Java Software Solutions, 8e (Lewis/Loftus)
Chapter 11 Sample Questions

Multiple-Choice Questions

1) A Java program can handle an exception in several different ways. Which of the following is not a way that a Java program could handle an exception?
A) ignore the exception
B) handle the exception where it arose using try and catch statements
C) propagate the exception to another method where it can be handled
D) throw the exception to a pre-defined Exception class to be handled
E) all of the above are ways that a Java program could handle an exception

Answer: D

Explanation: D) A thrown exception is either caught by the current code if the code is contained inside a try statement and the appropriate catch statement is implemented, or else it is propagated to the method that invoked the method that caused the exception and caught there in an appropriate catch statement, or else it continues to be propagated through the methods in the opposite order that those methods were invoked. This process stops however once the main method is reached. If not caught there, the exception causes termination of the program (this would be answer a, the exception was ignored). However, an exception is not thrown to an Exception class.

2) An exception can produce a "call stack trace" which lists
A) the active methods in the order that they were invoked
B) the active methods in the opposite order that they were invoked
C) the values of all instance data of the object where the exception was raised
D) the values of all instance data of the object where the exception was raised and all local variables and parameters of the method where the exception was raised
E) the name of the exception thrown

Answer: B

Explanation: B) The call stack trace provides the names of the methods as stored on the run-time stack. The method names are removed from the stack in the opposite order that they were placed, that is, the earliest method was placed there first, the next method second, and so forth so that the most recently invoked method is the last item on the stack, so it is the first one removed. The stack trace then displays all active methods in the opposite order that they were called (most recent first).

3) A finally clause will execute
A) only if the try statement that precedes it does not throw an exception
B) only if the try statement that precedes it throws an exception that is caught
C) only if the try statement that precedes it throws an exception that is not caught
D) only if the try statement that precedes it throws an exception, whether it is caught or not
E) in any circumstance

Answer: E

Explanation: E) A finally clause, if specified (because it is optional) will execute no matter what happens with respect to the try and catch statements.
4) Which of the following messages passed to the String str could throw a StringIndexOutOfBoundsException?
A) str.length()
B) str.charAt(2);
C) str.replace('a', 'A');
D) str.equals(str);
E) any of the above could throw a StringIndexOutOfBoundsException
Answer:  B
Explanation:  B) The StringIndexOutOfBoundsException is thrown if a parameter of a String method references a position in the String that is beyond the bounds of the String (i.e. a negative int or an int greater than or equal to the number of characters in the String). This can occur in either charAt or substring methods.

5) In order to have some code throw an exception, you would use which of the following reserved words?
A) throw
B) throws
C) try
D) Throwable
E) goto
Answer:  A
Explanation:  A) The reserved word throw is used to throw an exception when the exception is detected, as in: if (score < 0) throw new IllegalTestScoreException("Input score " + score + " is negative");

6) Assume Exceptionname is a checked exception. If a method uses a class that can generate Exceptionname, then either the method must include try and catch statements where a catch statement catches Exceptionname, or the method header must include the statement
A) throw Exceptionname
B) throws Exceptionname
C) catch Exceptionname
D) catches Exceptionname
E) implements Exceptionname
Answer:  B
Explanation:  B) A checked exception must be caught somewhere and so, if not caught in the method that might generate the exception, the exception must be thrown to whatever called the method. This is accomplished by stating that the method throws Exceptionname.
7) Which of the following is not true of the RuntimeException class?
A) All RuntimeExceptions throw checked exceptions
B) All RuntimeExceptions are Throwable objects
C) RuntimeException has child classes ArithmeticException and NullPointerException
D) RuntimeException objects are not Error objects
E) All of the above are true
Answer: A
Explanation: A) The answers in B, C and D are all true, RuntimeExceptions are Throwable objects and are not part of the Error class, and two types of RuntimeExceptions are ArithmeticException and NullPointerException. Exceptions that are not RuntimeExceptions include various checked exceptions, but RuntimeExceptions are not checked exceptions.

8) System.err is a(n)
A) input stream
B) GUI dialog box that indicates when an error has arisen
C) object
D) Error subclass
E) RuntimeException subclass
Answer: C
Explanation: C) There are three default streams available in Java, System.in, System.out, and System.err. All of these are objects with System.in being an input stream, System.out being an output stream and System.err being an error stream (which is also an output stream).

9) When a program terminates because a thrown exception is not handled, the program
A) starts up again automatically
B) opens a dialog box, which asks the user whether the program should start again, end, or enter a debugging mode
C) saves all output to a disk file called the "runStackTrace.txt"
D) opens a dialog box for the user to specify a disk file name, and all output is stored to that disk file
E) outputs a message indicating what and where the exception was thrown
Answer: E
Explanation: E) If a thrown exception is not caught anywhere in the program, the program terminates, displaying the contents of the run stack trace. The first item on the run stack trace is the exception that was thrown and where it was thrown (the line number of the method of the class where the exception was raised).

