20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by Mittul R
CB.EN.U4CYS21042
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

AMRITA
VISHWA VIDYAPEETHAM

Table of Contents

Imperative Programming Paradigm

What is Imperative Programming Language ?

- The word **"Imperative"** comes from the Latin word **"Impero"** meaning **"I Command"**
- Imperative programming is a Paradigm of Computer Programming in which the program describes a **sequence of steps that change the state of the computer.**
- It tells the Computer **"How"** to accomplish a task.
- To make programs simpler to understand, Statements have been Grouped into Sections using Blocks.
- Procedural and Object-Oriented Programming belong under Imperative Paradigm includes
  - C
  - C++
  - C
  - Java
  - Assembly
  - PHP

Key Concepts of Imperative Programming Language

How Imperative Programming tells the Computer **How to Accomplish a Task ?**

- Imperative programming involves giving explicit step-by-step instructions to the computer for performing a particular task.
- Key elements of imperative programming include variables, assignments, control structures (like loops and conditionals), and procedures or functions.
- In imperative programming, the focus is on providing precise instructions to the computer rather than on the overall logic or flow of the program.

The Main Principles of **Imperative Programming Language are :**

- State and Variables
- Assignment and Mutation
- Sequential Execution and Control Structures
- Procedures and Functions
- Side Effects and Explicit Algorithms
- Error Handling and Low-Level Control

Merits and De-Merits of Imperative Programming

Merits of Imperative Programming Language :

- Imperative programming is easy to read.
- Imperative Programming are comparatively easy to learn.
- Imperative Programming Conceptual model is very easy to understand

De-Merits of Imperative Programming Language :

- In Imperative Programming, the code gets confusing and quickly becomes very extensive
- There are more chances to errors at the time of editing the code.
- The maintenance blocks application development, which is the limitation of system-oriented programming.

Applications of Imperative Programming Language

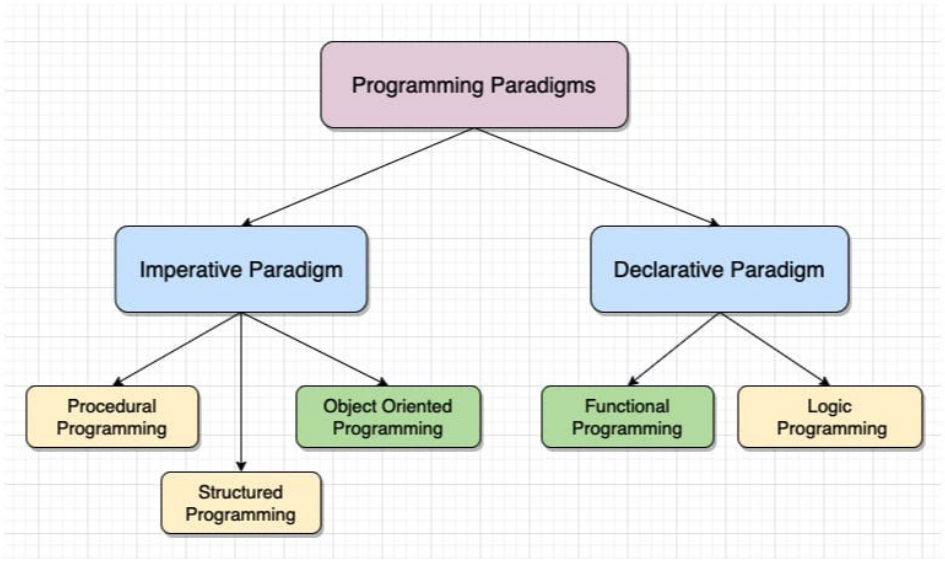

Figure: Programming Paradigm

Real Life Applications of Imperative Programming Language are :

- System Programming and Middleware
- Embedded Systems
- Game Development
- Device Drivers
- Network Programming

Imperative Programming - Java

Introduction to Java :

- Java is an object-oriented, platform-independent, and concurrent programming language that is widely used for various purposes.Designed to be platform-independent, object-oriented, and concurrent, Java is a high-level programming language with broad applications.Java stands out as a versatile and powerful programming language.
- **James Gosling, Mike Sheridan, and Patrick Naughton** initiated the Java language project in June 1991.**Sun Microsystems** released the first public implementation as **Java 1.0 in 1996.**
- The Latest Version of Java is **Java SE 21 LTS (19 - Sep - 2023)**

