Java SE 25: Programming I Training

Duration: **5 Day**

BİLGİNÇ IT Academy
info@bilginc.com
bilginc.com

Overview

Java SE 25: Programming I is an entry-level course designed to introduce programmers to the fundamental concepts of Java, its core language constructs, and data types. The program also introduces important topics such as exception handling, lambda expressions, and modular programming while focusing on application development with the latest major JDK versions.

Hands-on labs included throughout the course allow learners to put each lesson's concepts into practice and strengthen their Java programming skills through direct coding experience.

Prerequisites

It is recommended that participants have previous experience studying a programming language or possess existing foundational knowledge of Java programming.

Who Should Attend

This course is intended for:

- Developers
- Implementers
- System Integrators
- Systems Administrators

What You Will Learn

After completing this course, learners will be able to:

- Write Java code using variables, arrays, conditional statements, and loop constructs to meet programming objectives.
- Recognize and explain the principles of modular programming.
- Create and access static fields and methods.
- Encapsulate classes by applying access modifiers and overloaded constructors.
- Work with numeric, text, and string data using appropriate Java operators.
- Configure environment variables required for the Java compiler and runtime executables to operate correctly.
- Create basic Java classes and use object references to access class fields and methods.
- Demonstrate polymorphism by implementing Java interfaces.
- Handle checked exceptions within Java applications.
- Pass predicate lambda expressions as arguments to methods.
- Define and implement basic class hierarchies that support application requirements.

Java SE 25: Programming I Training

Duration: **5 Day**

BİLGİNÇ IT Academy
info@bilginc.com
bilginc.com

Outline

Course Introduction

- Course overview
- Course objectives
- Intended audience
- Course structure

Introduction to Java

- Understanding Java
- How Java operates
- Object-oriented programming principles
- Classes and objects
- Inheritance
- Java APIs
- Java keywords, reserved words, and special identifiers
- Java naming conventions
- Fundamental Java syntax rules
- Defining Java classes
- Accessing classes across packages
- Applying encapsulation with access modifiers
- Building a Java program with a main method
- Compiling and running Java programs
- Comments and documentation
- Using code snippets in Javadoc
- External snippets

Primitive Types, Operators, and Flow Control Statements

- Java primitive types
- Declaring and initializing primitive variables
- Restrictions affecting primitive declarations and initialization
- Java operators
- Assignment and arithmetic operators
- Arithmetic operations and type casting
- Additional mathematical operations
- Binary number representation
- Bitwise logical operators
- Equality, relational, and conditional operators
- Short-circuit evaluation
- Controlling program flow with if/else
- Ternary operator
- Flow control with switch
- Switch expressions that produce values
- Working with JShell (REPL)
- JShell commands
- Code snippets

Text, Date, Time, and Numeric Objects

BİLGİNÇ IT Academy

info@bilginc.com

bilginc.com

- String initialization and common operations
- String indexing
- Mutable text objects
- Text blocks
- Wrapper classes for primitive types
- Representing numbers with BigDecimal
- Method chaining
- Local Date and Time API
- Instants, durations, and periods
- Zoned date and time
- Representing languages and countries
- Formatting and parsing numeric, date, and time values
- Localizable resources
- Formatting message patterns

Classes and Objects

- Introduction to UML
- Modeling classes, interactions, and activities
- Designing classes
- Creating objects
- Defining instance variables and methods
- Object creation and access
- Local variables and recursive object references
- Local variable type inference
- Defining constants
- Static context and access
- Combining static and final
- Additional uses of static context
- Introduction to IntelliJ IDE

Improved Class Design

- Method overloading
- Variable numbers of arguments
- Defining and reusing constructors
- Overview of access modifiers
- Defining encapsulation and immutability
- Constants and immutability
- Enumerations and complex enumerations
- Java memory allocation and cleanup
- Parameter passing

Implementing Inheritance and Using Records

- Extending classes
- The Object class
- Reusing parent-class code through inheritance
- Instantiating classes and accessing objects
- Reference type casting rules
- Checking object types before casting
- Pattern matching with instanceof

BİLGİNÇ IT Academy

info@bilginc.com

bilginc.com

- Referencing code within the current or parent object
- Defining subclass constructors
- Flexible constructor bodies
- Class and object initialization
- Method overriding and polymorphism
- Reusing parent-class logic inside overridden methods
- Defining abstract, final, and sealed classes and methods
- Overriding Object class operations: `toString`, `equals`, and `hashCode`
- Comparing String objects
- Java records and custom record constructors
- Record patterns
- Pattern matching for switch
- Factory methods

Interfaces and Generics

- Java interfaces
- The multiple inheritance problem
- Implementing interfaces
- Default, private, and static methods in interfaces
- Interface hierarchies
- Inheritance of default methods
- Using interfaces as types
- Functional interfaces
- Generics and their applications
- Java interface examples: Comparable, Comparator, and Cloneable
- Composition pattern

Arrays and Loops

- Declaring, creating, and initializing arrays
- Multidimensional arrays
- Copying array contents
- The Arrays class
- Loops and processing arrays with loops
- Complex for loops
- Nested loops
- Using break and continue

Collections

- Introduction to the Java Collection API
- Collection API interfaces and implementation classes
- Creating and managing List, Set, and Deque objects
- Creating and managing HashMap objects
- Iterating over collections
- Sequenced collections
- Additional collection behaviors
- Using the `java.util.Collections` class
- Concurrent access to collections
- Preventing collection corruption
- Legacy collection classes

Java SE 25: Programming I Training

Duration: **5 Day**

BİLGİNÇ IT Academy
info@bilginc.com
bilginc.com

Nested Classes and Lambda Expressions

- Types of nested classes
- Static nested classes
- Member inner classes
- Local inner classes
- Anonymous inner classes
- Anonymous inner classes and functional interfaces
- Understanding lambda expressions
- Defining lambda expression parameters and bodies
- Using method references
- Default and static methods in functional interfaces
- Using default and static methods provided by Comparator and Predicate

Java Streams API

- Core characteristics of streams
- Creating streams with the Stream API
- Stream pipeline processing operations
- Using functional interfaces
- Primitive and bi-argument variants of functional interfaces
- Performing actions, filtering, and mapping within stream pipelines
- Combining streams with `flatMap`
- Additional intermediate stream operations
- Custom intermediate stream operations with gatherers
- Short-circuit terminal operations
- Processing streams with `count`, `min`, `max`, `sum`, and `average`
- Aggregating stream data with `reduce`
- General principles of the collect operation
- Using standard collectors
- Converting, grouping, and partitioning stream content
- Mapping and filtering within groups and partitions
- Gatherers
- Parallel stream processing and recommended guidelines
- Limitations of parallel stream processing
- Spliterator and its characteristics

Exception Handling, Logging, and Debugging

- Using the Java Logging API
- Categories of logging methods
- Guarded logging
- Managing and configuring log output
- Understanding Java exceptions
- Creating custom exceptions
- Throwing and catching exceptions
- Exceptions and application execution flow
- Helpful NullPointerExceptions
- Examples of throwing checked and unchecked exceptions
- Handling exceptions
- Automatic resource closure and suppressed exceptions
- Managing exception causes

BILGINÇ IT Academy

info@bilginc.com

bilginc.com

- Java debugger and debugging operations
- Manipulating program data while debugging
- Using assertions to validate program logic
- Program flow with and without exceptions

Exams and Assessments

This course helps learners build the knowledge and programming skills relevant to preparing for the separately available **Java SE 25 Developer Professional** exam.