Dt : 7/12/2020

***imp**

Variables in Java:

=>Variables are the data holders and which are used to hold data in the program.

Based on the DataTypes the variables are categorized into twp Types:

1.Primitive DataType variables

2.NonPrimitive DataType variables

1.Primitive DataType variables:

=>The variables which are declared with Primitive datatypes like byte,short,int,long,float,double,char and boolean are known as Primitive datatype variables.

Note:

=>These Primitive datatype variables will hold values.

2.NonPrimitive DataType variables:

=>The variables which are declared with NoPrimitive datatypes like Class,Interface,Array and Enum are known as NonPrimitive datatype variables.

Note:

=>These NonPrimitive datatype variables will hold object references.

(NonPrimitive datatype variables are also known as Reference variables)

Based on static keyword the variables are categorized into two types:

1.static variables

2.NonStatic variables

1.static variables:

def:

=>The variables which are declared with static keyword outside the methods are known as Static variables or Class variables.

MemoryLocation:

=>These static variables will get the memory within the class while Class loading and can be accessed with the class_name.

Scope and Visibility:

=>These static variables can be accessed by both static and NonStatic methods.

LifeTime:

=>These static variables are available until the class is available on Method_Area.

Note:

=>If static variables are not initialized then they are asiigned with default values.

2.NonStatic variables:

=>The variables which are declared without static keyword are known as NonStatic variables.

These NonStatic variables are categorized into two types:

(a)Instance variables

(b)Local variables

(a)Instance variables:

def:

=>The NonStatic variables which are declared outside the methods are known as Instance variables or Object Variables.

MemoryLocation:

=>These Instance variables will get the memory within the object while object creation and can be accessed with the Object_name.

Scope and Visibility:

=>These Instance variables are accessed only by NonStatic methods.

LifeTime:

=>Instance variables are available until the object is available on Heap_Area

Note:

=>If Instance variables are not initialized then they are assigned with default values.

(b)Local variables:

def:

=>The NonStatic variables which are declared inside the methods are known as Local variables.

MemoryLocation:

=>These Local variables will get the memory within the Method Frame while method execution.

Scope and Visibility:

=>These Local variables are accessed only inside the methods.

Note:

=>Local variables will not get default values.

==========

***imp**

Methods in Java:

=>Methods are the actions which are performed in the process of generating results.

These methods are categorized into two types:

1.static methods

2.NonStatic methods(Instance methods)

1.static methods:

=>The methods which are declared with static keyword are known as Static method or Class methods

=>These static methods will get the memory within the class while Class loading and can be accessed with the class_name.

structure of static methods:

```
static return_type method_name(para_list)
{
 //method_body
}
```

Coding Rule:

=>Static methods can access static variables directly but cannot access Instance variables directly.

These static methods are categorized into two types:

(i)Built-In static methods

(ii)User defined static methods

(i)Built-In static methods:

=>The static methods which are available from JavaLib are known as Built-In static methods.

(ii)User defined static methods:

=>The static methods which are defined by the programmer are known as User defined static methods

---

2.NonStatic methods(Instance methods):

=>The methods which are declared without static keyword are known as NonStatic methods or Instance methods.

=>These Instance methods will get the memory within the object while object creation and can be accessed with the Object_name.

Structure of Instance methods:

```
return_type method_name(para_list)
{
 //method_body
}
```

Coding Rule:

=>Instance methods can access both static variables and Instance Variables directly.

These Instance methods are categorized into two types:

(i)Built-In Instance methods.

(ii)User defined Instance methods

(i)Built-In Instance methods:

=>The Instance methods which are available from JavaLib are known as Built-In Instance methods.

Exp:

nextInt()

nextFloat()

...

(ii)User defined Instance methods:

=>The Instance methods which are defined by the programmer are known as User defined Instance methods.

---

define parameters?

=>parameters are the variables which are used to transfer the data from one method to another method.

Based on the parameters the methods are categorized into two types:

(a)Methods without parameters

(b)Methods with parameters

(a)Methods without parameters:

=>The methods which are declared without parameters are known as 0-parameter methods or Methods without parameter.

(b)Methods with parameters:

=>The methods which are declared with parameters are known as Parameterized methods or Methods with parameters.

---

define return_type?

=>return_type specify the methods will return the value after execution or not.

Based on return_type the methods are categorized into two types:

(a)Non Return_type methods

(b)Return_type methods

(a)Non Return_type methods:

=>The methods which donot return any value after method execution are known as Non Return_type methods.

(b)Return_type methods:

=>The methods which return the value after method execution are known as Return_type methods

Note:

=>In return_type methods we use 'return' statement to return the value after method execution.

=>The returned value will comeback to the method_call.

***imp**

Class generating Multiple Objects:

=>The class in Java can generate any number of objects without restriction.

=>The Multiple objects which are generated from the class are independent by their memory location on Heap_area.

=>The modification which is done in one object,will not effect remaining multiple objects.

Exp program:

wap to demonstrate variables?

```
class Display //SubClass
{
        int a=10;//Instance variables
```

```
	static int b=20;//Static variable
	void m1()//Instance method
	{
		int c=30;//Local variable
		a++;
		b++;
System.out.println("====Instance method m1()====");
System.out.println("The value a:"+a);
System.out.println("The value b:"+b);
System.out.println("The value c:"+c);
	}
	static void m2()//Static method
	{
		int c=40;//Local variable
		//a++;//Error
		b++;
System.out.println("====Static method m2()====");
//System.out.println("The value a:"+a);//Error
System.out.println("The value b:"+b);
System.out.println("The value c:"+c);
	}
}
class MainClass6 //MainClass
{
	public static void main(String[] args)
```

```
        {
Display d1 = new Display();//Object1
Display d2 = new Display();//Object2


d1.m1();
d2.m1();
Display.m2();
        }
}
```