DAY-15

------

VARIABLES:

----------

Variables are classified into two types based on:

1. Data representation

2. Positional declaration

1.Data representation: Depending on what type of data we will assign to a variable.

It has been classified into two types:

1.1 Primitive variable : If a variable holds primitive data then such variables are called Primitive varaible.

ex: int a = 46;

1.2 Reference variable : If a variable holds address of the memory location of an object then such type of variables are called as ref variable.

ex: Lion l1 = new Lion();

2.Positional declaration: based on where we declare the varibales it has been classified into 3 types.

2.1 Instance variable

2.2 Static variable

2.3 Local variable

INSTANCE VARIABLE:

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

WHAT IS A INSTANCE VARIABLE ?

--> Value of a variable if it changes from object to object then such variables are called as Instance Variables.

NOTE:

-----

--> For every object seperate copy of instance varibles is created.

WHERE THE INSTANCE VARIABLES ARE DECLARED ?

--> Example:

Class Demo

{

int x;

float y;

boolean z;

void method1()

{

}

}

Instance Variables are declared within the class but outside the methods/blocks.

WHERE DOES INSTANCE VARIABLES ARE LOCATED IN MEMORY?

--> Instance variables present within the object inside the 'heap' memory.Hence it is also called object varible or attribute.

WHAT IS THE SCOPE OF THE INSTANCE VARIABLE ?

--> Scope refers to memory allocation and deallocation.

For instance variable memory will be allocated during object creation and deallocation will occur during object destruction.

Hence scope of instance variable is same as object scope.

HOW WE CAN ACCESS THE INSTANCE VARIBLES ?

--> Instance variables can not be accessed from static area.

ex:

---

class Demo

{

int x =25;

public static void main(String[] args)

{

System.out.println(x);

}

}

because: jvm will not allow to access instance variable in static region.

Instance variables can be accessed by using refernce variable.

example:

--------

class Demo

{

int x =25;

public static void main(String[] args)

{

Demo d1 = new Demo();

System.out.println(d1.x); --> using object reference

}

}

NOTE: JVM always gives default values for Instance variables

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

ex:

---

class Demo2

{

byte b;

short s;

int i;

long l;

float f;

double d;

boolean bo;

char ch;

String st;

public static void main(String[] args)

{

Demo2 d1 = new Demo2(); --> 0

System.out.println(d1.b); --> 0

System.out.println(d1.s); --> 0

System.out.println(d1.i); --> 0

System.out.println(d1.l); --> 0

System.out.println(d1.f); --> 0.0

System.out.println(d1.d); --> 0.0

System.out.println(d1.bo); --> false

System.out.println(d1.ch); --> space

System.out.println(d1.st); --> null

}

}

IS JAVA IS PURELY (100%) OBJECT ORIENTED PROGRAMMING LANGUAGE ?

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

YES, When compared with previous programming languages.

NO, considering only java it doesnot support few features operator overloading, multiple inheritence hence it is not 100% object oriented.

also, if a programming language is 100% object oriented then everything in the program must be considered as object.

but, in java we have primitive datatypes(non-object)

we can make our project 100% object oriented by using wrapper classes.

example

-------

byte b = 10; Byte b = new Byte(10);

int x = 25; Int i = new Int(25);

NOTE:

-----

The disadvantage with wrapper class is that it will occupy more memory.

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