**Day-06**

**25-01-2025**

**================**

**Dynamic program:**

**============**

-> Dynamic program is depending on the input.

-> For every execution time of the program, we can send different value/values to variables then according to the values (inputs) the output can be.

-> To write the dynamic program, we need:

1) Memory Statement

2) Input Statement

3) Output Statement

**1) Memory Statement**

**==============**

-> Memory statement can help to create the memory

-> after that, in that memory the value what we have assigned to the name of the data (variable) that can be store.

Ex: int a = 100 // Initialization

Syntax for the memory statement:

datatype nameOfTheData(Variable) = value;

**Datatypes:**

**=======**

-> Datatypes describing two things:

1) Size of Memory

2) Value range

-> Two types of datatypes:

1) Primitive types

2) Reference types

1) Primitive types

============

-> also called as "Basic datatypes" or "fundamental datatypes".

-> There are total 8-primitive datatypes:

1) byte datatype ==> byte

2) short datatype ==> short

3) int datatype ==> int

4) long datatype ==> long

5) float datatype ==> float

6) double datatype ==> double

7) char datatype ==> char

8) Boolean datatype ==> Boolean

**variable:**

**======**

-> a name

can be used to store the value in the memory.

-> variable ==> named memory

**Byte Variable:**

**=========**

1 byte ==> 8-bits

can be used to store total of 2^8 values (256)

range ==> 0 to 255

range ==> -128 to -1 and 0 to 127

default value ==> 0

while creating the byte variable:

if no value is assigned to that variable,

then that variable can store with '0' in the heap memory

and that value dynamically linked with stack memory location which is presented with variable name.

**Difference between variable assignment and variable initialization:**

**============================================**

variable declaration: consisting of type of data and name of the data.

variable assignment: name of the data, assignment operator (=) and value

variable initialization: when we can define declaration and assignment both in the same line

class MyClass{

public static void main(String args[]){

byte a; // variable declaration

a = 100; // variable assignment

byte b = 121; // variable initialization

}

}

**Byte variable declaration:**

**=================**

Syntax:

byte name;

name = -127

x = 10

Note:

=====

when we have declared the variable but not assigned, we can get the "compilation failed" error thrown by the compiler because the java is "strongly typed programming language".

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Datatype** | **Keyword** | **Default value** | **Memory** | **Range** |
| Byte type | byte | 0 | 1 byte | -128 to 127 |
| Short type | short | 0 | 2 bytes | -32768 to 32767 |
| Integer type | int | 0 | 4 bytes | -2^31 to 2^31 – 1 |
| Long type | long | 0 | 8 bytes | -2^63 to 2^63 – 1 |
| Float type | float | 0.0 f | 4 bytes | 1.8X10^-38 to 3.4X10^38 |
| Double type | double | 0.0 | 8 bytes | 2.3X10^-308 to 1.8X10^308 |
| Character type | char | \u0000 | 2 bytes | \u0000 to \uFFFF |
| Boolean type | boolean | False | 1 bit | true, false |

byte ==> 8-bite

-128 to 127

-2^7 to 2^7 - 1

data ==> n-bits

range ==> -2^(n-1) to 2^(n-1) - 1

float a;

double b;

a = 1.000234f;

b = 1.234;

Note:

====

1) The float variable can store the value in the heap memory with 5 to 6 decimal places after the decimal point.

2) The double variable can store the value in the heap memory with 10 to 12 decimal places after the decimal point.

3) Character data can always represent with single quotes.

Ex: char a = 'a';

**Day-07**

**27-01-2025**

**=================**

**Identifier:**

=======

-> a name

which we can use to name any programming entity or element like: variables, methods, classes, modules etc.

**Identifier Rules/Naming Conventions:**

**=========================**

1) Identifier must define with:

alphabets (A to Z/a-z)

digits (0 to 9)

Underscore sign (\_)

Dollar sign ($)

2) NO identifier can begin with digit

it can allow to begin with either alphabet or \_ or $

(and the identifier consisting of digits in between)

Ex: byte 9abc = 10; ==> Syntax error

short abc = 100;

int \_12abc = 123;

long $abc\_12 = 321;

class Datatype{

public static void main(String args[]){

byte a$\_9abc = 120;

}

}

3) No keyword of Java as an identifier.

4) Identifiers of java are case sensitive.

