Exceptions (part 2)
December 4, 2006

Quick Review of Last Lecture

Exceptions

- An exception is an object that describes an unusual or erroneous situation.

Exception Handling

- Java has a predefined set of exceptions and errors that can occur during execution
- A program can deal with an exception in one of three ways:
  - ignore it
  - handle it where it occurs
  - handle it another place in the program
- The manner in which an exception is processed is an important design consideration

Exception Handling

- If an exception is ignored by the program, the program will terminate abnormally and produce an appropriate message
- The message includes a call stack trace that:
  - indicates the line on which the exception occurred
  - shows the method call trail that lead to the attempted execution of the offending line
- See Zero.java (page 533)
The Exception Class Hierarchy

- Classes that define exceptions are related by inheritance, forming an exception class hierarchy
- All error and exception classes are descendents of the `Throwable` class
- A programmer can define an exception by extending the `Exception` class or one of its descendents
- The parent class used depends on how the new exception will be used

Exception Hierarchy

- `java.lang.Object`
- `java.lang.Throwable`
- `java.lang.Exception`
- `java.lang.RuntimeException`
- `java.lang.ArbithmeticException`

On-line Java Documentation

- [http://java.sun.com/j2se/1.5.0/docs/api/index.html](http://java.sun.com/j2se/1.5.0/docs/api/index.html)

The try Statement

- To handle an exception in a program, the line that throws the exception is executed within a `try block`
- A try block is followed by one or more `catch` clauses
- Each catch clause has an associated exception type and is called an `exception handler`
- When an exception occurs, processing continues at the first catch clause that matches the exception type
The finally Clause

- A try statement can have an optional clause following the catch clauses, designated by the reserved word **finally**
- The statements in the finally clause always are executed
- If no exception is generated, the statements in the finally clause are executed after the statements in the try block complete
- If an exception is generated, the statements in the finally clause are executed after the statements in the appropriate catch clause complete

Examples:
- OutOfBounds.java
- OutOfBounds_Caught.java

Exception Hierarchy

- java.lang.Object
- java.lang.Throwable
- java.lang.Exception
- java.lang.RuntimeException
- java.lang.IndexOutOfBoundsException
- java.lang.ArrayIndexOutOfBoundsException

Examples:
- NullReference.java
- NullReference_Caught.java

Exception Hierarchy

- java.lang.Object
- java.lang.Throwable
- java.lang.Exception
- java.lang.RuntimeException
- java.lang.NullPointerException

Examples:
- ClassCast.java
- ClassCast_Caught.java
Exception Hierarchy

- `java.lang.Object`
- `java.lang.Throwable`
- `java.lang.Exception`
- `java.lang.RuntimeException`
- `java.lang.ClassCastException`

Example:

`ProductCodes.java` (page 536)

Valid Codes

- TRV2475A5R-14
- 4th – 7th pos = district number
- 10th position == zone
  - Zone ‘R’ is banned in district > 2000

Exception Propagation

- An exception can be handled at a higher level if it is not appropriate to handle it where it occurs
  - Exceptions propagate up through the method calling hierarchy until they are caught and handled or until they reach the level of the `main` method
  - A try block that contains a call to a method in which an exception is thrown can be used to catch that exception

Chapter 10
Sections 10.4 -10.6

Exception Propagation

- See `Propagation.java` (page 539)
- See `ExceptionScope.java` (page 540)
Checked Exceptions

- An exception is either checked or unchecked
- A checked exception either must be caught by a method, or must be listed in the throws clause of any method that may throw or propagate it
- A throws clause is appended to the method header
- The compiler will issue an error if a checked exception is not caught or asserted in a throws clause

Unchecked Exceptions

- An unchecked exception does not require explicit handling, though it could be processed that way
- The only unchecked exceptions in Java are objects of type RuntimeException or any of its descendants
- Errors are similar to RuntimeException and its descendants in that:
  - Errors should not be caught
  - Errors do not require a throws clause

THE END