Getting Started with Programming, OO & Java Basics for Non-Developers

Duration: *5 Day(s)*

Course Overview

Learning to program opens up a world of possibilities, whether you are looking to build applications, improve your problem-solving skills, or just understand how software works. Getting Started with Programming, OO, and Java 21 Basics for Non-Developers is a hands-on, expert-led course designed to make coding approachable, even if you have never written a line of code before. You will learn how programs work, how to think like a developer, and how to write and organize Java code in a way that makes sense. With plenty of hands-on practice, you will gain confidence using Java 21’s latest features, working with IDEs, and understanding key concepts like variables, loops, methods, and object-oriented programming.

This course is built for technically-minded beginners who want real coding experience in a structured, supportive environment. Over five days, you will write your own Java programs, explore core programming concepts, and see firsthand how to build applications from the ground up. You will also work with essential Java tools, handle errors, and learn best practices for writing clean, efficient code. With expert guidance and a focus on hands-on learning, you will walk away with practical skills that you can use immediately. Whether you are starting your journey into development, working with technical teams, or just curious about how programming works, this course will give you the foundation you need to get started.

Becoming a modern software developer is like learning a new language; it requires study, practice, and dedication well beyond this course to apply your new skills effectively. While this five day program won't transform you into an experienced developer, it will lay a solid foundation in coding basics using Java, while teaching you to think like a programmer. Although this course is technical in nature, our instructors will guide you every step of the way, providing a supportive environment for you to explore, ask questions, and prepare for your next learning milestones.

Review this course online at https://www.alta3.com/courses/TT2000

Objectives

- Write and execute Java programs using modern IDEs and gain a basic understanding of coding mechanics.
- Apply object-oriented programming principles to design and structure professional-grade applications.
- Utilize Java's core features to manage data, handle errors, and build efficient programs.
- Understand and adopt best practices for clean and maintainable Java coding.

Who Should Attend

- Aspiring developers from non-technical backgrounds
- Technical professionals collaborating with engineering teams
- Recent college graduates entering programming roles
- Technical managers needing coding knowledge for project oversight

Prerequisites

To ensure a smooth learning experience and to gain the most from attending this course, you should have the following incoming skills:

```
Basic computer literacy: Familiarity with computer operating systems, file management, and general navigation to ensure a smooth learning experience.
Foundational knowledge of IT concepts: Understanding of essential IT terminologies and concepts, such as computer networks, software applications, and data storage.
Analytical thinking: Ability to analyze problems and think critically to develop logical solutions, fostering a programmer's mindset.
```

Course Outline

Overview of Computer Programming

1. Explain what a program is
2. Explain why there are different types of languages
3. Explain what a compiler is
4. Explain what an interpreter is
5. Lab: Matching Terms

Features of a Program

6. Understand what the entry and exit points of an application are
7. Explain what variables are
8. Explain what programming instructions are
9. Explain what errors and exceptions are
10. Understand what programming algorithms are

Software Development Life Cycle

11. Explain the purpose of the software development life cycle
12. Explain what each phase is for
13. Explain the difference between the software development life cycle and a methodology

Thinking in Objects

14. Understand the difference between a class and an object

15. Deconstruct an object into attributes and operations
16. Map an object to a class
17. Define inheritance
18. Lab: Designing an Application

The Java Platform

19. Introduce the Java Platform
20. Explore the Java Standard Edition
21. Discuss the lifecycle of a Java Program
22. Explain the responsibilities of the JVM
23. Executing Java programs
24. Garbage Collection
25. Documentation and Code Reuse

Using the JDK

26. Explain the JDK's file structure
27. Use the command line compiler to compile a Java class
28. Use the command line Java interpreter to run a Java application class
29. Lab: Exploring MemoryViewer

The IntelliJ Paradigm

30. Introduce the IntelliJ IDE
31. The Basics of the IntelliJ interface
32. IntelliJ Projects and Modules
33. Creating and running Java applications
34. Tutorial: Working with IntelliJ 2023.2 (Community Edition)

Writing a Simple Class

35. Write a Java class that does not explicitly extend another class
36. Define instance variables for a Java class
37. Create object instances
38. Primitives vs Object References
39. Implement a main method to create an instance of the defined class
40. Java keywords and reserved words
41. Lab: Create a Simple Class

Adding Methods to the Class

42. Write a class with accessor methods to read and write instance variables
43. Write a constructor to initialize an instance with data
44. Write a constructor that calls other constructors of the class to benefit from code reuse

45. Use the this keyword to distinguish local variables from instance variables
46. Lab: Create a Class with Methods

Exploring Object-Oriented Programming

47. Real-World Objects
48. Classes and Objects
49. Object Behavior
50. Methods and Messages
51. Lab: Define and use a New Java class