Ex: Java

java

JAVA

class Datatype{

public static void main(String args[]){

byte Int = 120;

}

}

==========================================

**Literals:**

======

-> Literal is a value

-> By default any value for the variable can be understood by the JVM as an Integer.

-> Suppose, when we can assign the value to the byte variable from the range of exceeded we can get "compilation error" (Possible lossy conversion from integer to byte)

Because:

the value we have assigned to the byte variable can decode by the JVM as "integer" which holds the memory of 4-bytes. Storing of 4-bytes value into 1-byte is not possible.

-> when we can assign a integer range value to long variable, there is no error but there is the memory leakage issue. Because the integer holds 4-bytes of memory and long holds ==> 8-bytes of memory. Storing of 4-bytes value into 8-bytes memory leads the memory wastage.

To avoid this, we can explicitly pass an instruction to the JVM:

the integer which suffix with 'l' or 'L' as Long type

otherwise ==> integer type.

Ex: long l = 123; // 123 ==> 4-byte value can store in 8-bytes memory

long l = 123l; // 123 ==> 8-bytes value storing in 8-bytes memory

-> Any floating-point value, the JVM can understand as double by default.

So double ==> 8-bytes

float ===> 4-bytes

storing of 8 bytes value into 4-bytes memory is not possible.

So, that every float value must be suffixed with 'f' or 'F'.

-> float/double literals can also define in exponential format/scientific format.

Ex: a = 12345678000000 ==> 1.23456789000000 X 10^13 ==> 1.23456789 X 10^6 X 10^13 ==> 1.23456789 X 10^19 ==> 1.23456789e19 ==> 1.23e19

b = 123456 ==> 123456/10000 X 10000 ==> 1.23456 X 10^5 ==> 1.23456e5

**How the java is Strongly typed programming language?**

**===================================**

In java, while assigning a value to the variable:

the JVM can check the memory but before it can check the order of the datatypes.

byte << short << int << long << float << double

class Datatype{

public static void main(String args[]){

byte a = 1;

short b = 120;

int c = 1234567890;

long d = 123L;

float e = 1.2F;

double f = 1.23;

float g = 1e23f;

double h = 123e9;

// long i = 1.23f;

float i = 123l;

System.out.println(a);

System.out.println(b);

System.out.println(c);

System.out.println(d);

System.out.println(e);

System.out.println(f);

System.out.println(g);

System.out.println(h);

System.out.println(i);

}

}

**Day-08**

**28-01-2025**

**===============**

**Variable:**

**======**

Named Memory (name which can used to represent the memory location)

which can be used to store any value

**Variable Declaration**

**=============**

Syntax:

datatype identifier/variable-name;

It can be used to describe about two things:

1) Type of the data

2) Name of the data

**Variable Assignment**

**==============**

Syntax:

Variable-name = literal;

-> When we need to access the variable after the declaration we must assign a value to variable.

Without assignment if we can try to access that variable, we can get "compilation failed" error. Because the Java is Strongly Typed Programming language.

**Variable Initialization**

**==============**

Syntax:

datatype variable-name = value/literal;

-> If we can declare and assign the variable within the same line, we called as "Variable Initialization".

**Integer Literals:**

**===========**

->Integral Datatypes (byte, short, int and long) are possible to define with 4-types of literals:

1) Binary literal

2) Octal Literal

3) Decimal Literal

4) Hexadecimal Literal

-> Binary Literal is base-2 literal.

which can always possible to define with allowed digits are 0 and 1 only.

-> In java, that every binary literal can always prefix with'0b' or '0B'

**Note:**

**===**

For any integral definition, the JVM can automatically decode as "decimal".

-> Decimal literal is a base-10 literal.

Which allows to define with total 10 digits which are from 0 to 9

-> Octal literal is a base-8 literal.

which can possible to define with 0 to 7 only.

-> Octal literal can always prefix with '0'.

-> Hexadecimal literal is base-16 literal

which allows characters are:

0 to 9 and a to f or A to F

also called as "Alphanumeric value"

**Character Literals:**

**============**

for character variable, we can assign Unicode values from '0' to '65535'.

