**Exercise 1:**

**Write a Java program that demonstrates handling an ArithmeticException by catching and printing an error message.**

**Answer 1:**

**Exercise 2:**

**Create a program that reads an integer from the user and handles InputMismatchException when the input is not an integer.**

**Answer 2:**

**Exercise 3:**

**Write a program that uses a custom exception class to handle a scenario where a user is too young to access a certain feature (age < 18).**

**Answer 3:**

**Exercise 4:**

**Create a program that reads a file and handles FileNotFoundException when the specified file does not exist.**

**Answer 4:**

**Exercise 5:**

**Write a program that demonstrates a nested exception, where an IOException is caught within an SQLException.**

**Answer 5:**

**Exercise 6:**

**Write a program that demonstrates a NullPointerException.**

**Answer 6:**

**Exercise 7:**

**Create a program that divides two numbers and handles both ArithmeticException and NumberFormatException.**

**Answer 7:**

**Exercise 8:**

**Write a program that demonstrates using the finally block to close resources.**

**Answer 8:**

**Exercise 9:**

**Create a program with a custom exception class to handle a scenario where a user provides an invalid username.**

**Answer 9:**

**Exercise 10:**

**Write a program that demonstrates using multiple catch blocks for different exception types.**

**Answer 10:**

**Exercise 11:**

**Create a program that demonstrates throwing an exception explicitly using the throw keyword.**

**Answer 11:**

**Exercise 12:**

**Write a program that demonstrates chaining exceptions using the initCause method.**

**Answer 12:**

**Exercise 13:**

**Create a program that demonstrates using a throws clause in a method signature.**

**Answer 13:**

**Exercise 14:**

**Write a program that demonstrates rethrowing an exception from a catch block.**

**Answer 14:**

**Exercise 15:**

**Write a program that demonstrates using the try-with-resources statement to automatically close resources.**

**Answer 15:**

**Exercise 16:**

**Create a program that demonstrates using a custom exception hierarchy with inheritance.**

**Answer 16:**

**Exercise 17:**

**Write a program that demonstrates catching multiple exceptions in a single catch block.**

**Answer 17:**

**Exercise 18:**

**Create a program that demonstrates creating and throwing a custom unchecked exception.**

**Answer 18:**

**Exercise 19:**

**Write a program that demonstrates using the assert statement to throw an AssertionError.**

**Answer 19:**

**Exercise 20:**

**Create a program that demonstrates handling a custom exception by rethrowing it as a built-in exception.**

**Answer 20:**

**Exercise 21:**

**Write a program that demonstrates using the getSuppressed method to retrieve suppressed exceptions.**

**Answer 21:**

**Exercise 22:**

**Create a program that demonstrates using a custom exception with constructor overloading.**

**Answer 22:**

**Exercise 23:**

**Write a program that demonstrates using the getStackTrace method to print the call stack.**

**Answer 23:**

**Exercise 24:**

**Create a program that demonstrates creating and throwing a custom checked exception.**

**Answer 24:**

**Exercise 25:**

**Write a program that demonstrates catching an exception using a try block without a catch block but with a finally block.**

**Answer 25:**

**Exercise 26:**

**Write a program that demonstrates creating and throwing a custom runtime exception.**

**Answer 26:**

**Exercise 27:**

**Create a program that demonstrates handling exceptions in multi-level method calls.**

**Answer 27:**

**Exercise 28:**

**Write a program that demonstrates catching an exception using a try block with multiple catch blocks.**

**Answer 28:**

**Exercise 29:**

**Create a program that demonstrates using the printStackTrace method to print exception information.**

**Answer 29:**

**Exercise 30:**

**Write a program that demonstrates using the getCause method to retrieve the root cause of an exception.**

**Answer 30:**

**Exercise 31:**

**Write a program that demonstrates handling exceptions using a try block with a catch block for a specific subclass of an exception.**

