**Section 11.1 Introduction**

• An exception is an indication of a problem that occurs during a program’s execution.

• Exception handling enables programmers to create applications that can resolve exceptions.

#### Section 11.2 Example: Divide by Zero without Exception Handling

• Exceptions are thrown (p. [443](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec1_html#page_443)) when a method detects a problem and is unable to handle it.

• An exception’s stack trace (p. [444](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec1_html#page_444)) includes the name of the exception in a message that indicates the problem that occurred and the complete method-call stack at the time the exception occurred.

• The point in the program at which an exception occurs is called the throw point (p. [445](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec2_html#page_445)).

#### Section 11.3 Example: HandlingArithmeticExceptions andInputMismatchExceptions

• A try block (p. [447](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec3_html#page_447)) encloses code that might throw an exception and code that should not execute if that exception occurs.

• Exceptions may surface through explicitly mentioned code in a try block, through calls to other methods or even through deeply nested method calls initiated by code in the try block.

• A catch block (p. [448](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec5_html#page_448)) begins with keyword catch and an exception parameter followed by a block of code that handles the exception. This code executes when the tryblock detects the exception.

• At least one catch block or a finally block (p. [448](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec5_html#page_448)) must immediately follow the try block.

• A catch block specifies in parentheses an exception parameter identifying the exception type to handle. The parameter’s name enables the catch block to interact with a caught exception object.

• An uncaught exception (p. [449](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec7_html#page_449)) is an exception that occurs for which there are no matching catch blocks. An uncaught exception will cause a program to terminate early if that program contains only one thread. Otherwise, only the thread in which the exception occurred will terminate. The rest of the program will run but possibly with adverse results.

• Multi-catch (p. [449](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec7_html#page_449)) enables you to catch multiple exception types in a single catch handler and perform the same task for each type of exception. The syntax for a multi-catch is:

catch (*Type1* | *Type2* | *Type3* e)

• Each exception type is separated from the next with a vertical bar (|).

• If an exception occurs in a try block, the try block terminates immediately and program control transfers to the first catch block with a parameter type that matches the thrown exception’s type.

• After an exception is handled, program control does not return to the throw point, because the try block has expired. This is known as the termination model of exception handling (p. [449](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec7_html#page_449)).

• If there are multiple matching catch blocks when an exception occurs, only the first is executed.

• A throws clause (p. [450](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec9_html#page_450)) specifies a comma-separated list of exceptions that the method might throw, and appears after the method’s parameter list and before the method body.

#### Section 11.4 When to Use Exception Handling

• Exception handling processes synchronous errors (p.[451](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec10_html#page_451)), which occur when a statement executes.

• Exception handling is not designed to process problems associated with asynchronous events (p. [451](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec10_html#page_451)), which occur in parallel with, and independent of, the program’s flow of control.

#### Section 11.5 Java Exception Hierarchy

• All Java exception classes inherit directly or indirectly from class Exception.

• Programmers can extend the Java exception hierarchy with their own exception classes.

• Class Throwable is the superclass of class Exceptionand is therefore also the superclass of all exceptions. Only Throwable objects can be used with the exception-handling mechanism.

• Class Throwable (p. [451](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec10_html#page_451)) has two subclasses:Exception and Error.

• Class Exception and its subclasses represent problems that could occur in a Java program and be caught by the application.

• Class Error and its subclasses represent problems that could happen in the Java runtime system. Errors happen infrequently and typically should not be caught by an application.

• Java distinguishes between two categories of exceptions (p. [452](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec5_html#page_452)): checked and unchecked.

• The Java compiler does not check to determine if an unchecked exception is caught or declared. Unchecked exceptions typically can be prevented by proper coding.

• Subclasses of RuntimeException represent unchecked exceptions. All exception types that inherit from classException but not from RuntimeException (p. [452](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec5_html#page_452)) are checked.

• If a catch block is written to catch exception objects of a superclass type, it can also catch all objects of that class’s subclasses. This allows for polymorphic processing of related exceptions.

