Misc. Stuff and Odd Things
Lecture 9 (7/14/2006)

Useless Fact of the Day
- The smallest independent nation in the world is Vatican City. It is only 0.2 square miles in size, and has a population of 921.

Topics
- Creating variables and constants
- Writing comments
- Odd Things:
  - Integer division, type casting, comparisons, primitive method arguments

Creating a Variable
- Two steps:
  1. Declare the variable (if it's a local variable, declare it at the top of a method; if it's an instance variable, then declare it at the top of the class body): *(not needed if it's a local variable)*
     - `public Point2D.Double myPoint;`
  2. Initialize the variable to some value (do this in a method, normally):
     - `myPoint = new Point2D.Double(3, 7);`
Creating a Constant

- Both steps in one:
  1. Declare and initialize the constant on the same line (usually at the top of the class body):
  
```java
public static final int MY_CONSTANT = 7;
```

Commenting

- Very useful!
- So both you and others can look at your code and understand what you mean it to do very quickly
- Two ways:
  ```java
  // a commented line
  /* a comment which spans more than one line
   * and is also a poem
   * because of this rhyme */
  // a commented line
  ```

Odd Thing #1: Integer Division

- What will “z” be if we initialize it like this:
  ```java
double z = 10 / 9;
```
- It will be “1.0” -- but why?
- Integer division truncates
- Integer division occurs only if both halves of the division are integers -- if either half is a double, then non-truncating double division will happen
  ```java
double z = 10.0 / 9;
```
- ...with the above line, z is initialized to “1.11111111112”

Odd Thing #2: Type Casting

- Methods expect to receive certain types of variables as arguments
- Sometimes, you will have to cast (convert) one variable to a different type in order to give it to a method
- You can convert between the number types (int, double, float), but not between, say, a double and a Point2D.Double
- You convert to a type by putting the desired type in parentheses before the variable/number to be converted: “(int) 5.7” will truncate the .7 and leave you with the integer 5
- As an example, GeneralPath’s “moveTo” method expects two float arguments. Let’s say our variable “g” is a GeneralPath. We can convert integers to floats:
  ```java
g.moveTo((float) 3, (float) 7);
```
Odd Thing #3: Comparison

- You shouldn’t use `==` to compare Strings, or Sprites, or any other object
- It does not check to see if all of the contents of the objects are the same
- For Strings, we can use “s1.equals(s2)” to see if the String s1 is the same as the String s2

Odd Thing #4: Primitive Method Arguments

- When a method takes a *primitive* type of variable as an argument (int, double, char, boolean), it makes a *local copy*
- If the method modifies the passed-in argument, it only modifies this copy, and not the original!
- The only way primitively-typed information can travel *out* of a method is with the method’s return value

Primitive Method Arguments: Example

```java
public int multiplyByThree(int z) {
    z = z * 3;
    return z;
}
```

- Running “someMethod();” will output “2”:
  ```java
  public void someMethod() {
      double x = 2;
      multiplyByThree(x);
      System.out.println(x);
  }
  ```
  this didn’t change x!
  x was copied into the local variable z

- Running “someOtherMethod();” will output “6”:
  ```java
  public void someOtherMethod() {
      double x = 2;
      x = multiplyByThree(x);
      System.out.println(x);
  }
  ```
  this changes x, because we store the returned result of the method call in x