Quick review of last lecture

String Class

- Because strings are so common, we don’t have to use the `new` operator to create a string object
  
  ```java
  title = "Java Software Solutions";
  ```

- This is special syntax that works only for strings

- Each string literal (enclosed in double quotes) represents a `String` object

String Methods

- Once a `String` object has been created, neither its value nor its length can be changed
- Thus we say that an object of the `String` class is immutable
- However, several methods of the `String` class return new `String` objects that are modified versions of the original
- See the list of `String` methods on page 119 and in Appendix M

String Indexes

- It is occasionally helpful to refer to a particular character within a string
- This can be done by specifying the character’s numeric index
- The indexes begin at zero in each string
- In the string “Hello”, the character ‘H’ is at index 0 and the ‘o’ is at index 4
- See `StringMutation.java` (page 120)
String Mutations Example

mutation1=phrase.concat("", except from vending machines.");

mutation2= mutation1.toUpperCase();

mutation3= mutation2.replace('E', 'X');

mutation4= mutation3.substring(3, 30);

Class Libraries

- A class library is a collection of classes that we can use when developing programs.
- The Java standard class library is part of any Java development environment.
- Its classes are not part of the Java language per se, but we rely on them heavily.
- Various classes we've already used (System, Scanner, String) are part of the Java standard class library.
- Other class libraries can be obtained through third party vendors, or you can create them yourself.

The import Declaration

- When you want to use a class from a package, you could use its fully qualified name
  java.util.Scanner

- Or you can import the class, and then use just the class name
  import java.util.Scanner;

- To import all classes in a particular package, you can use the * wildcard character
  import java.util.*;
The import Declaration

- All classes of the `java.lang` package are imported automatically into all programs
- It’s as if all programs contain the following line:
  ```java
  import java.lang.*;
  ```
- That’s why we didn’t have to import the `System` or `String` classes explicitly in earlier programs
- The `Scanner` class, on the other hand, is part of the `java.util` package, and therefore must be imported

Where are the packages located?

- C:\Program Files\Java\jdk1.5.0\src.zip
- The zip file contains all libraries that ship with the java language.

Can you add new packages?

Create a directory `c:\<some_path>\ISU`

In that directory save the file `Cyclone.java`

At the top of `Cyclone.java` put:
```java
package ISU;

public class Cyclone {
  private String msg;
  public Cyclone (String message) {
    msg=message;
  }
  public void printMessage () {
    System.out.println(msg);
  }
}
```

Set your CLASSPATH to `c:\<some_path>\`

Cyclone.java

```java
package ISU;
public class Cyclone {
  private String msg;
  public Cyclone (String message) {
    msg=message;
  }
  public void printMessage () {
    System.out.println(msg);
  }
}
```

TestCyclone.java

```java
import ISU.Cyclone;

public class TestCyclone {
  public static void main(String[] args) {
    Cyclone cy= new Cyclone("Go Cyclones!");
    cy.printMessage();
  }
}
```

Chapter 3
Sections 3.4 - 3.5
The Random Class

• The Random class is part of the java.util package
• It provides methods that generate pseudorandom numbers
• A Random object performs complicated calculations based on a seed value to produce a stream of seemingly random values

Methods in The Random Class

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random ()</td>
<td>Construction: creates a new pseudorandom number generator.</td>
</tr>
<tr>
<td>nextDouble ()</td>
<td>Returns a random number between 0.0 (inclusive) and 1.0 (exclusive).</td>
</tr>
<tr>
<td>nextLong ()</td>
<td>Returns a random number that spans over all possible int values (positive and negative).</td>
</tr>
<tr>
<td>nextInt (int n)</td>
<td>Returns a random number in the range 0 to n-1.</td>
</tr>
</tbody>
</table>

Random Numbers Example

• See RandomNumbers.java (page 126)

The Math Class

• The Math class is part of the java.lang package
• The Math class contains methods that perform various mathematical functions
• These include:
  • absolute value
  • square root
  • exponentiation
  • trigonometric functions

