

COMP 202 – Week 3

- We now start exploring some key elements of the Java programming language and ways of performing I/O
- This week we focus on:
 - Introduction to objects
 - The String class
 - String concatenation
 - Creating objects
 - Class libraries
 - Wrapper classes
 - Standard I/O
 - Formatting output

COMP 202 - Week 3

1

Introduction to Objects

- Initially, we can think of an *object* as a collection of services that we can tell it to perform for us
- The services are defined by methods in a class that defines the object
- In the Bush program, we invoked the `println` method of the `System.out` object:

```
System.out.println ("We need an energy bill that encourages consumption.");
```

object method Information provided to the method (parameters)

COMP 202 - Week 3

2

The println and print Methods

- The `System.out` object provides another service as well
- The `print` method is similar to the `println` method, except that it does not advance to the next line
- Therefore anything printed after a `print` statement will appear on the same line
- See [Countdown.java](#)

COMP 202 - Week 3

3

Abstraction

- An *abstraction* hides (or ignores) the right details at the right time
- An object is abstract in that we don't really have to think about its internal details in order to use it
- We don't have to know how the `println` method works in order to invoke it
- Therefore, we can write complex software by organizing it carefully into classes and objects

COMP 202 - Week 3

4

The String Class

- Every character string is an object in Java, defined by the `String` class
- Every string literal, delimited by double quotation marks, represents a `String` object
- The *string concatenation operator* (+) is used to append one string to the end of another
- It can also be used to append a number to a string
- A string literal cannot be broken across two lines in a program
- See [Facts.java](#)

COMP 202 - Week 3

5

String Concatenation

- The plus operator (+) is also used for arithmetic addition
- The function that the + operator performs depends on the type of the information on which it operates
- If both operands are strings, or if one is a string and one is a number, it performs string concatenation
- If both operands are numeric, it adds them
- The + operator is evaluated left to right
- Parentheses can be used to force the operation order
- See [Addition.java](#)

COMP 202 - Week 3

6

Escape Sequences

- What if we wanted to print a double quote character?
- The following line would confuse the compiler because it would interpret the second quote as the end of the string

```
System.out.println("I said "Hello" to you.");
```

- An *escape sequence* is a series of characters that represents a special character
- An escape sequence begins with a backslash character (\), which indicates that the character(s) that follow should be treated in a special way

```
System.out.println ("I said \"Hello\" to you.");
```

COMP 202 - Week 3

7

Escape Sequences

- Some Java escape sequences:

<u>Escape Sequence</u>	<u>Meaning</u>
<code>\b</code>	backspace
<code>\t</code>	tab
<code>\n</code>	newline
<code>\r</code>	carriage return
<code>\"</code>	double quote
<code>\'</code>	single quote
<code>\\</code>	backslash

- See [Poem.java](#)

COMP 202 - Week 3

8

Creating Objects

- A variable either holds a primitive type, or it holds a *reference* to an object
- A class name can be used as a type to declare an *object reference variable*

```
String name;
```

- No object has been created with this declaration
- An object reference variable holds the address of an object
- The object itself must be created separately

Creating Objects

- We use the **new** operator to create an object

```
name = new String ("Sparky the clown");
```



This calls the **String constructor**, which is a special method that sets up the object

- Creating an object is called *instantiation*
- An object is an *instance* of a particular class

Creating Objects

- Because strings are so common, we don't have to use the **new** operator to create a **String** object

```
name = "Sparky the clown";
```

- This is special syntax that only works for strings
- Once an object has been instantiated, we can use the *dot operator* to invoke its methods

```
name.length()
```

String Methods

- The **String** class has several methods that are useful for manipulating strings, ex:
 - `String(String str)`, `char charAt(int index)`, `int compareTo(String str)`, `String concat(String str)`, `boolean equals(String str)`, `boolean equalsIgnoreCase(String str)`, `int length()`, `String replace(char oldChar, char newChar)`, `String substring(int offset, int endIndex)`, `String toLowerCase()`, `String toUpperCase()`, ...
- Many of the methods *return a value*, such as an integer or a new **String** object
- See the list of **String** methods in Appendix M
- See [StringMutation.java](#)

