catch (ExceptionN exVar3) {
    handler for exceptionN;
}

If no exceptions arise during the execution of the try block, the catch blocks are skipped.

If one of the statements inside the try block throws an exception, Java skips the remaining
statements in the try block and starts the process of finding the code to handle the exception.
The code that handles the exception is called the exception handler; it is found by propagating
the exception backward through a chain of method calls, starting from the current method.
Each catch block is examined in turn, from first to last, to see whether the type of the exception
object is an instance of the exception class in the catch block. If so, the exception object
is assigned to the variable declared, and the code in the catch block is executed. If no handler
is found, Java exits this method, passes the exception to the method that invoked the method,
and continues the same process to find a handler. If no handler is found in the chain of methods
being invoked, the program terminates and prints an error message on the console. The process
of finding a handler is called catching an exception.

Suppose the main method invokes method1, method1 invokes method2, method2
invokes method3, and method3 throws an exception, as shown in Figure 13.3. Consider the
following scenario:

I If the exception type is Exception3, it is caught by the catch block for handling
exception ex3 in method2. statement5 is skipped, and statement6 is executed.

I If the exception type is Exception2, method2 is aborted, the control is returned to
method1, and the exception is caught by the catch block for handling exception
ex2 in method1. statement3 is skipped, and statement4 is executed.

I If the exception type is Exception1, method1 is aborted, the control is returned to
the main method, and the exception is caught by the catch block for handling excep-
tion ex1 in the main method. statement1 is skipped, and statement2 is executed.

I If the exception type is not caught in method2, method1, and main, the program
terminates. statement1 and statement2 are not executed.

Call Stack

main method { ... try { ... invoke method1; statement1; } catch (Exception1 ex1) { Process ex1; } statement2; }
method1 { ... try { ... invoke method2; statement3; } catch (Exception2 ex2) { Process ex2; } statement4; }
method2 { ... try { ... invoke method3; statement5; } catch (Exception3 ex3) { Process ex3; } statement6; }
method3

Figure 13.3 If an exception is not caught in the current method, it is passed to its caller. The process is repeated until
the exception is caught or passed to the main method.