CS 104 - Object Oriented Programming (Theory) Lecture 1

Syed Adeel Ali Shah

September 23, 2024

Introduction of Teacher

Name: Syed Adeel Ali Shah
Position: Associate Professor
Institution: University of Engineering & Technology, Peshawar Profile: [https://sites.google.com/nwfpuet.edu.pk/adeel]

Course Marking Criteria

Sessional: 25 (8 marks for attendance)

Midterm Exam: 25

Final Exam: 50

Quizzes: 4 to 6

Assignments: 4 to 6

Course Conduction Policy

Theory: 3 hours per week

Lab: 3 hours per week

Use of Google Classroom is mandatory for course material distribution, announcements, and assignment submissions. Please accept the course invitation email already sent to your email at uetpeshawar.edu.pk domain.

If you have not received the invitation email, its your responsibility to inform the instructor of the course.

Plagiarism Policy

Zero Tolerance for Plagiarism:

Any form of plagiarism will result in zero marks for both the individual involved and the one providing the material.

AI-generated content will also receive zero marks.

Laptop Policy

Laptops are Mandatory:
Laptops are required during both theory and lab classes.

Types of Programming Languages

Low-Level Languages: Directly understandable by the computer hardware. Examples include machine language and assembly language.

High-Level Languages: Closer to human language and provide abstraction from machine details. Examples include Java, C++, Python, and JavaScript.

Procedural Languages: Organized around procedures or

routines. Examples include C and Pascal.
Object-Oriented Languages: Organized around objects that encapsulate data and behavior. Examples include Java, C++, and Python.
Functional Languages: Focus on computation as the evaluation of mathematical functions. Examples include Haskell and Lisp.
Scripting Languages: Interpreted languages often used for automation and rapid prototyping. Examples include Python, Ruby, and JavaScript.

Static vs. Dynamic Typing

Statically Typed Languages

Variable types are explicitly declared at compile time.

Type checking occurs during compilation.

Type errors are caught before runtime.

Variables hold values of fixed types.

Examples: Java, C++, Swift

Dynamically Typed Languages

Variable types are determined implicitly at runtime.

Type checking occurs at runtime.

Type errors may only be discovered during execution.

Variables can hold values of any type.

Examples: Python, JavaScript, Ruby

Compile Time vs. Interpretation Time

Compile-Time Languages: Languages like Java, C, and C++ are compiled languages where the source code is translated into machine code (binary code) before execution. Compilation occurs before program execution, and the resulting executable file can be run multiple times without recompilation.

Interpretation-Time Languages: Languages like Python, JavaScript, and Ruby are interpreted languages where the source code is executed line by line by an interpreter during runtime. Interpretation occurs at runtime, and the source code is not translated into machine code beforehand.

Hybrid Approaches: Some languages, like Java, use a hybrid approach where the source code is compiled into an intermediate bytecode, which is then interpreted by a virtual machine (JVM). This combines the advantages of both compilation and interpretation.

Java: Compiled to Bytecode

In Java, the source code is compiled into bytecode, a

platform-independent intermediate representation of the program. The bytecode is then interpreted by the Java Virtual Machine (JVM) at runtime, which translates it into machine code suitable for the underlying hardware.

This compilation to bytecode allows Java programs to be run on any device with a JVM, providing platform independence and enabling the ”Write Once, Run Anywhere” (WORA) capability.

Performance Comparison: Compilation vs. Interpretation

Compilation: Generally results in faster execution as the entire program is translated into machine code beforehand, and
optimizations can be applied during compilation.

Interpretation: May have slower startup time as the code is interpreted line by line during runtime, but it can offer advantages like portability and dynamic code execution.

Just-In-Time (JIT) Compilation: JIT compilers translate bytecode into native machine code at runtime, combining the benefits of both compilation and interpretation. spreading the compilation time across the program's runtime.

Why Java?

Java is a robust, high-level programming language with a large ecosystem and community support.

It offers platform independence through its ”Write Once, Run Anywhere” (WORA) philosophy, achieved by compiling Java source code to bytecode executed by the Java Virtual Machine (JVM). Java’s object-oriented nature promotes modularity, scalability, and code reuse, making it suitable for developing complex, enterprise-level applications.

Key Features of Java that make it Robust

Platform Independence: Java programs can run on any device with a JVM, ensuring portability across different platforms.

Memory Management: Java's garbage collection mechanism automatically manages memory allocation and deallocation, reducing the risk of memory leaks and segmentation faults.

Exception Handling: Java provides robust exception handling capabilities, allowing developers to gracefully handle runtime errors and maintain program stability.

Security: bytecode verification, access control mechanisms, and security manager, to protect against malicious attacks and ensure

the integrity and confidentiality of Java applications.

Type Safety: Java is a statically typed language, which means that variable types are checked at compile time, reducing the likelihood of type-related errors during runtime.

Java's Object-Oriented Foundation

Java is built upon object-oriented principles, including:

Encapsulation: Bundling data and methods within objects to restrict access. It promotes modularity, and data integrity.

Inheritance: Mechanism for creating new classes based on existing ones. It promotes code reuse.

Polymorphism: It allows objects of different types to be treated as objects of a common superclass, enabling code to be written in a more generic and reusable manner. It promotes flexibility. Method overloading, method overriding.