Inheritance, Abstraction, and Polymorphism

52. Encapsulation
53. Inheritance
54. Method Overriding
55. Polymorphism
56. Lab: Define and use Another Java Class

Language Statements

57. Arithmetic operators
58. Operators to increment and decrement numbers
59. Comparison operators
60. Logical operators
61. Return type of comparison and logical operators
62. Use for loops
63. Switch Expressions
64. Switch Expressions and yield
65. Lab: Looping (optional)
66. Lab: Language Statements
67. Lab: Switch Expressions

Using Strings and Text Blocks

68. Create an instance of the String class
69. Test if two strings are equal
70. Perform a case-insensitive equality test
71. Contrast String, StringBuffer, and StringBuilder
72. Compact Strings
73. Text Blocks
74. Unicode support
75. Lab: Fun with Strings
76. Lab: Using StringBuffers and StringBuilders

Fields and Variables

77. Discuss Block Scoping Rules
78. Distinguish between instance variables and method variables within a method
79. Explain the difference between the terms field and variable
80. List the default values for instance variables
81. Final and Static fields and methods
82. Lab: Field Test

Specializing in a Subclass

83. Constructing a class that extends another class
84. Implementing equals and toString
85. Writing constructors that pass initialization data to parent constructor
86. Using instanceof to verify type of an object reference
87. Pattern matching for instanceof
88. Overriding subclass methods
89. Safely casting references to a more refined type
90. Lab: Creating Subclasses

Using Arrays

91. Declaring an array reference
92. Allocating an array
93. Initializing the entries in an array
94. Writing methods with a variable number of arguments
95. Lab: Creating an Array

Formatting Strings

96. Format a String using the formatter syntax
97. Apply text formatting
98. Use String.format and System.out.printf
99. Lab: Textblocks

Records

100. Data objects in Java
101. Introduce records as carrier of immutable data
102. Defining records
103. The Canonical constructor
104. Compact constructors
105. Lab: Records

Java Packages and Visibility

106. Use the package keyword to define a class within a specific package
107. Discuss levels of accessibility/visibility
108. Using the import keyword to declare references to classes in a specific package
109. Using the standard type naming conventions
110. Visibility in the Java Modular System
111. Correctly executing a Java application class
112. The Java Modular System
113. Defining Modules

Utility Classes

114. Introduce the wrapper classes
115. Explain Autoboxing and Unboxing
116. Converting String representations of primitive numbers into their primitive types
117. Defining Enumerations
118. Using static imports
119. Deprecating classes and methods
120. Lab: Enumerations

Java Date/Time

121. The Date and Calendar classes
122. Introduce the new Date/Time API
123. LocalDate, LocalDateTime, etc.
124. Formatting Dates
125. Working with time zones
126. Manipulate date/time values

Inheritance and Polymorphism

127. Write a subclass with a method that overrides a method in the superclass
128. Group objects by their common supertype
129. Utilize polymorphism
130. Cast a supertype reference to a valid subtype reference
131. Use the final keyword on methods and classes to prevent overriding
132. Lab: Salaries - Polymorphism

Interfaces and Abstract Classes

133. Define supertype contracts using abstract classes
134. Implement concrete classes based on abstract classes
135. Define supertype contracts using interfaces

136. Implement concrete classes based on interfaces
137. Explain advantage of interfaces over abstract classes
138. Explain advantage of abstract classes over interfaces
139. Lab: Interfaces

Introduction to Exception Handling

140. Introduce the Exception architecture
141. Defining a try/catch blocks
142. Checked vs Unchecked exceptions
143. Lab: Exceptions

Exceptions

144. Defining your own application exceptions
145. Automatic closure of resources
146. Suppressed exceptions
147. Handling multiple exceptions in one catch
148. Enhanced try-with-resources
149. Helpful NullPointerException(s)
150. Lab: Exceptional
151. Lab: Helpful Nullpointers

Building Java Applications

152. Explain the steps involved in building applications
153. Define the build process
154. Introduce build scripts
155. Explain the standard folder layout
156. Resolving project dependencies
157. Tutorial: Importing code Using Maven

Introduction to Generics

158. Generics and Subtyping
159. Bounded Wildcards
160. Generic Methods
161. Legacy Calls To Generics
162. When Generics Should Be Used
163. Lab: DynamicArray
164. Lab: Adding Generics to Dynamic Array

Collections

165. Provide an overview of the Collection API

166. Review the different collection implementations (Set, List and Queue)
167. Explore how generics are used with collections
168. Examine iterators for working with collections
169. Lab: Create a simple Game using Collections