



TIPS AND TRICKS FOR DEBUGGING AND TROUBLESHOOTING

OBJECT ORIENTED PROGRAMMING I

Sercan Külcü | Object Oriented Programming I | 10.01.2023

Contents

Introduction	2
Logging	3
Debugging Tools	4
Best Practices	5
Conclusion	6

Introduction

Debugging is an important skill for any software developer, and Java developers are no exception. Even with the best planning and design, bugs can still find their way into our code. In this chapter, we will explore debugging techniques and best practices for troubleshooting Java code.

Logging

Logging is the process of collecting and storing information about a program's execution. Logging is a useful technique for debugging because it allows developers to review what the program was doing leading up to a crash or bug. Java has a built-in logging API, `java.util.logging`, that allows developers to log information at different levels of severity, from fine-grained debug information to severe error messages.

Debugging Tools

Java provides several debugging tools that can be used to help troubleshoot issues in our code. The most commonly used tool is the Java Debugger (jdb), which is a command-line tool that allows developers to interactively debug their code. Jdb provides several commands, including the ability to set breakpoints, inspect variables, and step through code line by line.

Another useful debugging tool is the Eclipse Integrated Development Environment (IDE). Eclipse provides a graphical user interface for debugging, allowing developers to set breakpoints and inspect variables using a point-and-click interface.

Best Practices

Effective debugging requires more than just knowledge of debugging tools and techniques. It also requires a structured approach to troubleshooting issues in our code. Here are some best practices to follow when debugging Java code:

Reproduce the problem: Before you start debugging, try to reproduce the problem. Reproducing the problem will help you identify the cause of the issue and confirm that any fixes you make have resolved the issue.

Isolate the issue: Once you've reproduced the problem, try to isolate the issue by narrowing down the area of code where the problem is occurring. This can be done by setting breakpoints or adding logging statements to the code.

Use the right debugging tool: Choose the right debugging tool for the job. Command-line tools like jdb are useful for simple debugging tasks, but an IDE like Eclipse is more powerful and provides a more user-friendly interface for debugging.

Analyze the data: When troubleshooting an issue, analyze the data involved in the issue. This includes the input data, output data, and any intermediate data structures used by the code. Understanding the data involved in an issue can help identify the cause of the issue.

Test your fixes: Once you've identified the issue and made changes to your code, test your fixes to confirm that the issue has been resolved.

Conclusion

Debugging and troubleshooting are important skills for Java developers. By using the right tools and following best practices, developers can quickly identify and resolve issues in their code. Effective debugging requires a structured approach, including reproducing the problem, isolating the issue, analyzing the data, and testing the fixes. With these skills and techniques, Java developers can quickly identify and fix issues in their code, leading to more stable and reliable software.