Throwing Exceptions

- A method can ignore the exception
  - It is thrown to where that method was called

```
void myMethod () throws SomeException
  { ... // No try
  } // myMethod
...
try
  { myMethod();
  }
catch (SomeException e)
  ...
```

The method *must* declare that it throws the exception

Handle or Declare Rule

- Programs need not cater for Errors or for Runtime Exceptions

- Programs *must* cater for all other exceptions:
  - *handle* (**`try/catch`**) *or declare* (**`throws`**)

Chapter 10
Input and Output

Basic Concepts

- Output is easy (**`System.out.print`** and **`System.out.println`**)
- Input is a little more complicated!
- I/O mechanisms often throw exceptions

Packages

- Java libraries are organised into *packages*
  - **`java.io`** is the I/O package
- Packages *must* be imported
  - Except for **`java.lang`**

Import all the classes from `java.io`

```
import java.io.*;
public class MyProgram
  { . . .
  } // class MyProgram
```

Other Common Packages

- Basic facilities (strings, etc.)
  - **`java.lang`** (imported automatically)

- GUI facilities:
  - **`java.awt`** (*Abstract Windowing Toolkit*)
  - **`javax.swing`** (newer)

- Utilities:
  - **`java.util`**

I/O Streams

- Conceptually, a program has a “stream” of data flowing in, and a “stream” of data flowing out

Handling Foreign Alphabets

- The stream objects in JDK 1.0 did not do this correctly
- JDK1.1 introduced *readers* and *writers*
  - Streams and readers/writers can be joined

Reading Text

- Connect a **`BufferedReader`** to **`System.in`** (an **`InputStream`**)

```
InputStreamReader isr =
            new InputStreamReader(System.in);
BufferedReader in = new BufferedReader(isr);
```

Reading Text (cont.)

- Reading from a **`BufferedReader`**
  - Use **`readLine`**

```
String s = in.readLine();
while (s != null)
  { // Process the line
    ...
    s = in.readLine(); // Read the next line
  }
```

Idiomatic Form

```
String s;
while ((s = in.readLine()) != null)
  { // Process the line
    ...
  }
```

An assignment

A comparison

Reading Numeric Information

- Read as a **`String`**, then convert

How?

- *Wrapper classes*
  - Provide classes for the primitive types
  - Used for utility functions
  - e.g. **`Double`** **`Integer`**

Wrapper Classes

```
int x = 3; // Primitive
Integer y = new Integer(3); // Wrapper
```

Converting Strings to Numeric Data Types

- Wrapper classes have static methods for conversions
  - e.g. **`Integer.parseInt`**

```
int list[] = new int[100];
int count = 0;
String s;
while ((s = in.readLine()) != null)
  { list[count++] = Integer.parseInt(s);
  }
```