Exception-Handling in Java

Lecture 9
Exceptions

• Java provides a neater, more structured alternative method for dealing with possible errors that can occur while a program is running.
Exceptions

- *Exceptions* are things that are not supposed to occur

- The word “exception” is meant to be more general than “error”.

- Some exceptions (like division by zero) are avoidable through careful programming

- Some exceptions (like losing a network connection) are not avoidable or predictable

- Java allows programmers to define their own means of handling exceptions when they occur
Exception-Handling Mechanism

1. Mechanism for creating special exception classes (whose instances are called `exception objects`)

2. The statement `throw e` is used to signal the occurrence of an exception and return control to the calling method and `e` refers to an exception object

3. The statement `try/catch` allows the calling method to “catch” the “thrown” exception object and take appropriate actions
Exception Example

• The body of a method may call other methods as well as doing its own calculations

• Here the body of m will execute unless an out-of-bounds exception occurs

```java
void m (){  
 try {  
    ... body of m ...  
 }  
 catch (ArrayIndexOutOfBoundsException ae) {  
    ... code to recover from error ...  
 }
```
finally Clause

• When exception is thrown control is transferred to method containing the catch block to handle the exception

• Control does not return to procedure in which the exception was thrown unless it contains a finally clause

• The finally clause can be used to clean up the programming environment after the exceptions has been handled
Finally clause Example

```java
void n() {
    ...
    try { ... open window ... p() ... }
    catch (SomeException se) { ... }
    finally { ... close window ... }
    ...
}
void p() { ... throw se ... }
```
Handling Multiple Exceptions

```java
void m() {
    ...
    try { ... n() ... }
    catch (ArrayIndexOutOfBoundsException ae) { ... }
    catch (NullPointerException npe) { ... }
    ...
}
```
Exception Hierarchy

• Try can catch any exception using the following code

```java
try { ... }
catch (Exception e) {
    ... handle any type of exception ... 
}
```

• You must be careful because Java executes the first catch statement it finds that capable of handling the exception
Which handler is executed?

• In this example the second handler is never executed

```java
try { ... }
catch (Exception e) { ... }
catch (ArrayIndexOutOfBoundsException ae) { ... }
```

• In this example the second handler is only executed if there is no array subscript error

```java
try { ... }
catch (ArrayIndexOutOfBoundsException ae) { ... }
catch (Exception e) { ... }
```
class InvalidIntegerException extends Exception {

    InvalidIntegerException (String s) {
        super(s);
    }

    InvalidIntegerException () {
        this("");
    }
}

Programmer Defined Exceptions
Method Header Throws Clauses

```java
void m() {
    ...
    try { ... N() ... } 
    catch (InvalidIntegerException iie) { ... } 
    ...
}
void n() throws InvalidIntegerException {
    ... p() ... 
}
void p() throws InvalidIntegerException {
    ... throw new InvalidIntegerException(); ... 
}
```