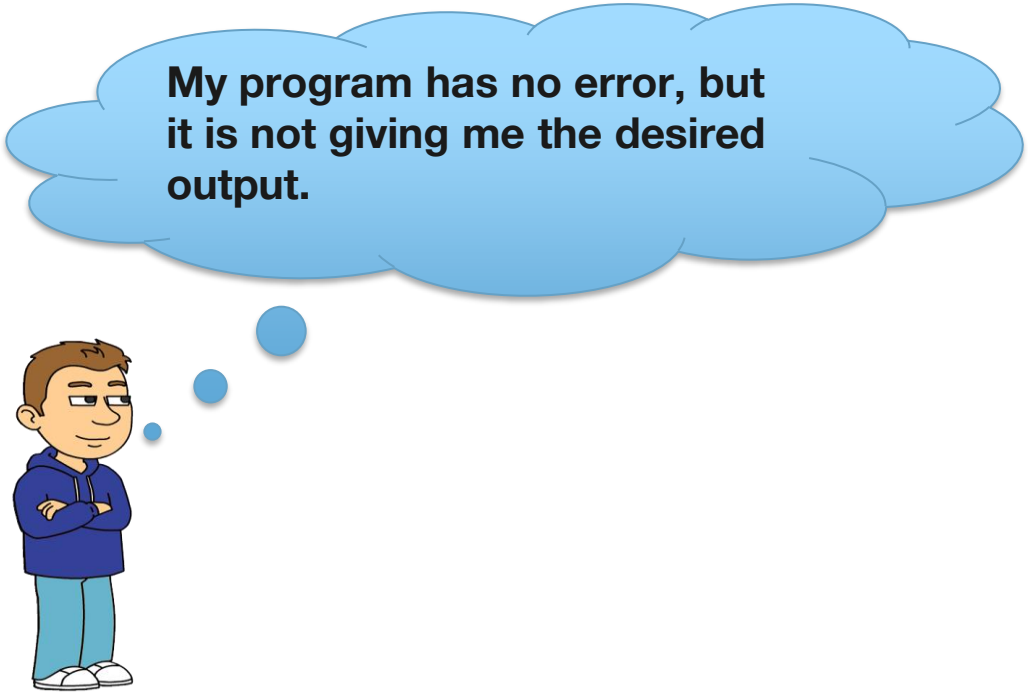
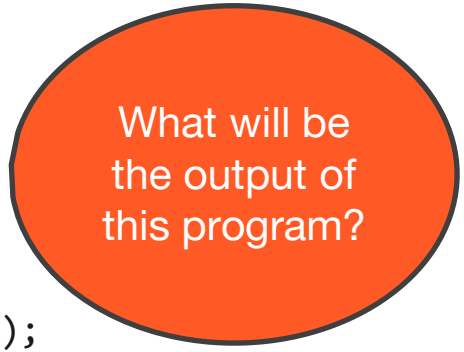
What Should I Do?



Display Perfect Numbers between 1 to 50

```
public class Main {
    public static void main(String[] args) {
        int counter;

        for (counter = 0; counter < 50; counter++);
        {
            System.out.println("The value is " + counter);
        }
    }
}
```



Has the logic been applied correctly?

Sum of Numbers

```
import java.util.Scanner;
public class Main {
    public static void main(String[] aa) {
        Scanner sysin = new Scanner(System.in);
        int number = 0;
        int sum = 0;
        do {
            System.out.println("Enter the number"
);
            number = sysin.nextInt();
            sum = sum - number;

        } while (sum <= 999);
        System.out.println("Stop :: The sum of nu
mbers exceed 999");
    }
}
```

A programmer is trying to calculate the sum of numbers entered till the highest 3-digit number.

Will this program help in achieving the objective?

Catch the Bugs

Debugging Java Program

Learning Objectives

- Explain debugging
- Use print statement for debugging
- List some of the common logical errors in Java
- Identify debugging tips for popular IDEs
- Debug using IDE (IntelliJ)

What Is a Bug?


class Main {
try {
File
= new Fi
"pic.jpg"
long
as
lastModifi
();
FileU
s.touch(
);
long la
= file.lastM
System.out.pri
("File date
ified1));
} catch (IOException
{

Debugging Using Print Statement

- Adding print statements at various stages of a program to debug is one of the most basic techniques used for debugging

- Its major disadvantage is that the programmer has to specify what to display before executing the program

- This technique, however is not one of the best for identifying bugs. It may be useful when there is limited interactivity.

Interactive Demo

- Write a program to display the sum of digits of an integer entered by a user. Debug the logic using a simple print statement.

Common Logical Errors in Java

- Misplacing a semicolon
- Missing the ‘break’ keyword in a Switch-Case Block
- Improper or missing initialization
- Incorrect operator precedence
- Assuming the condition to be true when it is not
- Not releasing the resources after use

Debugging Using IDEs

- All development environments provide a tool known as an interactive debugger, or more simply, a debugger

- Debugger tools help in examining the status of a program that is currently being executed

Tips for Debugging

- Add breakpoints to your code and then use
    - Step into
    - Step over
    - Step out
- Use “view breakpoints” to check the breakpoints
- If you have many variables set up “watches”
- To check the running codes use “evaluate expression”

Interactive Demo

Let us try to

- Debug a Java code using IDE (IntelliJ)
- Set breakpoints
- Watch variables
- Evaluate expressions

Key Takeaways

- Purpose of debugging
- Debug using print statement
- Common logical errors in Java
- Debugging tips for popular IDEs
- Using IntelliJ for debugging