**Boolean Literals:**

**===========**

-> are two:

true

false

Note:

=====

As java is strongly typed programming language, so the JVM never understand true as '1' and false as '0'.

class Practice{

public static void main(String[] args){

int a = 0b10101; // binary

int b = 0172; // octal

int c = 0xaf123;

char d = 'a';

// char e = -1;

// char e = 100l;

boolean e = true;

// float c = 0b1101.101f;

System.out.println(a);

System.out.println(b);

System.out.println(c);

System.out.println(d);

System.out.println(e);

// System.out.println(true + true);

}

}

**Day-09**

**29-01-2025**

**===============**

IO ==> Input and Output

Input ==> User/programmer can provide the input to the program as per the requirement using input device (Keyboard) this process is called "Input Operation".

Output ==> After the processing of the input, the program can give the output to the screen (output device) this process is called as "output operation".

-> To perform IO operations:

there are three different streams are available.

1) Input Stream ==> System.in

2) Output Stream ==> System.out

3) Error Stream ==> System. err

-> Java has a library with packages.

-> Packages is group of modules

-> Module is collection of: classes, object, methods etc.

-> The System is a pre-defined/built-in class in the java library which belonging to "java.util" package.

class System{

file in

file out

file err

}

**Input Statement**

**===========**

-> when we want to define the input statement, we need to import the "Scanner" class from the java.util package.

**Syntax:**

**import java.util. Scanner; (or) import java.util.\*;**

-> After the importing of Scanner class, we need to create an object for the Scanner class in the main() method.

-> Class --> Logical entity where as object --> physical entity

**Syntax:**

**Scanner object-name = new Scanner(System.in);**

**Output Statement**

**=============**

-> to define the output operation, we need a stream called as "System.out".

-> This stream have three built-in methods:

1) println()

2) print()

3) printf()

**Syntax:**

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

**====== Program =======**

import java.util.Scanner;

class DynamicProgram{

public static void main(String[] args){

Scanner scan = new Scanner(System.in);

System.out.println("Enter first value:");

int a = scan.nextInt();

System.out.println("Enter second value:");

int b = scan.nextInt();

int c = a + b;

System.out.println("The Sum of the given two numbers = "+c);

}

}

**How Java program can work?**

**===================**

1) We can create a java file which is having an extension with ".java". This file is called "source file"

2) Source file (.java) ====> compilation/compiler =====> Class Code (Byte code/.class)

javac (compilation)

class code ==> .class

3) class code (.class file) =====> Interpreter (JVM) =====> Object code

java (execution)

Note:

===

The Java is both compiler and interpreter dependent language.

**println() vs print() printf()**

**=================**

-> Every print() definition can write the output within the same line.

-> Every println() definition can write the output with new line.

-> printf() is same as print(),

but printf() must be integrate with format specifiers

Format specifiers are:

Integer ==> %d

Float ==> %f

char ==> %c

double ==> %lf

long ==> %ld

String ==> %s etc.

**====== Program ========**

import java.util.\*;

class Java Program{

public static void main(String[] args){

Scanner scan = new Scanner(System.in);

System.out.println("Enter your name:");

String name = scan.next();

System.out.print("Hi ");

System.out.print(name);

System.out.println("Hello ");

System.out.println(name);

System.out.printf("Hi ");

System.out.printf("%s",name);

}

}

**Day-10**

**30-01-2025**

**================**

**IDE (Integrated Development Environment)**

**=============================**

-> IDE is a software

in which we can develop (writing of code), Compile the code and execute the code

-> IDE is a platform

Different IDEs for JAVA development are:

VS (Visual Studio) Code

Eclipse

IntelliJ

STS etc.

**How to Install the Eclipse:**

**=================**

prerequisite: Java Software

1) We need to download and install the java software.

2) To check whether the java software has installed successfully or not:

open command prompt ===> type a command: "java --version" and hit enter button

3) Download the Eclipse software

open the browser ==> search about: "Eclipse download for any os"

(or)

https://www.eclipse.org/downloads/packages/release/luna/sr1/eclipse-ide-java-developers

according to your system configuration, you can download the eclipse software.

4) After the downloading:

you can extract that zip folder into normal folder.

5) Open that extracted folder,

double click on "eclipse" folder

double click on "eclipse.exe" file to get run the file for the installation.

6) Then click on next and follows the install and finish.

7) After the installation, open the eclipse and create the project:

file ==> new ==> project ==> select java project ==> enter the title ==> click on finish.