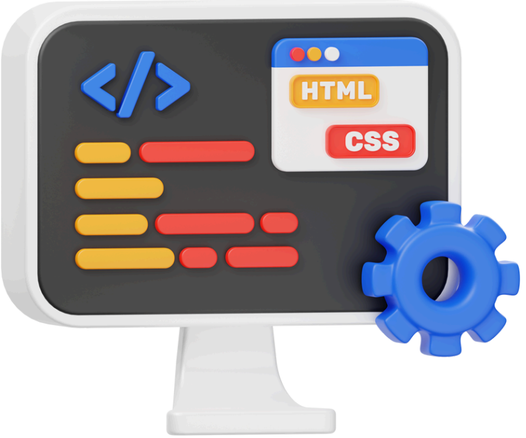
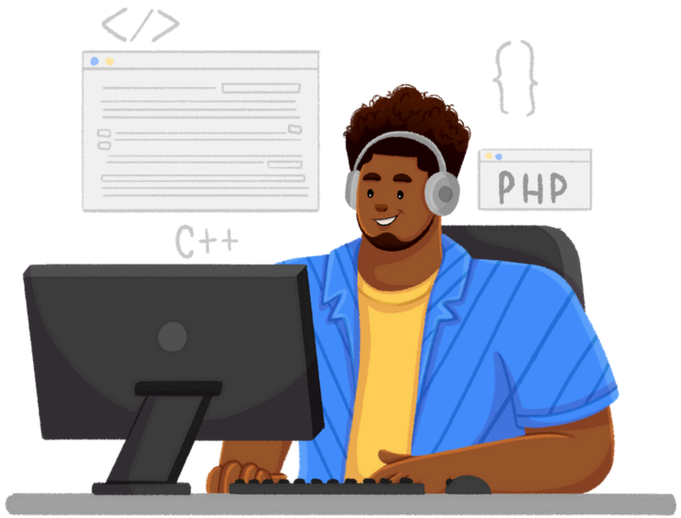
INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING (OOP)





EVOLUTION OF OOP

Pre-OOP Era (1950s-1970s)

- Procedural Language (Fortran)
- Structural Language (C)

Early Concepts Leading to OOP:

- Simula (1967): First to introduce classes, objects, inheritance, and polymorphism.

Formalization and Wider Adoption (1980s):

- Inclusion of oop concepts in C and the rise of C++.

Mainstream Adoption (1990s):

- Sun Microsystem builts Java.
- Microsoft builts C#.

Modern Era (2000s-Present):

- Agile works on oop principles.
- Functional programming.

OOP PRINCIPLES

Encapsulation

- Restricts access to an object's state using private attributes and public methods.

Abstraction

- Hides implementation details using abstract methods and classes.

Inheritance

- Inherits properties and methods from a parent class, allowing method overriding

Polymorphism

- Uses the same method name across different classes but with different implementations.

Java FEATURES

- Platform Independent
- Multithreading
- OOP Language
- Robust
- Garbage Collection
- Secure

Input statements

Scanner Class

< import java.util.Scanner >

Buffer reader class

Output statements

System.out.print()

System.out.println()

System.out.printf()

Command line arguments

```java
public class CommandLineExample {
    public static void main(String[] args) {

        // Checking if arguments are passed
        if (args.length > 0) {
            System.out.println("Command Line Arguments: ");
            for (String arg : args) {
                System.out.println(arg);
            }
        } else {
            System.out.println("No arguments passed.");
        }
    }
}
```

Input

```
java CommandLineExample Hello World 123
```

Output

```
Command Line Arguments:
Hello
World
123
```

Primitive Data Types in Java

B	byte (1)	F	float (4)
S	short (2)	D	double (8)
I	int (4)	C	char
L	long (8)	B	boolean

Reference Data Types in Java

C	Class
I	Interface
A	Array
S	String

Array

An array in Java is a collection of elements identified by an index or a key.

Array Declaration

```
int[] arr;        // Preferred style
int arr[];        // Valid but not preferred
```

Array Initialization

```
int[] arr = {10, 20, 30};  // Static
int[] arr = new int[5];  // Dynamic
```

Example:

```
public class SingleDimensionalArray {
    public static void main(String[] args) {
        // Declare and initialize a single-
dimensional array
        int[] numbers = {1, 2, 3, 4, 5};

        // Access and print elements of the array
        for (int i = 0; i < numbers.length; i++) {
            System.out.println(numbers[i]);
        }
    }
}
```

String

In Java, strings are objects that represent sequences of characters.

String creation using String Literal

```
String str1 = "Hello, World!";
String str2 = "Hello, World!";
```

String creation using new Keyword

```
String str1 = new String("Hello, World!");
String str2 = new String("Hello, World!");
```

String Characteristics

- Immutable: Once a string is created, it cannot be changed. Any operation that modifies a string will result in a new string being created.
- Reference Type: Strings are reference types in Java, meaning they store references to objects rather than the actual string data.

OPERATIONS ON STRINGS

1. **length(): Returns the length of the string.**
2. **charAt(index): Returns the character at the specified index.**
3. **substring(start): Returns a substring from the given starting index.**
4. **equals(String): Compares two strings for equality.**
5. **substring(start, end): Returns a substring between the specified start and end indices.**
6. **replace(old, new): Replaces occurrences of old with new in the string.**
7. **contains(substring): Checks if the string contains the specified substring.**

Thank You for Listening!