JAVA BASICS.

LOCAL, INSTANCE & STATIC VARIABLES.

LOCAL VARIABLES

DECLARATION :

Inside methods, constructors or blocks

SCOPE :

Only inside methods, constructors or blocks not outside

WHEN VARIABLES GET ALLOCATED :

When methods, constructors or blocks gets created memory is allocated to the variables

WHEN VARIABLES GET DESTROYED :

When methods, constructors or blocks gets finished with execution the variables get destroyes

WHICH MEMORY IS USED TO STORE THEM :

Stack Memory

DEFAULT VALUES :

Doesn't have any default value & needs to be initialized with a value

ACCESS SPECIFIERS :

Cannot be used with local variables

SYNTAX :

dataType variableName;

INSTANCE VARIABLES

DECLARATION :

Instance variables in Java are declared within a class, but outside any method, constructor, or block. They are associated with individual instances or objects of the class.

SCOPE :

The scope of an instance variable is throughout the entire class. It can be accessed by any method, constructor, or block within the class.

WHEN VARIABLES GET ALLOCATED :

Instance variables are allocated memory when an object or instance of the class is created

WHEN VARIABLES GET DESTROYED :

Instance variables are destroyed when the object they belong to is no longer referenced and becomes eligible for garbage collection.

WHICH MEMORY IS USED TO STORE THEM :

Instance variables are stored on the heap memory

DEFAULT VALUES :

Instance variables are automatically assigned default values if no explicit value is assigned. The default values depend on the data type:

- Numeric types (int, float, double, etc.) are assigned 0.
- Boolean type is assigned false.
- Object references are assigned null.
- char type is assigned '\u0000' (null character).

ACCESS SPECIFIERS :

Instance variables can have different access specifiers to control their visibility and accessibility:

- public: The instance variable can be accessed from any class.
- private: The instance variable can only be accessed within the same class.
- protected: The instance variable can be accessed within the same class, subclasses, and other classes in the same package.
- Default (no specifier): The instance variable can be accessed within the same package.

SYNTAX :

accessModifier dataType variableName;

HOW TO ACCESS :

It depends on type of method

- For Normal Method It can be called directly
- For Static method it can be called by creating object of class.

STATIC VARIABLES

DECLARATION :

Static variables in Java are declared within a class, outside any method, constructor, or block of code, and are marked with the static keyword. They are associated with the class itself rather than individual instances or objects of the class.

SCOPE :

The scope of a static variable is throughout the entire class. It can be accessed by any method, constructor, or block within the class, including both static and non-static members.

WHEN VARIABLES GET ALLOCATED :

Static variables are allocated memory when the class is loaded by the Java Virtual Machine (JVM) and exist for the duration of the program's execution.

WHEN VARIABLES GET DESTROYED :

Static variables are destroyed when the program terminates or when the class is unloaded from memory, which typically happens when the JVM shuts down.

WHICH MEMORY IS USED TO STORE THEM :

Static variables are stored in the static memory area, also known as the "method area" or "permanent generation."

DEFAULT VALUES :

Static variables are automatically assigned default values if no explicit value is assigned. The default values depend on the data type:

- Numeric types (int, float, double, etc.) are assigned 0.
- Boolean type is assigned false.
- Object references are assigned null.
- char type is assigned '\u0000' (null character).

ACCESS SPECIFIERS :

Static variables can have different access specifiers to control their visibility and accessibility:

- public: The static variable can be accessed from any class.
- private: The static variable can only be accessed within the same class.
- protected: The static variable can be accessed within the same class, subclasses, and other classes in the same package.
- Default (no specifier): The static variable can be accessed within the same package.

SYNTAX :

accessModifier static dataType variableName;

HOW TO ACCESS :

We can access static variables in three ways

- Directly
- By using class name

- By using object reference name

-----THANK YOU-----

Follow me on LinkedIn