int** i = 100; i>=1; i=i-1)

{

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

}

}

}

# Print even no from 0 to 100 package Loops;

**public class** for\_loop2\_print\_even\_nb\_from\_0\_to\_100

{

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

{

// i=0; i<=100; i=i+2

**for** (**int** i = 0; i <= 100; i=i+2)

{

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

}

}

}

# Print even no form 100 to 0 package Loops;

**public class** for\_loop4\_print\_even\_nb\_from\_100\_to\_0

{

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

{

**for** (**int** i = 100; i>=0; i=i-2)

{

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

}

}

}

# Print odd no from 0 to 100 package Loops;

**public class** for\_loop2\_print\_odd\_nb\_from\_0\_to\_100

{

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

{

//i=1; 1<=100; i=1+2

**for** (**int** i = 1; i <= 100; i=i+2)

{

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

}

}

}

# Print odd no from 100 to 0 package Loops;

**public class** for\_loop5\_print\_odd\_nb\_from\_100\_to\_0

{

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

{

//i=99; 99>=1; i=99-2

**for** (**int** i = 99; i>=1; i=i-2)

{

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

}

}

}

# 2. while loop

1. **Print no from 0 to 100 package** Loops;

**public class** while\_loop1\_print\_nb\_from\_1\_to\_100

{

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

{

**int** i=1;

**while** (i<=100)

{

System.***out***.println(i); i++;

}

}

}

# Syntax for while loop-

**while** (condition)

{

}

# Print no from 100 to 0 package Loops;

**public class** while\_loop2\_print\_nb\_from\_100\_to\_0

{

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

{

**int** i=100;

**while** (i>=0)

{

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

}

}

}

# Print even no from 0 to 100 package Loops;

**public class** while\_loop\_print\_even\_nb\_from\_0\_to\_100

{

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

{

**int** i=0;

**while** (i<=100)

{

System.***out***.println(i); i=i+2;

}

}

}

# Print even no from 100 to 0 package Loops;

**public class** while\_loop5\_print\_even\_nb\_from\_100\_0

{

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

{

**int** i=100;

**while** (i>=0)

{

System.***out***.println(i); i=i-2;

}

}

}

# Print odd no from 0 to 100 package Loops;

**public class** while\_loop3\_print\_odd\_nb\_from\_0\_to\_100

{

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

{

**int** i=1;

**while** (i<=100)

{

System.***out***.println(i); i=i+2;

}

}

}

# Print odd no from 100 to 0 package Loops;

**public class** while\_loop4\_print\_odd\_nb\_from\_100\_to\_0

{

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

{

**int** i=99;

**while** (i>=0)

{

System.***out***.println(i); i=i-2;

}

}

}

# 3. do while loop

Syntax of do while loop

# do

{

}

**while** (condition);

# Print no from 0 to 10 package Loops;

**public class** do\_while\_loop

{

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

{

**int** i = 1;

# do

{

System.***out***.println(i); i++;

}

**while** (1<=10);

}

}

# Print no from 0 to 100

**package** Loops;

**public class** do\_while\_print\_nb\_from\_0\_to\_100

{

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

{

**int** i=0;

# do

{

System.***out***.println(i); i++;

}

**while** (i<=100);

}

}

# Print no from 100 to 0 package Loops;

**public class** do\_while\_print\_nb\_from\_100\_to\_0

{

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

{

**int** i=100;

# do

{

System.***out***.println(i); i=i-1;

}

**while** (i>=0);

}

}

# Print even no from 0 to 100 package Loops;

**public class** do\_while\_print\_even\_nb\_from\_0\_to\_100

{

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

{

**int** i=0;

# do

{

System.***out***.println(i); i=i+2;

}

**while** (i<=100);

}

}

# Print even no from 100 to 0 package Loops;

**public class** do\_while\_print\_even\_no\_from\_100\_to\_0

{

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

{

**int** i=100;

# do

{

System.***out***.println(i); i=i-2;

}

**while** (i>=0);

}

}

# Print odd no from 0 to 100 package Loops;

**public class** do\_while\_print\_odd\_nb\_from\_0\_to\_100

{

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

{

**int** i=1;

# do

{

System.***out***.println(i); i=i+2;