The Main Principles of Java Programming Language are :

- It must be simple, object-oriented, and familiar.
- It must be robust and secure.
- It must be architecture-neutral and portable.
- It must execute with high performance.
- It must be interpreted, threaded, and dynamic

Merits and De-Merits of Java

Merits of Java :

- Simple
- Object-Oriented
- Secured
- Robust
- Platform - Independent
- Mulithreading

De-Merits of Java :

- Performance
- Memory Consumption
- Cost
- Less Machine Interaction
- Garabage Collection

Applications of Java

Figure: Java

Real Life Applications of Java Programming Language are :

- Web Development
- Mobile Applications (Android)
- Cloud-based Services (Java-based APIs)
- Embedded Systems
- Scientific Applications

Working of Java Program

Before getting into, **"How Java Program Works ?"** , we need to get familiarize in some concepts like :

- Byte Code
- JVM - **Java Virtual Machine**
- JIT - **Just In Time Compiler**
- JRE - **Java Run Time Environment**
- JDK - **Java Development Kit**

Working Principle :

- In Java, during the compilation process, the source code is translated into an intermediate representation known as Byte Code. This Byte Code is not specific to any one type of computer architecture and can be executed on any device equipped with a Java Virtual Machine (JVM).
- The JVM, upon running the program, interprets the Byte Code and translates it into the native machine code of the underlying processor. This interpretation process, often referred to as "just-in-time" compilation, occurs line by line.

Working of Java

- Allowing for the execution of the program without the need for the entire Byte Code to be read and processed as a single entity.
- To address the potential performance issues associated with interpretation, Java utilizes a Just-In-Time (JIT) compiler. The JIT component identifies and compiles frequently executed segments of code into native machine code. Subsequently, when these segments are encountered again during program execution, the pre-compiled native code is provided to the JVM, thereby enhancing performance.
- In order to execute a Java program, the necessary libraries and files are stored within the Java Runtime Environment (JRE). The JRE encompasses the Java Development Kit (JDK), which includes the Java Virtual Machine (JVM) and other essential tools for Java development. The JDK serves as a comprehensive package containing the resources required for Java application development and execution.

