Vim-Pire Styled Code Highlight

Infinite Recursion:

Infinite recursion is when the function never stops calling itself.

Halting Condition/Base Case:

Every recursive function should have a halting condition, which is the condition where the function stops calling itself.

OOP:

- It stands for object-oriented-programming.
- It organizes code around objects that represents entities with the help of attributes and methods.
- It focuses on modularity and reusability.

Key Principles:

1. Encapsulation: Bundles data and methods in a class, controlling access with modifiers like private and using getters/setters.
2. Inheritance: Allows a class to inherit properties and methods from another using extends.
3. Polymorphism: Enables objects to use the same interface with different behaviors via method overriding or overloading.
4. Abstraction: Hides complex details, exposing only essentials through interfaces or abstract classes.

Class:

- A Class is like an object constructor, or a "blueprint" for creating objects. which consists of `attributes` and `methods`.
- A class is a template for objects, and an object is an instance of a class.

Object:

- When the individual objects are created, they inherit all the variables and methods from the class.

Constructor:

1. A constructor in Java is a special method that is used to initialize object attributes.
2. The constructor is called when an object of a class is created.

Note:

1. The constructor name must match the class name, and it cannot have a return type (like void).
2. All classes have constructors by default: if you do not create a class constructor yourself, Java creates one for you.
3. Constructors can also take parameters, which is used to initialize attributes.

Example:

```
public class Main {
  int x;  // Create a class attribute

  // Create a class constructor for the Main class
  public Main(int y) {
    x = y;  // Set the initial value for the class attribute x
  }

  public static void main(String[] args) {
    Main myObj = new Main(10); // Create an object of class Main (This
        will call the constructor)
    System.out.println(myObj.x);
  }
}
```

this keyword:

The `this` keyword in Java refers to the `current object` in a method or constructor.

Why we need it?

To differentiate between class variables and parameters(both method, constructor) when they have the same name.

```
public class Product {
    private String name;

    // Constructor using 'this' to differentiate
    public Product(String name) {
        this.name = name;
    }

    // Setter method using 'this' to differentiate
    public void setName(String name) {
        this.name = name;
    }
```

```
    public void display() {
        System.out.println("Product Name: " + this.name);
    }
}
```

You can also use `this()` to call another constructor in the same class.

This is useful when you want to provide default values or reuse initialization code instead of repeating it.

```
public class Person {
    String name;
    int age;

    // Constructor with one parameter (defaults age to 18)
    public Person(String name) {
        this(name, 18);  // Calls the two-parameter constructor
    }

    // Constructor with two parameters
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void printInfo() {
        System.out.println(name + " is " + age + " years old.");
    }

    public static void main(String[] args) {
        Person p1 = new Person("Alice");      // Uses default age
        Person p2 = new Person("Bob", 25);    // Uses given age

        p1.printInfo();
        p2.printInfo();
    }
}
```

Note: The call to `this()` must be the `first statement` inside the constructor.

When to use `this`? keyword?

1. When a constructor or method has a parameter with the same name as a class variable, use this to update the class variable correctly.
2. To call another constructor in the same class and reuse code.

Framework:

A framework is a set of reusable, pre-built structure of codethat provides a ready-made architecture for solving a common set of problems.

> A framework gives you a pre-written code in a standardized architecture often controls part of the application's flow (known as Inver-

sion of Control), meaning your code fits into the framework, and the framework manages how it runs.

1) What is *structure of code* means in the above context?

   "Structure means organized code that follows the framework's rules for flow, design, and interaction."

Such as:

- Defines how code is organized (folders, files, classes)
- Follows design patterns (e.g., MVC, layered)
- Ensures consistency and readability
- Framework expects specific conventions (annotations, method names)
- Controls flow of execution (e.g., hooks, lifecycle methods)
- Allows easier maintenance and scaling

What happens when you declare a class as final in Java?

Declaring a class final in Java prevents it from being subclassed.

1. It is used to secure the class's behavior.
2. Enforce intended usage without extension(API Design).

Framework in the context of Java Collections:

A **framework** in the context of **Java Collections** is a set of classes and interfaces that provides a ready-made architecture for solving a common set of problems — specifically, working with groups of objects.

'The Java Collections Framework (JCF)' offers built-in structures for:

1. Storing collections of objects
2. Organizing them efficiently
3. Manipulating them through standard algorithms

It standardizes the way collections are managed, making code more reusable, efficient, and easier to maintain.

Example Java Code

```java
// Hello world example
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, Vim-Pire!");
    }
}
```