**Answer 31:**

**Exercise 32:**

**Create a program that demonstrates catching an exception using a try block with multiple catch blocks in a specific order.**

**Answer 32:**

**Exercise 33:**

**Write a program that demonstrates using the getLocalizedMessage method to retrieve the localized error message of an exception.**

**Answer 33:**

**Exercise 34:**

**Create a program that demonstrates creating and using a chained exception.**

**Answer 34:**

**Exercise 35:**

**Write a program that demonstrates using a custom exception with a constructor that takes a cause.**

**Answer 35:**

**Exercise 36:**

**Create a program that demonstrates throwing an exception inside a loop and catching it outside the loop.**

**Answer 36:**

**Exercise 37:**

**Write a program that demonstrates handling multiple exceptions using a single catch block with instanceof.**

**Answer 37:**

**Exercise 38:**

**Create a program that demonstrates using the fillInStackTrace method to rethrow an exception with a new stack trace.**

**Answer 38:**

**Exercise 39:**

**Write a program that demonstrates handling an exception using a method with a throws clause.**

**Answer 39:**

**Exercise 40:**

**Create a program that demonstrates using a custom exception class with overloaded constructors.**

**Answer 40:**

**Exercise 41:**

**Write a program that demonstrates handling an exception using nested try blocks.**

**Answer 41:**

**Exercise 42:**

**Create a program that demonstrates catching and rethrowing an exception using a catch block.**

**Answer 42:**

**Exercise 43:**

**Write a program that demonstrates catching and rethrowing an exception using a finally block.**

**Answer 43:**

**Exercise 44:**

**Create a program that demonstrates throwing a custom unchecked exception with a cause.**

**Answer 44:**

**Exercise 45:**

**Write a program that demonstrates catching and handling multiple exceptions using a catch block with | (OR) in Java 7.**

**Answer 45:**

**Exercise 46:**

**Create a program that demonstrates throwing an exception inside a loop and catching it within the loop.**

**Answer 46:**

**Exercise 47:**

**Write a program that demonstrates using a custom exception with a toString method.**

**Answer 47:**

**Exercise 48:**

**Create a program that demonstrates using the getStackTrace method to retrieve and print the call stack.**

**Answer 48:**

**Exercise 49:**

**Write a program that demonstrates creating and handling a custom exception with multiple constructors.**

**Answer 49:**

**Exercise 50:**

**Create a program that demonstrates using the initCause method to associate a cause with an exception.**

**Answer 50:**

**Exercise 51:**

**Create a program that demonstrates throwing a custom exception when a user provides an invalid email address.**

**Solution 51:**

**Exercise 52:**

**Write a program that demonstrates throwing a custom exception when a user tries to withdraw more money from an account than the available balance.**

**Solution 52:**

**Exercise 53:**

**Create a program that demonstrates using a user-defined exception for validating a password that must contain at least one uppercase letter.**

**Solution 53:**

**Exercise 54:**

**Write a program that demonstrates throwing a custom exception for a student registration system, where a student must provide a valid student ID.**

**Solution 54:**

**Exercise 55:**

**Create a program that demonstrates throwing a custom exception when a user tries to access a restricted area without proper authorization.**

**Solution 55:**

**Exercise 56:**

**Write a program that demonstrates using a user-defined exception for validating a product code that must start with a letter followed by four digits.**

**Solution 56:**

**Exercise 57:**

**Create a program that demonstrates throwing a custom exception when a user provides an invalid phone number format.**

**Solution 57:**

**Exercise 58:**

**Write a program that demonstrates using a user-defined exception for a library checkout system, where a book must be available to be checked out.**

**Solution 58:**

**Exercise 59:**

**Create a program that demonstrates using a user-defined exception for a ticket booking system, where a customer must provide a valid age.**

**Solution 59:**

**Exercise 60:**

**Write a program that demonstrates using a user-defined exception for a classroom grading system, where a student's grade must be within a valid range.**

**Solution 60:**

All the best.