**Day-02**

**25-02-2025**

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

Memory Statement:

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

datatype name-of-data = value;

**Identifiers:**

**=======**

a name

can be used to name any entity within the program

Ex: variables, classes, objects, method etc.

Identifier Rules/Naming Conventions:

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

1) Identifier should include only:

alphabets

digits

Underscore sign (\_)

dollar sign ($)

2) Identifier never start with digit

but allowed to begin/start with either alphabet or \_ or $

Ex: int 9a = 99; ==> invalid representation ==> syntax error

int a = 99;

int \_9ab = 77; ==> valid

3) Java keywords are always write in lowercase.

And java keywords never be use as an identifier.

4) Identifiers are case sensitive.

class Identifiers{

public static void main(String[] args)

{

//int 9a = 99;

// System.out.println(9a); Syntax error

int a = 99;

int \_9a = 77;

int $ab = 101;

// int void = 121; void ==> keyword

int Void = 121; // Void ==> not a keyword

System.out.println(a);

System.out.println(\_9a);

System.out.println($ab);

System.out.println(Void);

}

}

**Variables:**

**=======**

-> Variable is a name

which can be used to store the value of any type.

-> The variable definition consisting of:

1) Declaration

2) Assignment

3) Initialization

-> Variable Declaration describe:

1) Type of the variable

2) Name of the variable

Syntax:

datatype identifier;

Ex: int a;

-> Variable assignment describe:

the assignment of value to the variable

Syntax:

identifier = value;

Ex: int a; // declaration

a = 121; // assignment

-> Variable initialization:

when we can declare the variable and assignment of the variable within the same line of the program is called as "Initialization".

Syntax:

datatype variable-name = value;

Ex: int a = 100;

class Variables{

public static void main(String[] args)

{

int a; // variable declaration

a = 99; // assignment

System.out.println(a);

char ch = 'R'; // initialization

System.out.println(ch);

}

}

-> The variables can be defined in two ways:

1) Compile time variable definition/Variable with fixed value representation

2) Run time variable definition

-------------------------------

**Input Statement:**

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

-> Input Statement can be used to take/access the input. -> During the execution of the program, we can process the input using keyboard that is called as "Run time variable definition".

-> When we want to define the run-time variable, we need to use "Scanner" object which is belonging Scanner class.

-> Class -> collection of data and methods

for every class, we can create one or more than one object.

-> Object is the reference of the class.

Ex: House on a sight ==> Real object

House plan on a sheet ==> Imaginary object -> class

-> Scanner class is a pre-defined class in java has defined in the java package called as "util".

Java ==> packages ==> classes ==> data and methods

-> When we want to use "Scanner" class for the run time variable definition, we need to import "Scanner" class from "util" package.

Syntax:

import java.util.Scanner;

-> The import statement can always write above the class definition only.

-> After the importing of the "Scanner" class, we should create the object for the Scanner class within the main() method.

Syntax:

Scanner Scanner-Object-Name = new Scanner(System.in);

-> After the creation of the Scanner object, we should use that object for assigning a value to the declared variable.

using pre-defined methods of Scanner class:

byte ==> nextByte()

short ==> nextShort()

integer ==> nextInt()

long ==> nextLong()

float ==> nextFloat()

double ==> nextDouble()

boolean ==> nextBoolean()

char ==> next() etc.,

Syntax:

variable-name = Scanner-object-name. scanner-method();

import java.util.Scanner;

class RunTimeVariable{

public static void main(String[] args)

{

Scanner scan = new Scanner(System.in);

int a; // declaration

System.out.println("Enter the value for the given variable:");

a = scan.nextInt();

System.out.println(a);

}

}

Dynamic Program Example:

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

import java.util.Scanner;

class SumOfTwoNumbers{

public static void main(String[] args)

{

Scanner s = new Scanner(System.in);

int a, b;

System.out.println("Enter value for a :");

a = s.nextInt();

System.out.println("Enter value for b:");

b = s.nextInt();

int sum = a + b;

System.out.println("The Sum of two numbers = "+sum);

}

}

**Output Statement:**

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

-> Output Statement can be used to display anything on the screen.

-> To define the output statement, we have three different methods:

1) println() ==> print line

2) print()

3) printf() ==> printed format.

Here the formats of the data are:

integer ==> %d

long ==> %l

float ==> %f

double ==> %lf etc.,

-> All the above three methods are pre-defined methods in java

have defined in "lang" package of the java library.

Note:

=====

to use println(), print() or printf() within the program, no need to import the lang package.

-> println() can give new line with every definition.

-> print() can print all definitions within the same line.

import java.util.Scanner;

class SumOfTwoNumbers{

public static void main(String[] args)

{

Scanner s = new Scanner(System.in);

int a, b;

System.out.println("Enter value for a :");

a = s.nextInt();

System.out.println("Enter value for b:");

b = s.nextInt();

System.out.println(a);

System.out.println(b);

System.out.print(a);

System.out.print(b+"\n");

System.out.printf("%d\n",a);

System.out.printf("%d\n",b);

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

System.out.print("The Value of b = "+b+"\n");

System.out.printf("The Value of sum = %d\n",a+b);

}

}