Chapter 10

Exception handling

Lesson page 10-1. Output of thrown Exceptions and Errors

Activity 10-1-1 Throwing-an-Exception output

Question 1. An exception is an abnormal event that happens during program execution. Java has a system for handling exceptions as instances of class `Throwable` (or its subclasses).

Question 2. `Error`s are events that are typically disasterous and unrecoverable, and thus cause immediate termination of a program, while a thrown `Exception` can be handled by the program.

Question 3. The call stack is the list of methods (or frames for them) that have been called but whose execution has not yet completed.

Question 4. Sort of true: the methods are listed in the reverse order in which they were called.

Lesson page 10-2. The throwable object

Activity 10-2-1 Throwable objects

Question 1. Here's the hierarchy:

```
Throwable
    Exception
        RuntimeException
            ArithmeticException
            ArrayIndexOutOfBoundsException
            NullPointerException
            ...
    Error
            OutOfMemoryError
            ...
```

Question 2. Throw an `Error`.

Question 3. Throw an `Exception`.

Question 4. True.

Question 5. `ArithmeticException`, `NullPointerException`, `ArrayIndexOutOfBoundsException`, `IOException`, and `NumberFormatException`. (You could have listed others.)

Lesson page 10-3. Catching a thrown exception

Activity 10-3-1 The `try` statement

Question 1. The `try` statement has the form:

try <*try-block*>
<*catch-clause*>
...
<*catch-clause*>
finally <*finally-block*>

where a <*catch-clause*> has the form:

catch (<*parameter declaration*>) <*catch-block*>

and the <*parameter declaration*> declares a parameter of (some subclass of) class `Throwable`. If the <*finally-block*> is missing, there must be at least one <*catch-clause*>.

Question 2. The catch-clause has the form

catch (<*parameter declaration*>) <*catch-block*>

where the <*parameter declaration*> declares a parameter of (some subclass of) class `Throwable`.

Question 3. During execution of a try-block, either (1) no object is thrown or (2) an object `a0` (say) is thrown. In the latter case, either (a) instance `a0` can be assigned to the parameter `ae` of the catch clause or (b) it can't be assigned.

Activity 10-3-2 Using the try statement in JLiveWindow

Question 4. Exceptions `ArrayIndexOutOfBounds` and `NumberFormatException` are caught in method `getIntField`.

Question 5. A try-statement can have 0 or more catch-clauses.

Activity 10-3-3 Using the try statement in JLiveRead

Question 6. The try-block executes a return statement.

Activity 10-3-4 Propagation of a thrown object

Question 7. Propagation is the act of sending a thrown object to the first catch-clause clause that will catch it. The discussion of which catch-clause will catch it can be found under entry "propagation" of the ProgramLive glossary.

Question 8. The finally-block of the try-statement is executed (if present), and the object is thrown to the try-statement (which will throw it further).

Question 9. The finally-block is executed when an object is thrown in the try-block and no catch-clause of the corresponding try-statement catches it.

Activity 10-3-5 Exercises on exceptions

Lesson page 10-4. The throw statement

Activity 10-4-1 The throw statement

Question 1. The throw-statement has the form:

throw <*expression*> ;

where the expression yields an instance of (a subclass of) class `Throwable`. Execution of a throw statement "throws" the instance, causing propagation of the instance to a catch-block that catches it.

Question 2. The expresion is:

```
new ArithmeticException("Goobers. I just can't do that!")
```

Activity 10-4-2 Catching and throwing an Exception further.

Question 3. True.

Lesson page 10-5. Checked exceptions . . .

Activity 10-5-1 The throws clause

Question 1. All subclasses of `Throwable` must be caught except: (1) class `Error` and its subclasses and (2) class `RunTimeException` and its subclasses.

Question 2. The throws-clause has the form:

throws <*class name*> , ... , <*class name*>

where each <*class name*> is a (subclass of) `Throwable`.

Question 3. Put the clause **throws** `Exception` on every method.

Lesson page 10-6. Hints on using exceptions

Question 1. The four hints on using exceptions are:

1. Don't overuse exceptions.
2. Use exceptions when the method in which an abormal event occurs is not the best place to handle it.
3. Don't make try-blocks too small.
4. Don't hide exceptions.