**Introducing java language**

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform.

Language fundamentals:

1. Identifiers
2. Reserved words
3. Data types
4. Literals
5. Arrays
6. Types of variables
7. Var-org methods
8. Main method
9. Command line arguments

10.Java codding standards.

Identifiers: a name in java program is called identifiers which can use for identification purpose. It can be class name, variable name, method name, lable name.

Rules for defining identifiers:

1. A to Z
2. a to z
3. 0 to 9
4. $
5. -

identifiers never start with digit.

Java identifiers are case sensitive.

Variable name always is lower case.

Method name always lower case.

Class name always upper case.

Java is an Object-Oriented Language. As a language that has the Object-Oriented feature, Java supports the following fundamental concepts −

* Polymorphism
* Inheritance
* Encapsulation
* Abstraction
* Classes
* Objects
* Instance
* Method

Message Parsing

**Object** − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behavior such as wagging their tail, barking, eating. An object is an instance of a class.

**Class** − A class can be defined as a template/blueprint that describes the behavior/state of an object.

**Constructor**: A constructor doesn't have a return type.

* The name of the constructor must be the same as the name of the class.
* Unlike methods, constructors are not considered members of a class.
* A constructor is called automatically when a new instance of an object is created.

**Method**: A **Java method** is a collection of statements that are **grouped** together to perform an operation.

**Variable**: A **Java variable** is a piece of memory that can contain a data value.

* Local variables
* Instance variables
* Class/Static variables.

Local Variables

* Local variables are declared in methods, constructors, or blocks.
* Local variables are created when the method, constructor or block is entered and the variable will be destroyed once it exits the method, constructor, or block.
* Access modifiers cannot be used for local variables.
* Local variables are visible only within the declared method, constructor, or block.
* Local variables are implemented at stack level internally.

There is no default value for local variables, so local variables should be declared and an initial value should be assigned before the first use.

Instance Variables

* Instance variables are declared in a class, but outside a method, constructor or any block.
* When a space is allocated for an object in the heap, a slot for each instance variable value is created.
* Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.
* Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.
* Instance variables can be declared in class level before or after use.
* Access modifiers can be given for instance variables.
* The instance variables are visible for all methods, constructors and block in the class. Normally, it is recommended to make these variables private (access level). However, visibility for subclasses can be given for these variables with the use of access modifiers.
* Instance variables have default values. For numbers, the default value is 0, for Booleans it is false, and for object references it is null. Values can be assigned during the declaration or within the constructor.
* Instance variables can be accessed directly by calling the variable name inside the class. However, within static methods (when instance variables are given accessibility), they should be called using the fully qualified name. *ObjectReference.VariableName*.

Class/Static Variables

* Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
* There would only be one copy of each class variable per class, regardless of how many objects are created from it.
* Static variables are rarely used other than being declared as constants. Constants are variables that are declared as public/private, final, and static. Constant variables never change from their initial value.
* Static variables are stored in the static memory. It is rare to use static variables other than declared final and used as either public or private constants.
* Static variables are created when the program starts and destroyed when the program stops.
* Visibility is similar to instance variables. However, most static variables are declared public since they must be available for users of the class.
* Default values are same as instance variables. For numbers, the default value is 0; for Booleans, it is false; and for object references, it is null. Values can be assigned during the declaration or within the constructor. Additionally, values can be assigned in special static initializer blocks.
* Static variables can be accessed by calling with the class name *ClassName.VariableName*.

When declaring class variables as public static final, then variable names (constants) are all in upper case. If the static variables are not public and final, the naming syntax is the same as instance and local variables.