#### Section 11.6 finally Block

• Programs that obtain certain types of resources must return them to the system to avoid so-called resource leaks (p. [454](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec16_html#page_454)). Resource-release code typically is placed in a finally block (p. [454](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec16_html#page_454)).

• The finally block is optional. If it’s present, it’s placed after the last catch block.

• The finally block will execute whether or not an exception is thrown in the corresponding try block or any of its corresponding catch blocks.

• If an exception cannot be caught by one of that tryblock’s associated catch handlers, control proceeds to the finally block. Then the exception is passed to the next outer try block.

• If a catch block throws an exception, the finallyblock still executes. Then the exception is passed to the next outer try block.

• A throw statement (p. [457](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec20_html#page_457)) can throw any Throwableobject.

• Exceptions are rethrown (p. [458](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec21_html#page_458)) when a catch block, upon receiving an exception, decides either that it cannot process that exception or that it can only partially process it. Rethrowing an exception defers the exception handling (or perhaps a portion of it) to another catch block.

• When a rethrow occurs, the next enclosing try block detects the rethrown exception, and that try block’scatch blocks attempt to handle it.

#### Section 11.7 Stack Unwinding and Obtaining Information from an Exception Object

• When an exception is thrown but not caught in a particular scope, the method-call stack is un wound, and an attempt is made to catch the exception in the next outer try statement.

• Class Throwable offers a printStackTrace method that prints the method-call stack. Often, this is helpful in testing and debugging.

• Class Throwable also provides a getStackTracemethod that obtains the same stack-trace information that’s printed by printStackTrace (p. [461](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec24_html#page_461)).

• Class Throwable’s getMessage method (p. [461](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec24_html#page_461)) returns the descriptive string stored in an exception.

• Method getStackTrace (p. [461](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec24_html#page_461)) obtains the stack-trace information as an array of StackTrace Elementobjects. Each StackTraceElement represents one method call on the method-call stack.

• StackTraceElement methods (p. [461](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec24_html#page_461)) getClassName,getFileName, getLineNumber and get MethodName get the class name, filename, line number and method name, respectively.

#### Section 11.8 Chained Exceptions

• Chained exceptions (p. [462](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec8_html#page_462)) enable an exception object to maintain the complete stack-trace in formation, including information about previous exceptions that caused the current exception.

#### Section 11.9 Declaring New Exception Types

• A new exception class must extend an existing exception class to ensure that the class can be used with the exception-handling mechanism.

#### Section 11.10 Preconditions and Postconditions

• A method’s precondition (p. [465](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec28_html#page_465)) must be true when the method is invoked.

• A method’s postcondition (p. [465](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec28_html#page_465)) is true after the method successfully returns.

• When designing your own methods, you should state the preconditions and postconditions in a comment before the method declaration.

#### Section 11.11 Assertions

• Assertions (p. [465](http://proquest.safaribooksonline.com/9780133813036/ch11lev2sec28_html#page_465)) help catch potential bugs and identify possible logic errors.

• The assert statement (p. [466](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec11_html#page_466)) allows for validating assertions programmatically.

• To enable assertions at runtime, use the -ea switch when running the java command.

#### Section 11.12 try-with-Resources: Automatic Resource Deallocation

• The try-with-resources statement (p. [467](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec11_html#page_467)) simplifies writing code in which you obtain a resource, use it in atry block and release the resource in a correspondingfinally block. Instead, you allocate the resource in the parentheses following the try keyword and use the resource in the try block; then the statement implicitly calls the resource’s close method at the end of the tryblock.

• Each resource must be an object of a class that implements the AutoCloseable interface (p. [467](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec11_html#page_467))—such a class has a close method.

• You can allocate multiple resources in the parentheses following try by separating them with a semicolon (;).

### Self-Review Exercises

[**11.1**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec16_html#ch11ans1) List five common examples of exceptions.

[**11.2**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec16_html#ch11ans2) Why are exceptions particularly appropriate for dealing with errors produced by methods of classes in the Java API?