Working of Java

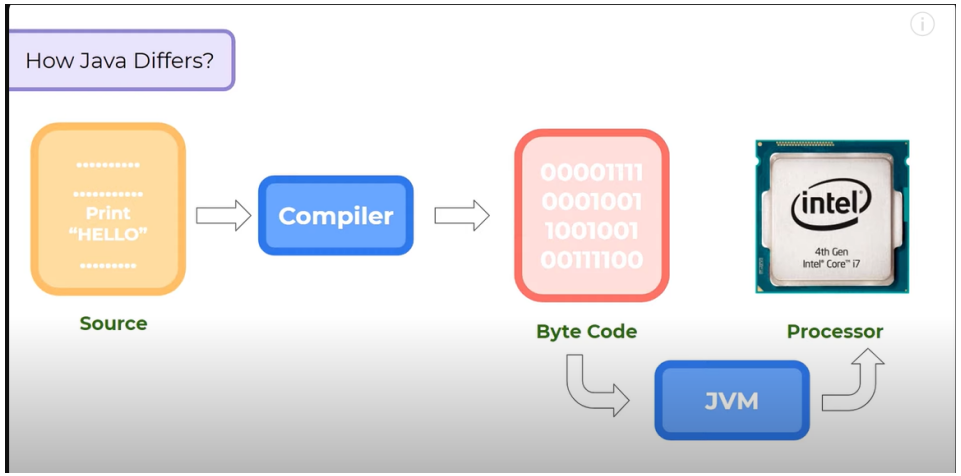

Figure: Byte Code

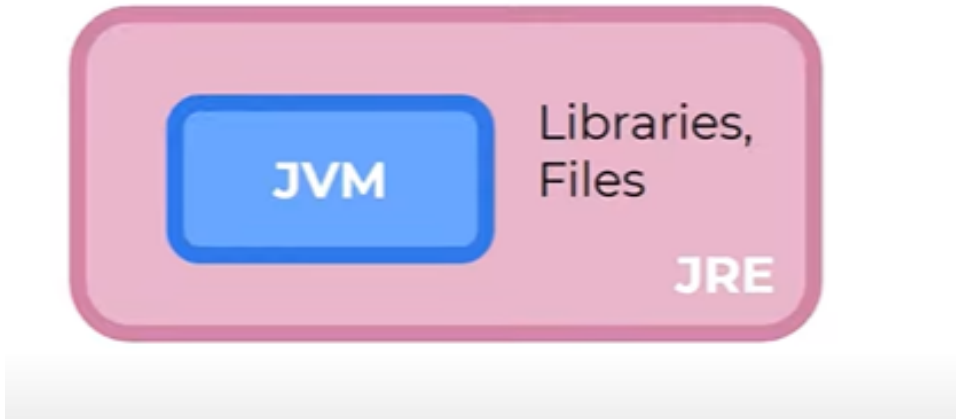

Figure: JRE

Working of Java

Figure: JDK

Rules for Writing a Java Program

- The Name of Class Definition , should be as same as File Name
- Main Class , should be always defined by the keyword **public**
- Main Method , should be always defined by the keyword **public static void**

Example Program

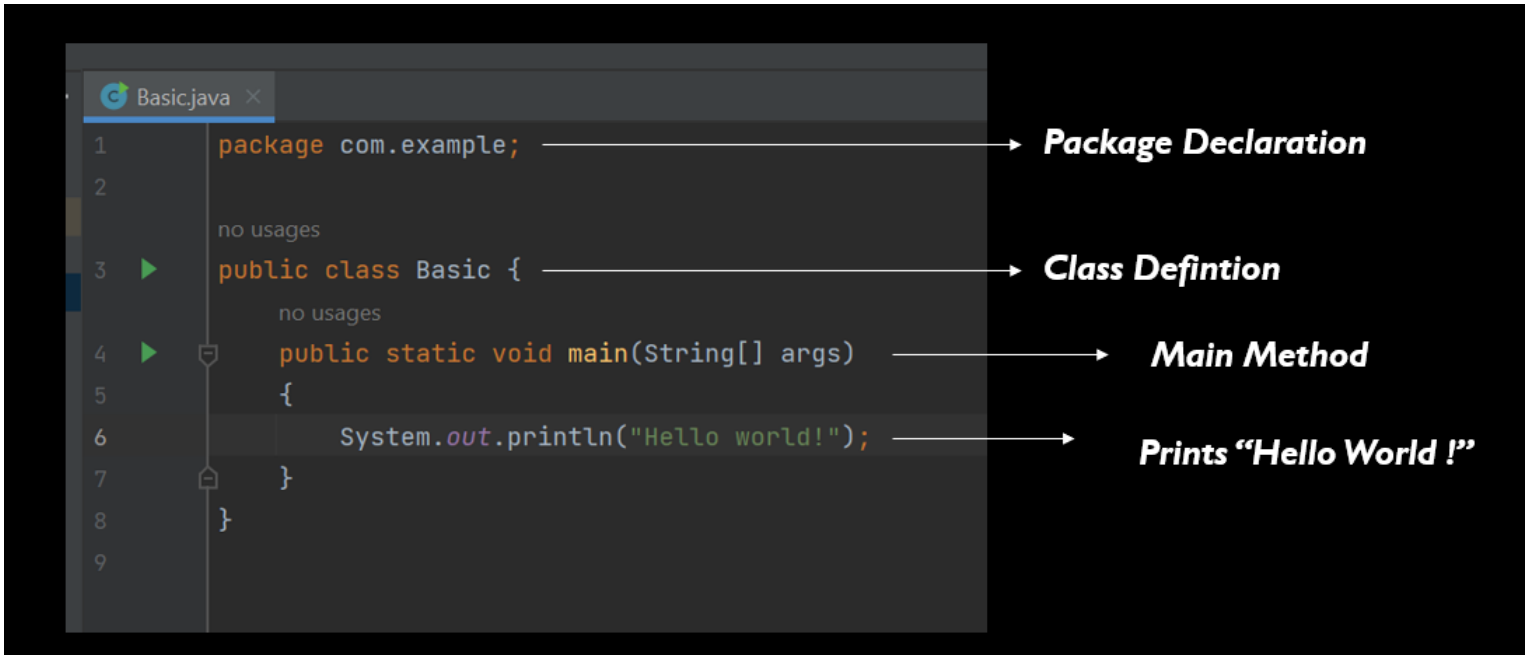

Figure: Hello World Program

- The Program is compiled using the keyword **javac <Filename.java>**
- Once if the program is compiled without any errors , then **java <Filename.class>** is used to print the output to the console