Class Libraries

- A *class library* is a collection of classes that we can use when developing programs
- There is a *Java standard class library* that is part of any Java development environment
- These classes are not part of the Java language per se, but we rely on them heavily
- The `System` class and the `String` class are part of the Java standard class library
- 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 of the packages in the standard class library are:

<u>Package</u>	<u>Purpose</u>
<code>java.lang</code>	General support
<code>java.applet</code> (<code>java.awt</code>	Creating applets for the web <i>Graphics and graphical user interfaces</i>)
<code>javax.swing</code>	Additional graphics capabilities and components
<code>java.net</code>	Network communication
<code>java.util</code>	Utilities

The import Declaration

- When you want to use a class from a package, you could use its *fully qualified name*

```
java.util.Random
```

- Or you can *import* the class, then just use the class name

```
import java.util.Random;
```

- To import all classes in a particular package, you can use the ***** wildcard character

```
import java.util.*;
```

COMP 202 - Week 3

15

The import Declaration

- All classes of the `java.lang` package are automatically imported into all programs
- That's why we didn't have to explicitly import the `System` or `String` classes in earlier programs
- The `Random` class is part of the `java.util` package
- It provides methods that generate pseudo-random numbers
- We often have to *scale* and *shift* a number into an appropriate range for a particular purpose
- See [RandomNumbers.java](#)

COMP 202 - Week 3

16

Class Methods

- Some methods can be invoked through the class name, instead of through an object of the class
- These methods are called *class methods* or *static methods*
- The **Math** class contains many static methods, providing various mathematical functions, such as absolute value, trigonometry functions, square root, etc.

```
temp = Math.cos(90) + Math.sqrt(delta);
```

Wrapper classes

- Two categories of data in Java : primitive data and objects.
- We use *wrapper classes* to manage primitive data as objects
- Each wrapper class represents a particular primitive type

```
Integer everest = new Integer(8850);
```

- We have just « *wrapped* » the primitive integer value 8850 into an object referenced by the **everest** variable.
- All wrapper classes are part of the `java.lang` package: **Byte, Short, Integer, Long, Float, Double, Character, Boolean, Void.**

Some methods of the Integer class

- **Integer (int value)**
- **byte byteValue()**
- **double doubleValue()**
- **float floatValue()**
- **int intValue()**
- **long longValue()**
- **static int parseInt (String str)**
- **static String toBinaryString(int num)**

COMP 202 - Week 3

19

The Scanner Class

- **The Scanner class is part of the Java standard class library**
- **Part of the `java.util` package**
- **Contains several methods for reading input values for various types from various sources (keyboard, file)**
- **A Scanner object assumes that white space (delimiters) separates elements of the input (tokens)**
- **See [Echo.java](#)**

COMP 202 - Week 3

20

Some methods of the Scanner class

- `Scanner (InputStream source)`
- `Scanner (File source)`
- `Scanner (String source)`
- `String next ()`
- `String nextLine ()`
- `double nextDouble ()`
- `float nextFloat ()`
- `int nextInt ()`
- `long nextLong ()`
- `short nextShort ()`
- `String nextBoolean ()`
- `Boolean hasNext ()`

COMP 202 - Week 3

21

Formatting Output

- The `NumberFormat` class has static methods that return a formatter object

```
getCurrencyInstance ()  
getPercentInstance ()
```

- Each formatter object has a method called `format` that returns a string with the specified information in the appropriate format
- See [PriceOfHappiness.java](#)

COMP 202 - Week 3

22

Formatting Output

- The `DecimalFormat` class can be used to format a floating point value in generic ways
- For example, you can specify that the number be printed to three decimal places
- The constructor of the `DecimalFormat` class takes a string that represents a pattern for the formatted number
- See [CircleStats.java](#)