Question 1:

Write a Java program that demonstrates the use of try-catch blocks to handle exceptions. Use a try block to read an integer from the user, and catch any NumberFormatException that might occur if the input is not a valid integer.

Question 2:

Create a Java method that takes two integers as input and performs division. Handle the ArithmeticException that might occur if the second integer is zero, and display an appropriate error message.

Question 3:

Write a Java program that reads a file name from the user and attempts to open and read the contents of the file using FileInputStream. Handle both FileNotFoundException and IOException appropriately.

Question 4:

Implement a Java method that takes an array of integers as input and calculates the average. Use a custom exception, NoDataException, to handle the case where the array is empty.

Question 5:

Create a Java program that demonstrates the use of a finally block. Open a resource (e.g., a file or a network connection) in the try block, perform some operations, and ensure that the resource is closed properly in the finally block, even if an exception is thrown.

Question 6:

Write a Java function that takes a string as input and converts it to an integer using Integer.parseInt(). Handle the NumberFormatException that might occur if the input is not a valid integer, and provide a default value of 0 in such cases.

Question 7:

Implement a Java program that demonstrates the concept of chained exceptions. Create a custom exception, CustomException, and throw it from a method. Then catch and wrap it in a new RuntimeException, providing additional context information.

Question 8:

Write a Java method that reads a list of numbers from the user until a non-numeric input is entered. Handle the InputMismatchException that might occur when reading numbers.

Question 9:

Create a Java program that simulates a scenario where a resource is acquired (e.g., a database connection or a file) and then explicitly released. Use a try-with-resources block to ensure proper resource management.

Question 10:

Implement a Java method that generates a random number between 1 and 100. If the generated number is greater than 80, throw a custom exception named HighNumberException. Catch this exception and print an appropriate message.