Name: Urvi Patel Date: 26.07.2023
Subject: CORE JAVA UID:22BIT034

DEBUGGING

What is Debugging?

Debugging, in computer programming and engineering, is a multistep process that involves identifying a problem, isolating the source of the problem and then either correcting the problem or determining a way to work around it. The final step of debugging is to test the correction or workaround and make sure it works.

Why do we need Debugging?

Debugging gets started when we start writing the code for the software program. It progressively starts continuing in the consecutive stages to deliver a software product because the code gets merged with several other programming units to form a software product.

Following are the benefits of Debugging:

- Debugging can immediately report an error condition whenever it occurs. It prevents hampering the result by detecting the bugs in the earlier stage, making software development stress-free and smooth.
- It offers relevant information related to the data structures that further helps in easier interpretation.

- Debugging assist the developer in reducing impractical and disrupting information.
- With debugging, the developer can easily avoid complex one-use testing code to save time and energy in software development.

Steps involved in Debugging

Following are the different steps that are involved in debugging:

1. Identify the Error: Identifying an error in a wrong may result in the wastage of time. It is very obvious that the production errors reported by users are hard to interpret, and sometimes the information we receive is misleading. Thus, it is mandatory to identify the actual error.
2. Find the Error Location: Once the error is correctly discovered, you will be required to thoroughly review the code repeatedly to locate the position of the error. In general, this step focuses on finding the error rather than perceiving it.
3. Analyze the Error: The third step comprises error analysis, a bottom-up approach that starts from the location of the error followed by analyzing the code. This step makes it easier to comprehend the errors. Mainly error analysis has two significant goals, i.e., evaluation of errors all

over again to find existing bugs and postulating the uncertainty of incoming collateral damage in a fix.

4. Prove the Analysis: After analyzing the primary bugs, it is necessary to look for some extra errors that may show up on the application. By incorporating the test framework, the fourth step is used to write automated tests for such areas.
5. Cover Lateral Damage: The fifth phase is about accumulating all of the unit tests for the code that requires modification. As when you run these unit tests, they must pass.
6. Fix & Validate: The last stage is the fix and validation that emphasizes fixing the bugs followed by running all the test scripts to check whether they pass.

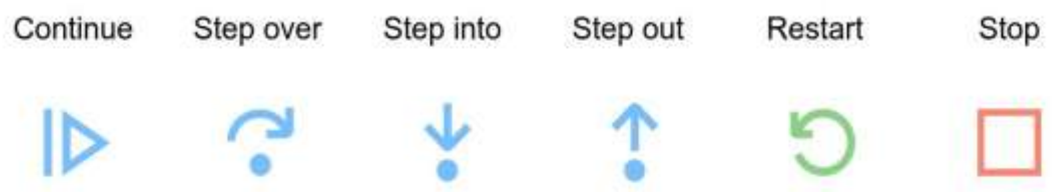
Debugging Strategies/Tools

Breakpoints are a big part of what makes debuggers useful. As their name indicates, they are points you can declare in your code in which the debugger will stop running the program. When the program stops, you'll be able to check all the information as it is in that particular moment.

So breakpoints allow us to see the actual information the program is working with, without the need of logging anything into the console.

You can identify a breakpoint by the little red dots that appear to the left of the line numbers in your code (or by looking into the section mentioned above).

Now when you run the debugger you'll see there's a tiny left arrow on top of the first breakpoint, indicating that's where the program execution has stopped.



- The Continue button runs the program and stops only on user-defined breakpoints.
- With Step Over, if there is a function call, it executes it and returns the result. You don't step into the lines that are inside the function. You just go directly to the function return value.
- Step Into goes inside the function line by line until it returns, and then you go back to the next line right after the function call.
- With the Step Out button, if you have stepped in a function you can skip the remaining execution of the function and go directly to the return value.

- **Restart runs the debugger from the top all over again and Stop exits the debugger.**

So there you go, that's a very powerful debugger built into your code editor. As you can see, with this tool we can check a lot of information at the same time, just by setting breakpoints wherever we want and without the need of any console.logs.

Radare2

Radare2 is known for its reverse engineering framework as well as binary analysis. It is made up of a small set of utilities, either utilized altogether or independently from the command line. It is also known as r2.
It is constructed around disassembler for computer software for generating assembly language source code from machine-executable code. It can support a wide range of executable formats for distinct architectures of processors and operating systems.

WinDbg

WinDbg is a multipurpose debugging tool designed for Microsoft Windows operating system. This tool can be used to debug the memory dumps created just after the Blue Screen of Death that further arises when a bug check is issued. Besides, it is also helpful in debugging the user-mode crash dumps, which is why it is called post-mortem debugging.

Valgrind

The Valgrind exist as a tool suite that offers several debugging and profiling tools to facilitate users in making faster and accurate program. Memcheck is one of its most popular tools, which can successfully detect memory-related errors caused in C and C++ programs as it may crash the program and result in unpredictable behavior.

Testsigma

Testsigma is a leading open-source test automation platform with a unified, fully customizable platform that works out of the box. It allows you to build end-to-end tests 5X faster for web, mobile apps, & APIs with English scripts that self-heal, enabling maintenance-free testing.

Here are some of the reasons why Testsigma is a leading debugging tool –

- **A codeless test automation approach makes it easier for an individual to create quality programs.**

- **The NLP-based approach helps you to create and maintain test cases especially needed for fast-paced continuous testing.**

- **Auto-heals failing test cases with Testsigma's engine finding a probable solution/fix to an issue**

- **Customize your test attributes as per the need and gather only the details which are required.**

- **An open-source unified platform for Android, iOS apps, websites etc.**

- **Offers end-to-end automation including UI, API, DB etc.**

- **Enterprise-level capabilities and data management efficiency, not just restricted to excel sheets.**

Airbrake is a bug-reporting solution which is ideally suited for small and midsize business requirements. It is primarily cloud-based, and a developer-centric tool that can be integrated with the applications as well.

One of the reasons to choose Airbrake is that it can be installed in a matter of few minutes in multiple

languages such as JAVA, Python, Ruby etc. It enables a simple error-monitoring system that makes it very easy to spot all project issues and errors.

There are flexible usage-based pricing with Monthly or Yearly plans based on your requirements.

Here are some reasons Airbrake is one of the best debugging tools available:

- **Real-time error alerts.**
- **You can discover errors right down to the lines of code.**
- **Easy installation.**
- **Low maintenance.**

Chrome DevTools

Being the number one web browser, Google Chrome is the 1st choice for any user. Chrome DevTools, are debugging tools built right into the browser. With these tools, you can edit pages and debug the problems quickly in minimal time, which in turn helps you work and deploy better websites faster.

Now here is why DevTools is considered one of the best –

- **Gives you powerful tools for inspecting and editing your code.**

- **Allows you to measure the performance of your pages with the help of performance panels.**
- **Analyze and review how your pages perform on different browsers and devices.**