**To build in exception and error handling to your application, you can use the following steps:**

Exception and error handling is an important aspect of software development that helps to prevent the application from crashing or producing unexpected results. Exception and error handling involves anticipating potential exceptions or errors and responding appropriately to them to maintain the stability of the application.

**Identifying potential exceptions:** Anticipate the potential exceptions or errors that might occur in the application and identify the source of these exceptions.

**Wrapping critical code with try-catch blocks:** Wrap the critical code with try-catch blocks to catch and handle exceptions.

**Implementing the catch blocks:** Implement the catch blocks to handle the exceptions that were identified in step 1. In the catch block, you can log the exception, display an error message, or take any other appropriate action.

**Throwing custom exceptions:** In addition to handling exceptions, you can also throw custom exceptions to provide more descriptive error messages and improve the debugging process.

Here's an example of how you can build in exception and error handling to your application in Java:

public class MyClass {

public void myMethod() {

try {

// Critical code that might throw an exception

int result = 1 / 0;

} catch (ArithmeticException e) {

// Handle the exception

System.out.println("An ArithmeticException occurred: " + e.getMessage());

}

}

}

Actually, the critical code that might throw an exception is wrapped in a try-catch block. If an Arithmetic Exception occurs, it is caught and handled by the catch block, which prints an error message to the console. By using exception and error handling in this way, you can ensure that the application continues to run smoothly even if an exception occurs.