10) The difference between a checked and an unchecked exception is
A) checked exceptions need not be listed in a throws clause
B) unchecked exceptions must be listed in a throws clause
C) neither kind of exception follows the rules of exception propagation
D) an unchecked exception requires no throws clause
E) a checked exception always must be caught by a try block; an unchecked exception does not
Answer: D
Explanation: D) A checked exception must either be caught or it must be listed in a throws clause. An unchecked exception requires no throws clause. Both kinds of exceptions follow the rules of exception propagation.
True/False Questions

1) All run-time Errors throw Exceptions.
Answer: FALSE
Explanation: Java classifies any Throwable object as either an Error or an Exception, and so no run-time Errors throw an Exception. Run-time Errors cause termination of the program. Exceptions can be handled so that the program continues executing (but only if the Exception is handled properly).

2) If an exception is thrown and is not caught anywhere in the program, then the program terminates.
Answer: TRUE
Explanation: Exceptions are events of interest or are run-time situations that cannot be handled normally without an exception handler. An exception is caught by an associated exception handler. If the exception is thrown but not handled, then the program must terminate.

3) A try statement must have at least one catch statement, but could have many catch statements, and may or may not have a finally clause.
Answer: TRUE
Explanation: All try statements must have at least one catch statement or there is no reason to have the try statement. There can be one catch statement for each type of exception that might be thrown. The programmer may specify as many catch statements as is felt necessary. Further, a try statement may have a finally clause, but does not need one.

4) Programmers can define their own Exceptions by extending the Exception class or one of the descendants of the Exception class.
Answer: TRUE
Explanation: Java predefines a number of Exception classes so that many of the types of Exceptions that can cause a program to terminate are handled. However, Java allows programmers to add to the language by defining their own Exceptions for whatever unique situations they might encounter. These new Exceptions will either be children of the Exception class, or children of subclasses or descendants of Exception.

5) The following defines a new Exception called ANewException.

```
public Exception ANewException
{
    public ANewException(String message)
    {
        super(message);
    }
}
```

Answer: FALSE
Explanation: The definition is nearly correct, but it must define a class, not an Exception. Exception is a class and ANewException should extend Exception. The header of the class definition should be public class ANewException extends Exception.
6) The difference between the throw reserved word and the throws reserved word is that throw is used within a method body, while throws is used within a method header.
Answer: TRUE
Explanation: Throw is an imperative command that is used within a method to create and throw a new exception. Throws is a compiler directive that tells the compiler that the current method may issue a throw that is not caught within the method and thus may escape to an outer level.

Free Form Questions

1) Explain or provide an example showing how each of the following Exceptions could arise.
   a) ArithmeticException
   b) NullPointerException
   c) NumberFormatException
   d) IndexOutOfBoundsException
   e) IOException
Answer: a) An arithmetic operation could not take place because of an illegal action such as division by zero or square root of a negative number
b) An object was not instantiated (it had the value null) before a message was passed to it
c) An expected value for a numeric operation was not a legal number, such as trying to parse a String value to create an equivalent int where the String contains a non-digit character
d) A numeric parameter used as an index into an object such as an array, String or Vector was out of range, such as charAt(-1)
e) A mishap arose when inputting from keyboard or inputting or outputting to disk such as disk not in disk drive.

2) Rewrite the following method using try and catch statements instead of the throws clause in the method's header.

   public String getInput(String filename) throws IOException
   {
      Scanner infile = new Scanner(new File(filename));
      String response = infile.nextLine();
      return response;
   }

Answer:
public String getInput(String filename)
{
   String response = null;
   try
   {
      Scanner infile = new Scanner(new File(filename));
      response = infile.nextLine();
   }
   catch (IOException ex) { response = null; }
   return response;
}
3) Test scores should fall between 0 and 100. Assume an Exception called TestScoreException has been implemented and imported. Write code to input 10 test scores, compute their average, but throw a TestScoreException in case any inputs violate the proper range.

Answer:
int sum = 0;
int value;
Scanner scan = new Scanner (System.in);

for (int j = 0; j < 10; j++)
{
    System.out.println("Enter the next score");
    value = scan.nextInt();
    if (value < 0 || value > 100) throw new TestScoreException(value + " is out of range");
    else sum += value;
}
double average = (double) sum / 10;

4) By now you almost certainly have run a Java program that has experienced a NullPointerException. Explain the reason(s) that a NullPointerException may be generated.

Answer: NullPointerExceptions occur when a reference is used before the reference has been initialized. More frequently they occur when a reference is used while the reference contains a null value.

5) What are the three standard I/O streams and what purposes do they fulfill?

Answer: The three standard I/O streams are System.in, System.out, and System.err. System.in is the standard system input stream. Most usually it is connected to the keyboard. System.out is the standard system output stream. It usually is connected to the display. Both System.in and System.out are used for "normal" input and output. System.err usually is connected to the standard error output. This, too, usually is connected to the display. It is used for the display of warning and error messages that otherwise might become "lost" due, perhaps, to output redirection.