## ChatGPT Output

Debugging is an essential skill for any developer. It is the process of identifying and fixing errors or abnormalities in your program. The Eclipse IDE provides robust support for debugging Java applications, allowing you to walk through your code line by line, inspect variables, and monitor the execution flow of your program.

To debug a Java program in Eclipse, follow the steps below:

- Set Breakpoints: Breakpoints are the points where you want your program to pause during execution. This allows you to inspect the state of your application at that point. To set a breakpoint in Eclipse, double-click in the left margin of the line you want to pause at, or right-click in the editor and select "Toggle Breakpoint".
- Start Debug Mode: Right-click on your project or class in the Package Explorer and select "Debug As" > "Java Application". Eclipse will ask if you wish to switch to the Debug Perspective. Click "Switch" to proceed.
- Inspect Variables and Control Execution: Once in Debug Mode, you can step through your code, inspect variables, and control the execution flow. Use the buttons in the Eclipse toolbar to control execution. The most commonly used buttons are F5 (Step Into), F6 (Step Over), and F8 (Resume).
- Inspect Variables: In Debug Mode, you can inspect the values of variables at any point in your program. This is done in the "Variables" view in the Debug Perspective. You can also define custom Java expressions to inspect their values.
- Use Step Filters: Step Filters in Eclipse allow you to control which types of code the debugger will step into. This can be useful to avoid stepping into library code or other types of code you're not interested in debugging. To create a Step Filter, right-click on any item in the Debug view, select "Edit Step Filters", check the "Use Step Filters" box, and select the packages you want to filter out.

Remember, debugging is a powerful tool for understanding the flow and state of your program at any given point. It allows you to find and fix errors that may not be immediately apparent during the coding process.

## Extra Things I Found

One of the variables I set was an integer but entered in as hex.

```
1  int hex_number = 0xAABBCCDD;
```

However, in the variables window it was displayed in decimal. I then realized that you could go into

the `expressions` window and type in a java expression and it would output it. I was able to get it to display the number as hex by using the expression:

```
1  Int.toHexString(hex_number).toUpperCase()
```