[**11.3**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec16_html#ch11ans3) What is a “resource leak”?

[**11.4**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec16_html#ch11ans4) If no exceptions are thrown in a try block, where does control proceed to when the try block completes execution?

[**11.5**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec16_html#ch11ans5) Give a key advantage of using catch(ExceptionexceptionName).

[**11.6**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec16_html#ch11ans6) Should a conventional application catch Error objects? Explain.

[**11.7**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec16_html#ch11ans7) What happens if no catch handler matches the type of a thrown object?

[**11.8**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec16_html#ch11ans8) What happens if several catch blocks match the type of the thrown object?

[**11.9**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec16_html#ch11ans9) Why would a programmer specify a superclass type as the type in a catch block?

[**11.10**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec16_html#ch11ans10) What is the key reason for using finally blocks?

[**11.11**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec16_html#ch11ans11) What happens when a catch block throws anException?

[**11.12**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec16_html#ch11ans12) What does the statement throw exceptionReferencedo in a catch block?

[**11.13**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec16_html#ch11ans13) What happens to a local reference in a try block when that block throws an Exception?

### Answers to Self-Review Exercises

[**11.1**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec15_html#ch11que1) Memory exhaustion, array index out of bounds, arithmetic overflow, division by zero, in-valid method parameters.

[**11.2**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec15_html#ch11que2) It’s unlikely that methods of classes in the Java API could perform error processing that would meet the unique needs of all users.

[**11.3**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec15_html#ch11que3) A “resource leak” occurs when an executing program does not properly release a resource when it’s no longer needed.

[**11.4**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec15_html#ch11que4) The catch blocks for that try statement are skipped, and the program resumes execution after the last catchblock. If there’s a finally block, it’s executed first; then the program resumes execution after the finally block.

[**11.5**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec15_html#ch11que5) The form catch(Exception exceptionName) catches any type of exception thrown in a try block. An advantage is that no thrown Exception can slip by without being caught. You can handle the exception or rethrow it.

[**11.6**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec15_html#ch11que6) Errors are usually serious problems with the underlying Java system; most programs will not want to catch Errors because they will not be able to recover from them.

[**11.7**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec15_html#ch11que7) This causes the search for a match to continue in the next enclosing try statement. If there’s a finally block, it will be executed before the exception goes to the next enclosing try statement. If there are no enclosing trystatements for which there are matching catch blocks and the exceptions are declared (or unchecked), a stack trace is printed and the current thread terminates early. If the exceptions are checked, but not caught or declared, compilation errors occur.

[**11.8**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec15_html#ch11que8) The first matching catch block after the try block is executed.

[**11.9**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec15_html#ch11que9) This enables a program to catch related types of exceptions and process them in a uniform manner. However, it’s often useful to process the subclass types individually for more precise exception handling.

[**11.10**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec15_html#ch11que10) The finally block is the preferred means for releasing resources to prevent resource leaks.

[**11.11**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec15_html#ch11que11) First, control passes to the finally block if there is one. Then the exception will be processed by a catch block (if one exists) associated with an enclosing try block (if one exists).

[**11.12**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec15_html#ch11que12) It rethrows the exception for processing by an exception handler of an enclosing try statement, after thefinally block of the current try statement executes.

[**11.13**](http://proquest.safaribooksonline.com/9780133813036/ch11lev1sec15_html#ch11que13) The reference goes out of scope. If the referenced object becomes unreachable, the object can be garbage collected.

### 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.

**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.

**11.16 (Catching Exceptions with Superclasses)** Use inheritance to create an exception superclass (calledExceptionA) and exception subclasses ExceptionB andExceptionC, where ExceptionB inherits from ExceptionAand ExceptionC inherits from ExceptionB. Write a program to demonstrate that the catch block for type ExceptionAcatches 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 classExceptionA). 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 typeException.

**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 methodssomeMethod and someMethod2. Method someMethod2 should initially throw an exception. Method someMethod should callsomeMethod2, catch the exception and rethrow it. CallsomeMethod 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.