Output

```
C:\jdk-19.0.2\bin\java.exe "-javaagent:C:\Program
Hello world!
```

Figure: Output of Hello World Program

Main.java

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the Value for First Number: ");
        double num1 = scanner.nextDouble();
        System.out.print("Enter the Value for Second Number: ");
        double num2 = scanner.nextDouble();
        scanner.close();
        double sum = num1 + num2;
        System.out.println("Addition of " + num1 + " and " + num2 + " is: " + sum);
    }
}

```

Figure: Addition of Two Numbers

Output

```
Enter the Value for First Number: 23
Enter the Value for Second Number: 27
Addition of 23.0 and 27.0 is: 50.0
```

Figure: Addition of Two Numbers

Logic Programming

- Logic programming is a **programming, database and knowledge** representation paradigm based on **formal logic**
- A logic program is a set of sentences in **Logical Form,** representing knowledge about some problem domain.
- Computation is performed by applying **Logical Reasoning** to that knowledge, to solve problems in the domain
- The Logic programming paradigm isn't made up of instructions - rather it's made up of **Facts and Clauses**
- In Logic oriented programming we will build an entire program based on the **Notion of Logical Deduction.**
- The Logic programming using its knowledge base and tries to come up with a conclusion like **True or False**
- Logic programming languages that include **Negative Conditions** have the knowledge representation capabilities of a **Non-Monotonic Logic.**

Introduction of Logical Programming

The Different Programming Languages that comes under Logical Programming Paradigm are :-

- Prolog
- Mercury
- Datalog
- Answer Set Programming (ASP)

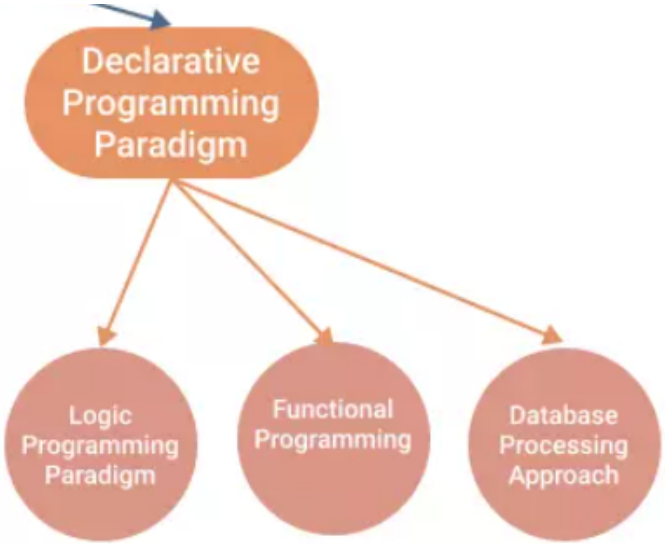

Figure: Logic Programming

Key Concepts of Logic Programming

Key Concepts of Logic Programming are :-

- Logic-Based Representation
- Facts and Rules
- Predicates and Relationships
- Queries and Inference
- Backtracking
- Unification
- Declarative Nature
- Horn Clauses
- Recursion
- Non Determinism

Merits and De-Merits of Logic Programming Paradigm

Merits of Logical Programming

- Declarative Nature
- Natural Language Processing
- Rule-Based Systems
- Backtracking
- Parallelism

De-Merits of Logical Programming

- Limited Efficiency for Some Tasks
- Complexity in Debugging
- Limited Support for Numeric Computation
- Learning Curve
- Not Universally Applicable

Logic Programming - Mercury

- Mercury is a **Functional Logic Programming Language made for real-world uses.**
- It is a purely **Declarative Logic Programming Language.**
- It is related to both **Prolog and Haskell.**
- It features a strong, static, polymorphic type system, and a strong mode and determinism system.
- It has the same syntax and the same basic concepts such as the **Selective Linear Definite Clause Resolution (SLD) Algorithm.**
- As such, it is often compared to its predecessor (Prolog) in features and run-time efficiency.
- Unlike the original implementations of Prolog, it has a separate **compilation phase**, rather than being directly interpreted.
- This allows a much **Wider Range of Errors** to be **detected** before running a program.
- It features a strict static type and mode system and a module system

Figure: Mercury - Logic Programming

Key Concepts of Mercury Programming Language

The Main Key Concepts of Mercury Programming Language are :

- Declarative and Logical Programming
- Purely Functional
- Strong, Static Typing
- Constraint Logic Programming
- Pattern Matching
- Backtracking
- Mode and Determinism Annotations
- High-Level Abstraction
- Modules and Namepaces

Working of Mercury Program

- The File extension used for Mercury programming language source files is typically **".m"**
- The Latest Version of Mercury Programming Language is **Mercury 20.06** released in **June 2020.**

How to Run Mercury Program in Terminal ?

- Create a Mercury source file , for example **"hello.m"**
- Compile the Mercury program using the command ,

Compile

