Welcome to Com S 227 – day 3

What’s an object?

• A software entity that is a representation or *model* of a thing or concept in the world
• Has *attributes* (its state) and *operations*
  – The operations are called *methods*
• Example: The Java libraries include a pre-defined object called *System.out*
  – *System.out* is a software representation of an *output console* for text
Invoking a method

- The System.out object has a method called `println()`. We invoke the method using the “dot” operator
  - The expression we want to print goes in the parentheses, and is called the *argument* to the method
  - The semicolon at the end is just a required piece of syntax, forming a *statement*:
    ```java
    System.out.println("Hello, world!"); 
    ```

Classes

- The definition for a type of object is called a *class*
  ```java
  public class HelloPrinter {
    ...
  }
  ```
- “HelloPrinter” is an *identifier* (a name we chose for the class)
- `public` and `class` are *keywords* in Java
  - I.e., part of the syntax (grammar) for the language
A simple Java program

• Our class has one method, called main()
  – main() contains one statement
• The entry point for executing a program must always be a method called main with this form

```java
public class HelloPrinter
{
    public static void main(String[] args)
    {
        System.out.println("Hello, world!");
    }
}
```

Language ingredients

• Keywords (reserved words such as public, class, void, ~50 others...)
• Identifiers (names such as HelloPrinter or System)
• Literal values (42, 3.14, “Hello, world!”)
• Operators (+, -, *, /, etc.)
• Syntax rules (“every statement ends with a semicolon”)
Compile errors

• If a syntax rule is violated, the compiler can’t translate the code into machine instructions
  – Compile errors are flagged in Eclipse with red squiggles
• Comments are ignored by the compiler
  – Comments are used to document code
  – See code example for two forms of comments
    • // - style for “internal” comments
    • /** - */ style for “Javadoc” comments

Every value has a type

• 8 primitive types
  – int for whole numbers
  – double for “floating-point” numbers
  – boolean for true/false values
  – (There are 5 others, not used too much)
• Everything else is a kind of object
  – System.out has type PrintStream
  – “Hello” is a literal value of type String
Expressions

• An expression is something that represents a value
  – As opposed to a statement, which means “do this”
• Every literal is an expression
• We can create new expressions by combining them with operators
  
  \[(2 + 3) \times 5\]
  
  \[3.14 / 2.0\]

Examples

• The println() method can be used to print many types of expressions, e.g.
  
  `System.out.println((2 + 3) * 5);`

• (See sample code...)
Variables

• A variable can be used to store a value
• Must be *declared first* with a type
• Value is assigned using the symbol “=“
  – Called the assignment operator, does not mean “equals”!!
  – Works right-to-left only
• Examples (see sample code)

Restrictions on identifiers

• May contain only letters, numbers, and underscores
  – Compiler-generated identifiers may contain ‘$’
• Must start with letter or underscore
Conventions for identifiers

- Variable names start with lowercase letter
- Multiple words use camelCase
- Variable names should be meaningful
- Method names start with lowercase letter
- Class names start with upper case letter