**Chapter 11**

**Exercises**

**11.14 (Exceptional Conditions)**

List the various exceptional conditions that have occurred in programs throughout this text so far. List as many additional exceptional conditions as you can. For each of these, describe briefly how a program typically would handle the exception by using the exception-handling techniques discussed in this chapter. Typical exceptions include division by zero and array index out of bounds.

ANSWER:

* Division by zero
* Array index out of bounds
* Null pointer exception
* OutOfMemoryError
* StackOverflowError

For handling exceptions:

* Catch the exception using a try-catch block
* Handle the exception by providing a meaningful error message or taking alternative actions
* Optionally, rethrow the exception if it cannot be handled locally

**11.15 (Exceptions and Constructor Failure)**

Until this chapter, we’ve found dealing with errors detected by constructors to be a bit awkward. Explain why exception handling is an effective means for dealing with constructor failure.

ANSWER:

Exception handling is effective for dealing with constructor failure because:

* Constructors cannot return error codes or null values
* Exception handling provides a standardized way to propagate error information from the constructor to the caller
* The caller can then decide how to handle the exception, such as by providing a default value or terminating the program

**11.16 (Catching Exceptions with Superclasses)**

Use inheritance to create an exception superclass (called ExceptionA) and exception subclasses ExceptionB and ExceptionC, where ExceptionB inherits from ExceptionA and ExceptionC inherits from ExceptionB. Write a program to demonstrate that the catch block for type ExceptionA catches exceptions of types ExceptionB and ExceptionC.

**11.17 (Catching Exceptions Using Class Exception)**

Write a program that demonstrates how various exceptions are caught with catch (Exception exception) This time, define classes ExceptionA (which inherits from class Exception) and ExceptionB (which inherits from class ExceptionA). In your program, create try blocks that throw exceptions of types ExceptionA, ExceptionB, NullPointerException and IOException. All exceptions should be caught with catch blocks specifying type Exception.

**11.18 (Order of catch Blocks)**

Write a program demonstrating that the order of catch blocks is important. If you try to catch a superclass exception type before a subclass type, the compiler should generate errors.

**11.19 (Constructor Failure)**

Write a program that shows a constructor passing information about constructor failure to an exception handler. Define class SomeClass, which throws an Exception in the constructor. Your program should try to create an object of type SomeClass and catch the exception that’s thrown from the constructor.

**11.20 (Rethrowing Exceptions)**

Write a program that illustrates rethrowing an exception. Define methods someMethod and someMethod2. Method someMethod2 should initially throw an exception. Method someMethod should call someMethod2, catch the exception and rethrow it. Call someMethod from method main, and catch the rethrown exception. Print the stack trace of this exception.

**11.21 (Catching Exceptions Using Outer Scopes)**

Write a program showing that a method with its own try block does not have to catch every possible error generated within the try. Some exceptions can slip through to, and be handled in, other scopes.