Chapter 8: Exceptions and I/O Streams

- Two additional mechanisms for controlling process execution are exceptions and threads
- Chapter 8 focuses on:
  - exception processing
  - catching and handling exceptions
  - creating new exceptions

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Exceptions

- An *exception* is an object that describes an unusual or erroneous situation
- Exceptions are *thrown* by a program, and may be *caught* and *handled* by another part of the program
- A program can therefore be separated into a normal execution flow and an *exception execution flow*
- An *error* is also represented as an object in Java, but usually represents a unrecoverable situation and should not be caught

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Exception Handling

- A program can deal with an exception in one of three ways:
  - ignore it
  - handle it where it occurs
  - handle it an another place in the program
- The manner in which an exception is processed is an important design consideration

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Exception Handling

- If an exception is ignored by the program, the program will terminate and produce an appropriate message
- The message includes a *call stack trace* that indicates on which line the exception occurred
- The call stack trace also shows the method call trail that lead to the execution of the offending line
- See `Zero.java`

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

The `try` Statement

- To process an exception when it occurs, the line that throws the exception is executed within a *try block*
- A try block is followed by one or more *catch* clauses, which contain code to process an exception
- Each catch clause has an associated exception type
- When an exception occurs, processing continues at the first catch clause that matches the exception type
- See `ProductCodes.java`

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

The `finally` Clause

- A try statement can have an optional clause designated by the reserved word `finally`
- If no exception is generated, the statements in the finally clause are executed after the statements in the try block complete
- Also, if an exception is generated, the statements in the finally clause are executed after the statements in the appropriate catch clause complete

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Exception Propagation

- If it is not appropriate to handle the exception where it occurs, it can be handled at a higher level
- Exceptions *propagate* up through the method calling hierarchy until they are caught and handled or until they reach the outermost level
- A try block that contains a call to a method in which an exception is thrown can be used to catch that exception
- See `Propagation.java`

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Exceptions

- An exception is either *checked* or *unchecked*
- A checked exception can only be thrown within a try block or within a method that is designated to throw that exception
- The compiler will complain if a checked exception is not handled appropriately
- An unchecked exception does not require explicit handling, though it could be processed that way

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

The `throw` Statement

- A programmer can define an exception by extending the appropriate class
- Exceptions are thrown using the throw statement:

  `throw` *exception-object*`;`

- See `CreatingExceptions.java`
- Usually a throw statement is nested inside an if statement that evaluates the condition to see if the exception should be thrown

Copyright 1997 by John Lewis and William Loftus. All rights reserved.