Elective 2 - Object-oriented Programming

**LESSON 1**

**Java -** is a high-level, class-based, object-oriented programming language designed to have as few implementation dependencies as possible.

**Java-** is widely used in web development, mobile apps, enterprise software, and many other areas.

**Year of Creation:** Java was created in 1995.

**Inventor:** James Gosling, along with Mike Sheridan and Patrick Naughton, at Sun Microsystems.

**Initial Purpose:** Originally designed for interactive television

**Green Team:** The original project was called the "Green Project," and the language was initially named "Oak" after an oak tree outside Gosling's office. It was later renamed "Java" after Java coffee.

**1995:** Java 1.0 was released **1998:** Introduction of Java 2

**2004:** Java 5.0 (J2SE 5.0) was released **2009:** Oracle acquired Sun Microsystems

**2014:** Java 8 introduced lambdas **2023:** The latest version, Java 20

**LESSON 2**

**public class HelloWorld**: This declares a class named HelloWorld.

**public static void main(String[] args):** This is the main method that serves as the entry point for the program.

**System.out.println("Hello, World!");:** This line prints the text to the console.

**Classes** - In Java, everything is defined within classes. A class is a blueprint for creating objects (instances of the class.

- are defined using the class keyword. They can be instantiated to create objects.

-It encapsulates data (attributes/fields) and behavior (methods/functions)

-We define a class named Car with two attributes: brand (string) and year (integer).

**INPUT/OUTPUT (IO) –** for reading input from the user, you can use classes like Scanner:

-For output, you've already seen the System.out.println() method, which prints to the console.

**PACKAGES -** Packages are used to organize classes into namespaces to prevent naming conflicts and promote modularity.

- Classes in the same package share the same namespace.

-Packages are declared at the beginning of the Java file:

**Built-in Packages** - There are a number of predefined Java packages like java, javax, org, etc. It's generally best to avoid using these names to prevent conflicts with standard libraries.

**Subpackages -** You can have subpackages within packages to further organize your code.

**LESSON 3**

**SYNTAX -** In C, you use #include to include the standard input/output library. In Java, you don't need such include statements for libraries; instead, you use import statements for external classes and libraries.

-In C, you define functions using the syntax returnType functionName(parameters), while in Java, you use accessModifier returnType functionName(parameters)

**FUNCTION VS. METHOD -** In C, you define functions outside of the main function. In Java, you define methods inside a class.

In C, the main function is the entry point for the program. In Java, it's also the entry point, but it's part of a class.

**PRINTING OUTPUT - I**n C, you use printf to print formatted output. In Java, you use System.out.println to print to the console.

**DATA TYPES-** In C, you use data types like int and returnType explicitly. In Java, you specify data types before variable and method declarations, but you also have to specify the access modifier (public in this case for the main method).

**OBJECT-ORIENTED-** In Java, everything is within classes, adhering to the objectoriented programming paradigm. In C, you have functions outside of classes (or structs), and it doesn't enforce OOP principles.

**LESSON 4**

**VARIABLES AND DATA TYPES-** In Java, variables are used to store data. You must declare a variable before using it, specifying its data type.

**STATEMENTS AND EXPRESSIONS -** Statements are complete lines of code that perform actions, while expressions are smaller parts of a statement that produce a value.

**CONDITIONAL STATEMENTS-** Conditional statements are used to make decisions in your code.

**LOOPS -** Loops allow you to repeat a block of code multiple times. The basic example of loops are the for loop, while loop and do-while loop.

**FOR LOOPS -** The for loop is typically used when you know how many times you want to repeat a block of code.

**WHILE LOOP -** The while loop is used when you want to repeat a block of code as long as a condition is true.

**DO-WHILE LOOP-** The do-while loop is similar to the while loop but guarantees that the block of code is executed at least once before checking the condition.

**EXCEPTIONS (TRY - CATCH) -** Exceptions are used to handle runtime errors gracefully.

**LESSON 5**

**ARRAYS-** Arrays in Java are homogeneous collections of elements with a fixed size.

**ARRAY LIST-** An implementation of a dynamic array, which can grow or shrink in size. • Allows efficient element access by index. • Suitable for most list operations when the size is not fixed.

**LINKED LIST-** Implements a doubly-linked list, allowing for efficient insertion and removal at both ends of the list. • Suitable when frequent insertions and removals are needed.

**HASH MAP** - A key-value pair data structure that provides fast retrieval of values based on their associated keys. • Suitable for implementing associative arrays and dictionaries.

**HASH SET**- An implementation of a set that does not allow duplicate elements. • Useful for storing a collection of unique elements.

**TREE SET -** A set implementation that maintains elements in sorted order. • Useful for maintaining a sorted collection of elements.

**STACK-** A Last-In-First-Out (LIFO) data structure. • Used for tasks like implementing undo functionality.

**QUEUE**- A First-In-First-Out (FIFO) data structure. • Useful for tasks like task scheduling.