```
mmc –make hello
```

- Then Run the compiled program ,

Run the makefile

```
./hello
```

Example of Mercury Program

```mercury
:- module hello_world.
:- interface.
:- import_module io.

:- pred main(io::di, io::uo) is det.

:- implementation.
main(!IO) :-
    io.write_string("Hello, World!\n", !IO).
```

Figure: Mercury - Hello World Program

```bash
mmc --make hello_world
./hello_world
```

Figure: To Compile Mercury Program

Notable Features of Both Paradigms

Imperative Programming - Java

- Uses statements to change the program's state.
- Focuses on how to perform tasks step by step.
- Relies on variables and assignment statements to store and manipulate data.
- Includes control structures like loops and conditional statements for flow control.
- Commonly used in languages like C, C++, and Java.
- Emphasizes the sequence of steps to achieve a specific goal.

Logic Programming - Mercury

- Based on formal logic and rules.
- Focuses on defining relations and rules rather than step-by-step instructions.
- Uses logical inference to derive conclusions from given facts and rules.
- Prolog is a popular language that follows the logic programming paradigm.
- Declarative in nature, where the programmer specifies what should be achieved rather than how to achieve it.
- Well-suited for problems involving complex relationships and constraints.

Analysis and Challenges Faced

Analysis

- Strength and Weakness of both paradigms ie **Imperative and Logic** and their associated programming languages **Java and Mercury** are discussed in the **Merits and De-Merits State**

Challenges faced in Imperative Programming - Java

- Java can be overwhelming due to its extensive features such as complexity , Memory Management , Threading and Concurrency. Memory management challenges can be addressed with memory profiling tools and best practices. Developing concurrent programs in Java requires understanding thread safety and synchronization, which can be tackled by learning about Java's concurrency utilities and best practices.

In Logic Programming - Mercury

- Exploring Mercury poses challenges in understanding logic programming, especially for us as we are familiar with imperative programming. Limited resources compared to mainstream languages like Java make finding learning materials challenging. Efficient execution and performance considerations in logic programming add to the complexity.

Comparsion and Discussions

Aspect	Java	Mercury
Paradigm	Imperative, Object-oriented	Logic, Functional
Typing	Strongly typed	Strongly typed
Memory Management	Manual memory management	Automatic memory management
Concurrency	Built-in threading support	Support for parallel execution
Learning Curve	Moderate	Steeper learning curve
Community Support	Extensive	Limited
Platform Independence	Platform-independent (via JVM	Limited platform independence
Usage	General-purpose	Specialized for logic programming

Figure: Mercury - Hello World Program

Conclusion

- Java is a powerful, object-oriented language that is widely used for general-purpose programming. It has a strong typing system, requires manual memory management, and offers built-in threading support. With its extensive community support and platform independence, Java is a great choice for various programming tasks.

- On the other hand, Mercury is a unique language that focuses on logic programming. It combines logic and functional programming paradigms and offers automatic memory management. While it supports parallel execution, it does have a steeper learning curve compared to Java. Additionally, Mercury has limited platform independence and community support, but it excels in its specialized field of logic programming.

References

- https://www.techtarget.com/whatis/definition/imperative-programming: :text=Imperative%20programming%20is%20a%20software,models%20are
- https://www.ionos.com/digitalguide/websites/web-development/imperative-programming/
- https://www.lsraheja.org/wp-content/uploads/2019/09/Imperative-Programming.pdf
- https://www.geeksforgeeks.org/java/
- https://www.simplilearn.com/tutorials/java-tutorial/what-is-java
- https://www.linode.com/docs/guides/logic-programming-languages/
- https://www.allassignmenthelp.com/blog/logic-programming-what-are-its-techniques/
- https://mercurylang.org/information/doc-latest/reference_manual.pdf
- https://mercurylang.org/documentation/papers/mfug_talk.pdf
- ChatGPT Prompt for getting Basic Hello World of Mercury Language