**Day 1**

**Understanding errors in Java**

In java errors occurs when your program fails to compile or program crashes when executing or the program doesn’t produce the desired results. You can encounter all or some these scenarios as a programmer/ developer, and your ability to fix or rectify errors is one of the most important skills you can possess as a programmer.

**Compile Errors /Syntax Errors**

Compiler errors are also called syntax errors, because it means that your code broke the rules of the language.  
Compiler errors can be for things like forgotten semicolons or misspelled variables, but they can also be for violating the rules of Java, like using a non-static variable from a static function.

**Runtime Errors**  
A runtime error (or exceptional event) is a problem that arises during the execution of a program. When a runtime error occurs, the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, a runtime errors are to be handled to avoid program/application crash.

**Activity 1**

A custom exception in Java allows you to define your own exception class. This class extends the Exception class or one of its subclasses to create a specialized exception tailored to your application's needs.  
A custom exception in Java allows you to define your own exception class. This class extends the Exception class or one of its subclasses to create a specialized exception tailored to your application's needs.

**DAY 1 REFLECTIONS**

Errors in Java represent serious, often unrecoverable problems that usually occur at a system level. These issues typically indicate problems that are beyond the control of the application and should not be caught or handled by regular application code.

**Day 2**

**Logic Errors**

Logic errors, often referred to as bugs, occur when the code does not perform as intended due to flaws in the program's logic or algorithm. Unlike syntax errors or runtime errors, logic errors do not result in immediate crashes or error messages. Instead, they cause unexpected or incorrect behavior in the program.

**Debugging**

General debugging techniques  
As said before, debugging is often the realm of ingenuity and uncertainty. Yet a number of tricks can be adopted in the daily programming activity to ease the hunt for problems  
1. Exploiting compiler features  
2. The abused println() debugging technique  
3. Logging

**Activity 1**

Logging in Java involves capturing and recording information about an application's execution. It helps in understanding how the code behaves, identifying issues, and monitoring the flow of the application. Java provides a built-in logging framework through the java.util.logging package, and there are also popular third-party logging libraries like Log4j and Logback.

**Daily Notes - Activity 2**

 NetBeans, as an Integrated Development Environment (IDE), includes a robust debugger tool that helps developers identify and fix issues in their Java applications. The debugger in NetBeans provides various features for code inspection, variable tracking, breakpoints, stepping through code, and more.

**Day 3**

**Common Errors in Java**

 “TimeoutException”  
This Java software error message occurs when a blocking operation times out.  
“NumberFormatException”  
This Java software error message occurs when the application tries to convert a string to a numeric type, but that the number is not a valid string of digits.

**Video Reflection**

 Design patterns are crucial in software development for several reasons:  
-Reusable Solutions: Design patterns offer tested, proven solutions to recurring design problems.  
-Abstraction and Communication: They provide a common language for developers to communicate about design concepts.  
-Scalability and Maintainability: Patterns promote flexible, scalable designs that are easier to maintain.

**DAILY NOTES - FACTORY METHOD PATTERNS**

A factory pattern or factory method pattern defines an interface or abstract class for creating an object, but uses the subclasses to decide which class to use in the following instances:  
when a class doesn't know what subclasses will be required to be created,  
when a class wants the subclasses to specify the objects that are to be created, and  
when the parent classes choose the creation of objects for its subclasses.

**DAILY NOTES - DAY 3 REFLECTIONS**

 In Java, developers commonly encounter various types of errors during coding, testing, and running applications. Some common errors include:  
Syntax Errors: These occur due to mistakes in the syntax of the code. For instance, missing semicolons, incorrect use of brackets, or misspelled keywords can cause syntax errors.  
Logic Errors: Logic errors, also known as bugs, occur when the code does not perform as intended due to flaws in the logic or algorithm. The program runs, but it produces incorrect results or unexpected behaviors.

**Day 4**

**WHAT IS REFACTORING**

Refactoring is a powerful Agile technique that is used by developers to improve existing software. When a system's source code is easily understandable, the system is more maintainable, leading to reduced costs and allowing precious development resources to be used elsewhere.

**WHEN IS REFACTORING USED**

 Ideally, refactoring would be part of a continuous quality improvement process. In other words, refactoring would be seamlessly interwoven with other day-to-day activities of every software developer.

**EXAMPLE OF REFACTORING**  
-Renaming  
This is used when a method, variable, class or other Java item has a name that is misleading or confusing.  
-Moving a class  
This is used when a class is in the wrong package. It should, therefore, be moved to another package where it fits better.  
-Extract method  
This is used when a long method needs to be broken up to enhance readability and maintainability.  
-Extracting a superclass  
This is used when an existing class provides functionality that needs to be modified in some way.

**Activity 1**

 public class Main {  
public static void main(String[] args) {  
try {  
System.out.println(3 / 0); // This line will generate an ArithmeticException  
} catch (ArithmeticException e) {  
System.err.println("ArithmeticException: " + e.getMessage());  
}}}

**DAY 4 REFLECTIONS**

Code refactoring is the process of restructuring existing code without changing its external behavior. It involves improving the design, structure, and readability of the code to make it easier to understand, maintain, and extend while keeping its functionality intact.