CLASS LIBRARY, FORMATTING, WRAPPER CLASSES

Java Class Library (Packages)

Formatting Output

Wrapper Classes and Autoboxing

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 (Look them up on Sun website)

Other class libraries can be obtained through third party vendors, or you can create them yourself

PACKAGES

The classes of the Java standard class library are organized into *packages*

Some packages in the standard class library are:

Package	Purpose
java.lang	General support
java.applet	Creating applets for the web
java.awt	Graphics and graphical user interfaces
javax.swing	Additional graphics capabilities
java.net	Network communication
java.util	Utilities
javax.xml.parsers	XML document processing

THE IMPORT DECLARATION

When you want to use a class contained in a package, you can use its *fully qualified name*

```
java.util.Scanner scan = ...
```

Or you can *import* the package containing the class and just use the class name `Scanner`

```
import java.util.Scanner;

Scanner scan = ...
```

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:

```
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, so that class must be imported as part of its package

FORMATTING OUTPUT

Look at NumberFormat and DecimalFormat classes in the text

They provide you with ways to output numbers with a predefined precision

For example:

Printing double value of Pi	3.141592…
Printing only 2 decimal digits	3.14

LEADING BLANKS FOR NUMBERS

There is no Java library mechanism to put leading blanks on digit strings to achieve right hand alignment of column of numbers

Need to write nested conditional code:

```
System.out.println( "Number is: " +
       (n<10? "   " + n :
       (n<100? "  " + n :
       (n<1000? " " + n :
        n))));
```

WRAPPER CLASSES

The `java.lang` package contains a *wrapper class* that corresponds to each primitive type:

Primitive Type	Wrapper Class
`byte`	`Byte`
`short`	`Short`
`int`	`Integer`
`long`	`Long`
`float`	`Float`
`double`	`Double`
`char`	`Character`
`boolean`	`Boolean`
`void`	`Void`

WRAPPER CLASSES

The following declaration creates an `Integer` object which is a reference to an object with the integer value 40

```
Integer age = new Integer(40);
```

An object of a wrapper class is used in situations 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 may contain static methods that help manage the associated type

- For example, the `Integer` class contains a method to convert digits stored in a `String` to an `int` value:

```
num = Integer.parseInt(str);
```

Wrapper classes often contain useful constants

- For example, the `Integer` class contains `MIN_VALUE` and `MAX_VALUE` for 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 wrapping a value of 42

The reverse conversion (called *unboxing*) also occurs automatically as needed