1 Variable:

A **variable** in Java is a named storage location that holds a value, which can change during program execution, and is defined by its type.

2 Identifier: An **identifier** in Java is a name used to identify a variable, method, class, or other user-defined item in a program.

class and objects

Class: A **class** in Java is a blueprint for creating objects, defining their properties and behaviors.

Objects: An **object** is an instance of a class, representing a specific entity with its own state and behavior.

4. Literals: A **literal** in Java is a fixed value that is directly represented in the code. Literals can be of various types, including integer literals, floating-point literals, character literals, string literals, and boolean literals.

5. **Java Conditional Statements:**

1. **if Statement**: Executes a block of code if a specified condition is true.
2. **if-else Statement**: Executes one block of code if a condition is true and another block if it is false.
3. **else-if Ladder**: A chain of if-else statements that allow multiple conditions to be checked sequentially.
4. **switch Statement**: A multi-way branch statement that selects one of many code blocks to execute based on the value of a variable or expression.
5. **ternary Operator (?:)**: A shorthand for if-else that evaluates a condition and returns one of two values based on whether the condition is true or false.

6. **Java Loops:**

1. **for Loop**: Repeats a block of code a specified number of times based on a counter.(Entry Control)
2. **while Loop**: Repeats a block of code as long as a specified condition is true. (Entry Control)
3. **do-while Loop**: Similar to the while loop, but guarantees that the block of code is executed at least once before the condition is checked. (Exit Control)

7. Array: An **array** in Java is a data structure that stores a fixed-size sequence of elements of the same type, allowing for efficient storage and access using an index.

8. **Access modifiers** in Java are keywords that determine the visibility or accessibility of classes, methods, and variables. They control how the members of a class can be accessed from other classes or packages. The four main access modifiers are:

1. public: Members are accessible from any other class.
2. protected: Members are accessible within the same package and by subclasses.
3. default (no modifier): Members are accessible only within the same package.
4. private: Members are accessible only within the same class.

9. **Constructor:** A **constructor** in Java is a special method used to initialize objects. It has the same name as the class, does not have a return type, and is called when an object is created. Constructors can be parameterized or no-argument.

10. Inheritance: Inheritance is a fundamental concept in object-oriented programming that allows a new class (subclass) to inherit properties and methods from an existing class (superclass). This promotes code reuse and establishes a hierarchical relationship between classes.

11. Method Overloading: Method overloading is a feature in Java that allows multiple methods in the same class to have the same name but different parameter lists (type, number, or order of parameters). It enables the creation of methods that perform similar functions with different inputs.

12. Packages: A **package** in Java is a namespace that organizes a set of related classes and interfaces. It helps in avoiding name conflicts, controlling access, and making it easier to locate and use classes. Packages can be built-in (like java.util) or user-defined.

String

A **String** in Java is a sequence of characters used to represent text. Strings are immutable, meaning once created, their values cannot be changed. The String class provides various methods for string manipulation, such as concatenation, substring extraction, and searching.

Abstraction

Abstraction is an object-oriented programming principle that focuses on hiding the implementation details and showing only the essential features of an object. In Java, abstraction can be achieved using abstract classes and interfaces, allowing developers to define methods without providing their implementation.

Polymorphism

Polymorphism is the ability of a variable, function, or object to take on multiple forms. In Java, it primarily refers to method overriding (runtime polymorphism) and method overloading (compile-time polymorphism). It allows methods to perform different functions based on the object that is invoking them.

Encapsulation

Encapsulation is an object-oriented programming concept that involves bundling the data (attributes) and methods (functions) that operate on the data into a single unit, known as a class. It restricts direct access to some of the object's components and provides controlled access through methods (getters and setters), enhancing data security and integrity.