}

**while** (i<=100);

}

}

# Print odd no from 100 to 0 package Loops;

**public class** do\_while\_print\_odd\_nb\_from\_100\_to\_0

{

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

{

**int** i=99;

# do

{

System.***out***.println(i); i=i-2;

}

**while** (i>=0);

}

}

# Constructor

* + A constructor in Java is a special method that is used to initialize objects/variables.
  + The constructor is called when an object of a class is created.
  + At the time of constructor declaration below points need to follow:

1. Constructor name should be same as class name
2. You should not declare any return type for the constructor (like void).
3. any no of constructor can be declared in a java class but constructor name should be same as class name,but arguments/parameter should be different.
   * Use of Constructor
4. To initialize data member/variable
5. To copy/load non-static members of class into object --> when we create object of class
   * Types of Constructor
6. Default Constructor
7. User defined Constructor

# Types of variable:

1. **local variable**
   * Creating **variable inside method/block** is known as local variable.
   * Scope of local variable **remains only within the method** & they are **temporary**.

# global variable

* + Creating variable **outside method/block** is known as global variable.
  + Scope of global variable **remains throught the class** & they are **permanent**.

**Program on global variable package** Typesofvariables;

**public class** sample

{

**int** a=10; //global variable, scope throught the class

**public void** m1()

{

**int** b=20;

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

}

**public void** m2()

{

**int** c=30; System.***out***.println(c); System.***out***.println(a);

}

}

**Call sample class to test class or main method package** Typesofvariables;

**public class** Testsample

{

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

{

//classname objectname/variablename = new classname();

sample s = **new** sample();

//variable name.methodname();

s.m1();

s.m2();

}

}

# Consol window output

20

10

30

10

# class/static variable

* + **Declaring** the variable using **static keyword** is known as class/static variable because to access to static variable class name is used.
  + To access static variable from diff class we need to make use of below statement: classname.variablename;

# Program on static variable

**public class** sample1

{

**static int** *a*=10; //global variable/static variable/class variable

**public void** m3()

{

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

}

# public static void m4()

{

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

}

}

**Sample1 call from the testsample1 or main method package** Typesofvariables;

**public class** Testsample1

{

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

{

//classname.variablename; **int** c=sample1.*a*; System.***out***.println(c);

//System.out.println(sample1.a);

}

}

# Instance/non-static variable

* + All the non-static variables are known as instance variable because to access non-static variable instance (object) need to be created.
  + To access non-static variable we need to make use of below statement:

-Create object of class.

-objectname.variablename;

**Program on nonstatic / instance variable package** Typesofvariables;

**public class** sample2

{

**int** a=40;

String studentname="abc";

**public void** m5()

{

}

**public void** m6()

{

}

}

# Call nonsataic / instance variable to different class or main method package Typesofvariables;

**public class** Testsample2

{ //call non statics/ instance variable to different class

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

{

sample2 s2 = **new** sample2();

//int c=s2.a;

//System.out.println(c);

System.***out***.println(s2.a); System.***out***.println(s2.studentname);

}

}

**this keyword-**

* + this keyword is use to access global variable from the same class

**Testsample-**

**package** This\_super\_keywords;

**public class** Testsample

{

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

{

sample s = **new** sample(); s.m1();

}

}

# Sample-

**package** This\_super\_keywords;

**public class** sample

{

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

**public void** m1()

{

**int** a=20; //local variable System.***out***.println(a);

System.***out***.println(**this**.a); //call global variable this.a

}

}

1. Java Keywords & Identifiers Java Keywords:- ¬

Keywords are predefined, reserved words used in Java programming that have special meanings to the compiler.

For example:

abstract , assert , Boolean, break , byte, case ,

Catch, char, class, const, continue, default, do, double, else, enum

Extends, final, finally, float, for, goto, if, implements, import, instanceof

Int, interface, long, native, new, package, private, protected, public, return

short ,static ,strictfp ,super ,switch , synchronized ,this ,throw, throws,, transient

try, void ,volatile, while.

Java identifiers:- ¬

Identifiers are the name given to variables, classes, methods, etc. ¬

Rules for Naming an Identifier

1. Identifiers cannot be a keyword.

2. Identifiers are case-sensitive.

3. It can have a sequence of letters and digits.The first letter of an identifier cannot be a digit.

4. Whitespaces are not allowed. Similarly, you cannot use symbols such as @, #, and so on