Abstraction: Hiding implementation details and exposing

essential features through interfaces and abstract classes It promotes
simplification of complex systems.

Java Syntax and Language Constructs

Syntax: Java syntax is similar to C and C++, featuring braces () for block structure, semicolons (;) to terminate statements, and keywords for control flow and declarations.

Classes and Objects: Java is class-based, with objects being instances of classes. Classes define attributes (fields) and behaviors (methods) of objects.

Packages: Java organizes classes into packages to facilitate

modularity and namespace management, preventing naming conflicts. Interfaces and Abstract Classes: Java supports interface-based and abstract class-based programming for defining contracts and providing default implementations.

Java Development Tools and Ecosystem

Java Development Kit (JDK): JDK provides the tools and libraries necessary for Java development, including the compiler, runtime environment (JRE), and various utilities.

Integrated Development Environments (IDEs): Popular Java IDEs like Eclipse, IntelliJ IDEA, and NetBeans offer comprehensive features for code editing, debugging, and project management. Build Tools: Tools like Apache Maven and Gradle automate the

build process, managing dependencies and generating executable artifacts. Version Control Systems: Version control systems like Git enable collaborative development and facilitate code management, branching, and merging.

Java Ecosystem and Industry Applications

Enterprise Software Development: Java is widely used for building robust, scalable enterprise applications, including ERP systems, CRM solutions, and financial platforms.

Web Development: Java-based frameworks like Spring, Hibernate, and Struts power web applications, providing features for MVC architecture, ORM, and dependency injection.

Mobile App Development: Android, the world's leading mobile

operating system, relies heavily on Java for developing native Android applications. Whatsapp, Facebook, Instagram

Cloud Computing and Microservices: Java is a popular choice for developing cloud-native applications and microservices, leveraging technologies like Docker, Kubernetes, and AWS.

Definitions of Enterprise-Level Applications I

Enterprise Resource Planning (ERP) Systems: Integrated software systems used to manage core business processes, including finance, human resources, and supply chain, within an organization. Customer Relationship Management (CRM) Solutions: Software platforms designed to manage and analyze customer interactions and data throughout the customer lifecycle to improve customer satisfaction and drive sales growth.

Financial Platforms: Technology solutions utilized for managing financial transactions, investments, and risk in various industries, such as banking, investment management, and insurance.

Human Resource Management (HRM) Systems: Software tools employed to automate and streamline HR processes, including payroll, recruitment, employee onboarding, and performance management.

Definitions of Enterprise-Level Applications II

Supply Chain Management (SCM) Solutions: Software applications used to optimize the flow of goods and services, from procurement to production to distribution, to improve efficiency and reduce costs in the supply chain.

Business Intelligence (BI) Tools: Software platforms that enable

organizations to collect, analyze, and visualize data to gain insights into business operations, make informed decisions, and drive strategic initiatives.

Document Management Systems: Software solutions for storing, organizing, and managing electronic documents and files, facilitating collaboration, version control, and compliance within organizations.

Enterprise-Level Applications of Java I

Enterprise Resource Planning (ERP) Systems:

- Oracle ERP Cloud (Oracle Corporation)
- SAP ERP (SAP SE)
- Workday ERP (Workday, Inc.)

Customer Relationship Management (CRM) Solutions:

Salesforce CRM (Salesforce)

Oracle CRM (Oracle Corporation)

Financial Platforms:

PayPal

Supply Chain Management (SCM) Solutions:

Oracle Supply Chain Management Cloud (Oracle Corporation)

Manhattan Associates (Manhattan Associates)

SAP Supply Chain Management (SAP SE)

Business Intelligence (BI) Tools:

Oracle Analytics Cloud (Oracle Corporation)

Enterprise-Level Applications of Java II

E-Commerce Platforms:

- Amazon
- eBay
- Alibaba

Telecommunication Software:

- WhatsApp

Getting Started with Java: Setting Up Environment

Install JDK Corretto:

JDK Corretto is a no-cost, multiplatform, production-ready distribution of the Open Java Development Kit (OpenJDK)

maintained by Amazon. Download and install JDK Corretto from the official website:
https://aws.amazon.com/corretto/

Install IntelliJ IDEA:

IntelliJ IDEA is a powerful integrated development environment (IDE) for Java development.
Download and install IntelliJ IDEA Community or Ultimate edition from the official website: https://www.jetbrains.com/idea/

Configure IntelliJ IDEA:

Launch IntelliJ IDEA after installation.
Configure JDK Corretto as the default JDK in IntelliJ IDEA:

Topics to be covered in the course I

Java Expressions, Statements, Code Blocks, and Methods

Control Flow Statements

Object-Oriented Programming (OOP) Part 1: Classes, Constructors, and Inheritance

OOP Part 2: Composition, Encapsulation, and Polymorphism

Arrays, Java Built-in Lists, Autoboxing, and Unboxing Inner and

Abstract Classes, Interfaces

Java Generics

Naming Conventions and Packages, Static and Final Keywords

Java Collections

JavaFX

Input Output Including java.util

Topics to be covered in the course II

Concurrency in Java

Lambda Expressions

Regular Expressions

Debugging and Unit Testing

Databases

Java Network Programming