

Java Bootcamp for Software Testers

Course Summary

Description

With the increasing use of Test Driven Development and other collaborative methodologies, like DevOps, that require increased collaboration between programmers and testers, having an understanding of how the Java programming language works from a testing perspective is becoming a necessary skill for software testers.

This course is a hands-on introduction to the Java programming language from a software tester's point of view. The goal of this course is not to teach students how to program in Java but rather to understand how Java is designed to be used and what the common programming and code design failures are most likely to be made by programmers. The Java programming concepts will be presented not from the perspective of writing code, but rather from the perspective of doing critical analysis of existing Java code, and developing strategies for developing tests and test methods that can be used in both structural and functional testing of Java applications.

The course reviews sound and generally accepted best practices for software testing, and demonstrates how they can be "tuned" to be more sensitive to the sort of errors that Java developers and designers are prone to make.

The course walks through the basics of both the Java language, and good OO programming standards and practices, and shows how that knowledge can be used to plan and design tests that identify not only functional problems in the code, but poorly designed or non-standard code that can be the source of errors in the code development.

One of the methodologies that is becoming standard in Java development is the use of test driven development or TDD. Students will be introduced into the TDD methodology through hands on examples which are designed to demonstrate the role the tester plays in these collaborative methodologies like TDD and how to work effectively with testers in these environments.

The course concludes with a look at code reviews and walkthroughs and the most effective roles, practices and contributions that testers can provide in those peer review settings. All of the theory presented is reinforced with examples, real world case studies, hands on exercises and worked examples.

Topics

- Review of Basic Testing Concepts and Practices
- Object Oriented Programming Concepts and Best Practices
- How Java works: Runtime environments, JVMs, bytecode, class libraries, etc.
- Configuration issues and pitfalls for Java
- Java coding best practices
- Writing functional and structural tests for Java code
- Common errors in Java class design and code structure
- Understanding inheritance and common errors made implementing inheritance
- Exception handling and best practices
- Java APIs and Class Libraries
- Code smells and symptoms of code design problems
- Test Driven Development
- Acceptance Test Driven Development
- Code Reviews and Code Walk throughs
- Automated tool for testing and analysis.
- Current best practices and standards

Java Bootcamp for Software Testers

Course Summary (cont'd)

Audience

The course is intended primarily for software testers although it is also appropriate for developers looking to improve the quality of their programming.

Prerequisites

Since this is a course for testers, students should have a basic knowledge of software testing practices and techniques. Some experience with a programming language is helpful, but a basic entry level knowledge of programming concepts, ideas and terminology is assumed.

Duration

Five days

Java Bootcamp for Software Testers

Course Outline

I. Review of basic testing terms and concepts.

This introductory module is intended to establish a common baseline of terms, concepts, and ideas that will be used in the rest of the course. Topics reviewed are coverage, types of testing, basic testing principles. The testing concepts of verification versus validation, completeness, test sensitivity, etc. This and the following module are not intended to teach the testing topics but rather are for the purpose of establishing a common baseline for the rest of the material.

II. Review of basic testing techniques.

Review of test execution, setting testing goals and integration into the development cycle. Testing within the waterfall and agile development processes. Role of the tester and quality with respect to developer. Planning the testing program

III. Object Oriented Development.

This module provides a basic review of the OO development paradigm. How it differs from the structured paradigm (under which software testing was developed as a discipline). Differences in OO versus structured development and systems design and how software testing has modified and extended standard practices and techniques to accommodate those differences.

IV. The Java language and architecture.

Introduction to how the Java language works. The virtual machine, JRE, and class libraries. The distributed and modular architecture of the language and the importance of configuration testing for Java apps. Overview of the "Java ecosystem" and the sandbox model. The Java philosophy and process for managing the distributed model. Common sorts of problems that developers overlook.

V. Java Structured code

In this module, we look at the parts of the java programming language that are similar to most other programming languages: operators, data types, variables, scope. Specific features of Java that are "error prone" such as the difference between the bitwise and logical operators are covered. The basics of loops and flow control are reviewed as well as the specific techniques for testing those constructs.

VI. The Java Class and program structure

This module looks at the basic structure of Java classes, focusing on instance variables and methods. Good Java design techniques are covered such as designing to interfaces, data hiding, encapsulation, and the proper design of methods. Static methods and variables are discussed.

VII. Java Classes, objects, instances and references

In this module we look at the dynamics of using objects in Java: how they are created and referred to with reference variables. We look at the typical errors introduced by shallow versus deep copies and the sorts of typical problems that we see with reference variables, initialization and object versus class scoping of methods and variables. Specifically, we look at test strategies that allow us to identify and detect the sorts of errors that often are introduced by developers in managing object instances.

VIII. Testing Java Class Design

Continuing from the previous module, we examine typical ways Java classes can be poorly designed, and the impact that has on the quality of code. Examples are broken interfaces, bad choices in parameters and return values, and inappropriate modularization. Emphasis is on looking for the code smells that suggest structural problems and test strategies to reveal those problems.

IX. Inheritance.

One of the major issues in moving to OO testing is dealing with the errors that are introduced by the bad use of inheritance. The basics of inheritance in OO languages, with emphasis on Java, is covered along with the best practices for use of inheritance. Bad inheritance design is a very common source of errors but is also one of the more subtle sources of errors as well. This module covers the mechanics of inheritance, overriding methods, shadowing variables, working with superclass methods and variables and the standard best practices for these (for example, this Liskov substitution principle).

X. Testing inheritance

Continuing from the previous module, this one covers the ways to explore and test for problems in inheritance, and how these can interact with the problems also covered in previous modules on testing class design.

Java Bootcamp for Software Testers

Course Outline (cont'd)

XI. Exceptions and exception handling and assertions

This module covers the Java exception handling mechanism and the best practices on how it should be used. The module covers the effective use of try-throw-catch blocks, assertions and exception classes and hierarchies. The problems of nested try blocks and re throwing exceptions are covered.

XII. Testing Exception Handling

Continuing from the previous module, this covers planning a set of exception tests and testing for exception failures, especially in complex exception handling cases.

XIII. Java APIs.

This module is a high level overview of the Java collections API and the Java Reflection API. Typical programmer problems encountered with these APIs are covered as well as suggested test methods.

XIV. The Java Class Libraries

An overview of the main Java class libraries: IO, RMI, Networking, JDBC etc. This module gives the testers a sense of the functionality of each module, typical patterns of use and typical types of testing one would want to cover for the most common types of errors in library use.

XV. Code Smells

This is an introduction into the Test Driven Development and refactoring idea of Code Smells, which are identifiable structural characteristics that usually illustrate poorly designed code. The occurrence of code smells can often identify specific areas of code to test and what sort of testing may be most useful at identifying code problems.

XVI. Test Driven Development

This module is an introduction to the procedures, techniques, and tools of TDD with typical examples of how it is used by developers to write Java code. The emphasis of the module is on the role of the tester and testing practices in TDD and how the quality of the tests used directly affect the quality of the code produced.

XVII. Acceptance Test Driven Development

Similar in structure and approach to the previous module, this module introduced the processes, tools and techniques of ATDD and the role of the tester in developing the acceptance tests using Gherkin as part of the development process

XVIII. Code Reviews and Code Walkthroughs

Peer reviews are an integral part of development processes for producing high quality code. How these work, and how they are structured are covered in the module but from the perspective of the role of the tester and the optimal way testers can contribute when included in the review process.

XIX. Java Standards and Best Practices

There are a wide collection of Java standards and quasi-standards out in the Java community such as Java EE, Spring, Servlets, and many more. This module is a survey of what these are and what established best practice standards exist for the overall Java community