Math Class

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cos (double)</td>
<td>Returns the cosine of a double, an angle.</td>
</tr>
<tr>
<td>sin (double)</td>
<td>Returns the sine of a double, an angle.</td>
</tr>
<tr>
<td>tan (double)</td>
<td>Returns the tangent of a double, an angle.</td>
</tr>
<tr>
<td>sqrt (double)</td>
<td>Returns the square root of a double, a non-negative number.</td>
</tr>
<tr>
<td>pow (double, double)</td>
<td>Returns x raised to the power of y, where x is a double, y is a double.</td>
</tr>
<tr>
<td>log (double)</td>
<td>Returns the natural logarithm of a double, a positive number.</td>
</tr>
<tr>
<td>exp (double)</td>
<td>Returns e raised to the power of a double, where e is Euler's number.</td>
</tr>
<tr>
<td>abs (int)</td>
<td>Returns the absolute value of an int, a non-negative integer.</td>
</tr>
<tr>
<td>abs (long)</td>
<td>Returns the absolute value of a long, a non-negative long.</td>
</tr>
<tr>
<td>abs (float)</td>
<td>Returns the absolute value of a float, a non-negative float.</td>
</tr>
<tr>
<td>abs (double)</td>
<td>Returns the absolute value of a double, a non-negative double.</td>
</tr>
<tr>
<td>max (int, int)</td>
<td>Returns the greater integer of two integers.</td>
</tr>
<tr>
<td>max (long, long)</td>
<td>Returns the greater long of two longs.</td>
</tr>
<tr>
<td>max (float, float)</td>
<td>Returns the greater float of two floats.</td>
</tr>
<tr>
<td>max (double, double)</td>
<td>Returns the greater double of two doubles.</td>
</tr>
<tr>
<td>min (int, int)</td>
<td>Returns the smaller integer of two integers.</td>
</tr>
<tr>
<td>min (long, long)</td>
<td>Returns the smaller long of two longs.</td>
</tr>
<tr>
<td>min (float, float)</td>
<td>Returns the smaller float of two floats.</td>
</tr>
<tr>
<td>min (double, double)</td>
<td>Returns the smaller double of two doubles.</td>
</tr>
</tbody>
</table>

The Math Class

• The methods of the Math class are static methods (also called class methods)
• Static methods can be invoked through the class name – no object of the Math class is needed
  value = Math.cos(90) + Math.sqrt(delta);
• See Quadratic.java (page 129)
• We discuss static methods further in Chapter 6
Run examples from the book

Formatting Output

- Two new classes
  - DecimalFormat
  - NumberFormat

NumberFormat Example

double dollars=5.994;
NumberFormat fmt = NumberFormat.getCurrencyInstance();
System.out.println ( "Price = " + fmt.format(dollars) );

RESULT:
Price = $5.99

Methods in NumberFormat Class

DecimalFormat

- format(double number)
  - Returns a string containing the specified number formatted according to this object's pattern.
- getCurrencyInstance()
  - Returns a NumberFormat object that represents a currency format for the current locale.

Methods in DecimalFormat Class

DecimalFormat

- format(String pattern)
  - Returns a string containing the specified number formatted according to the pattern.
- applyPattern(String pattern)
  - Applies the specified pattern to this DecimalFormat object.
  - Returns a string containing the specified number formatted according to the current pattern.
DecimalFormat Example

double miles = 0.5395;
DecimalFormat fmt = new DecimalFormat("0.###");
System.out.println ( "Miles = " + fmt.format(miles) );
RESULT:
Miles = 0.540
Miles = 0.54

The printf Method

• Provided as a courtesy to C programmers
• System.out.printf("ID: %5d	Name: %s", id, name);

The printf convention

• %d print an int argument in decimal
• %ld print a long int argument in decimal
• %c print a character
• %s print a string
• %f print a float or double argument
• %e same as %f, but use exponential notation
• %g use %e or %f, whichever is better
• %o print an int argument in octal (base 8)
• %x print an int argument in hexadecimal (base 16)
• %p print a single %

Wrapper Classes

• The java.lang package contains wrapper classes that correspond to each primitive type:

<table>
<thead>
<tr>
<th>Primitive Type</th>
<th>Wrapper Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>Bytes</td>
</tr>
<tr>
<td>short</td>
<td>Short</td>
</tr>
<tr>
<td>int</td>
<td>Integer</td>
</tr>
<tr>
<td>long</td>
<td>Long</td>
</tr>
<tr>
<td>float</td>
<td>Float</td>
</tr>
<tr>
<td>double</td>
<td>Double</td>
</tr>
<tr>
<td>char</td>
<td>Character</td>
</tr>
<tr>
<td>boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>void</td>
<td>Void</td>
</tr>
</tbody>
</table>

Integer Class

• The following declaration creates an Integer object which represents the integer 40 as an object

    Integer age = new Integer(40);

• An object of a wrapper class can be used in any situation where a primitive value will not suffice
• For example, some objects serve as containers of other objects
• Primitive values could not be stored in such containers, but wrapper objects could be
Wrapper Classes

• Wrapper classes also contain static methods that help manage the associated type
  For example, the Integer class contains a method to convert an integer stored in a String to an int value:

    num = Integer.parseInt(str);

• The wrapper classes often contain useful constants as well
  For example, the Integer class contains MIN_VALUE and MAX_VALUE which hold the smallest and largest int values

Autoboxing

• Autoboxing is the automatic conversion of a primitive value to a corresponding wrapper object:

    Integer obj;
    int num = 42;
    obj = num;

• The assignment creates the appropriate Integer object
  The reverse conversion (called unboxing) also occurs automatically as needed

Autoboxing Examples

    Integer obj1;
    int num1 = 69;
    obj1 = num1;   // automatically creates an integer object

    Integer obj2= new Integer(69);
    int num2;
    num2 = obj2;   // automatically extracts the int value

THE END