More Math Strings

Chris La Pilla
CompSci 6
Summer 2007

Review: Average

• Many ways of solving same problem (num1, num2, and num3 are ints):
  
  double average = (num1 + num2 + num3)/3.0;
  double sum = num1 + num2 + num3;
  double average = sum/3;
  double average = (num1 + num2 + num3 + 0.0)/3;
  double average = ((double) num1 + num2 + num3)/3;

Casting

• Can convert between primitive types by casting:
  int num = (int) 10.87;  //num is 10
  double num = (double) 10.87;  //num is 10.87

• Casting comes before multiplication, division, and modulo in order of operations

• Casting from double to int is done by truncating (cutting off) the decimal part

• In general, casting can result in a loss of information, so be sure it is what you really need

Math Class

• In addition to the operations covered, there is a class Math that contains methods for other useful mathematical functions

• These methods are called (or invoked) as follows:
  double root = Math.sqrt(x);  //one parameter
  double power = Math.pow(x, y);  //two params

• These methods all take doubles as the input parameters (but you can use ints instead, just as you can assign an int value to a double variable)
Mathematical Methods (cont.)

• Like literals, variables, and expressions, the value returned by a method has a particular type, called the method's return type
• All of these methods return doubles, except:
  - Math.round: returns a long (why not an int?)
  - Math.min and Math.max return an int if both parameters are ints; if either parameter is a double, they return a double
  - Math.abs returns an int if the input is an int and a double if the input is a double

Mathematical Methods (cont.)

• Like literals, variables, and expressions, method calls can be used any place where a value of its return type is expected
• Examples:
  ```java
  int grade = (int) Math.round(0.5 * (x1 + x2));
  double ans = Math.max(0, Math.pow(x + 1, y));
  ```

Strings

• We've already seen String literals:
  "Hello, World!"
• Like ints and doubles, we can have String variables
• As usual, they can be used in place of literals wherever a value of type String is expected:
  ```java
  String hello = "Hello, World!";
  System.out.println(hello);
  ```

Concatenation

• For Strings, + is the concatenation operator
• Concatenation means to put two Strings end to end to yield a new String, e.g.
  ```java
  String hello = "Hello, ";
  String world = "World!";
  System.out.println(hello + world);
  //above prints "Hello, World!"
  ```

Concatenation (cont.)

• If either of the operands of + is a String, the other operand is converted to a String and concatenated:
  ```java
  int age = 30;
  System.out.println("Her age is " + age);
  //prints "Her age is 30"
  ```
• What is printed here?
  ```java
  System.out.println("Sum: " + 10 + 5);
  ```

More on String Literals

• String literals cannot contain:
  - Line breaks:
    ```java
    String test = "line1
    line2"; //error
    ```
  - Double quotes (since these delimit the literal):
    ```java
    String quote = ""Why not?" he asked."; //error
    ```
• What if we need to put these characters into a String?
Escape sequences

- Escape sequences are codes that represent certain characters
- Backslash \ followed by another character
  \n – newline
  \t – tab
  \" – double quotes
  \\ – backslash

Escape sequences examples

System.out.println("She said, \\
"Why?\"");
She said, "Why?"
System.out.println("\n**\n***");
*
**
***

Notes on syntax: Identifiers

- Recall that variables have a type and an identifier
- Valid identifiers may only consist of letters, numbers, and the underscore _
- However, they cannot start with a number
- Valid: num2, my_num, myNum, _num
- Errors: 2num, my\num

Conventions for Identifiers

- Recall: conventions are not enforced by the compiler, but are standard practices to which programmers adhere
- Following conventions helps other Java programmers understand your code more easily
- By convention, variable identifiers should begin with a lower case letter
- For multiple words, capital letters are used at the beginning of each subsequent word
- Examples: grade, grade2, myGrade, meanClassGrade

Constants

- A constant is a variable whose value never changes
- Declare a constant with the final keyword:
  final int PENNIES_PER_DOLLAR = 100;
  final String COURSE_NAME = "CompSci 6";
- Good programming practice to use constants instead of using literals in your code with no explanation
  – These literals are referred to as “magic numbers”
- By convention, constants' identifiers are all caps

User input

- Starter code from yesterday collected user input
- This can easily be done using a Scanner object
- Before it can be used, you have to initialize it as follows:
  Scanner reader = new Scanner(System.in);
- In order to use Scanner, you need to include the line:
  import java.util.Scanner; at the top of your program
Scanner

- Once initialized, you read user input using one of three methods:
  
  ```java
  String line = reader.nextLine();
  int numInt = reader.nextInt();
  double numDouble = reader.nextDouble();
  ```

- These methods pause the program and wait for the user to type something in
- The Scanner does not give the user a prompt, i.e. a message asking for input; you must do this with a separate print statement

Scanner example

```java
// reads an integer
Scanner scanner = new Scanner(System.in);
System.out.print("Enter an integer: ");
int num1 = scanner.nextInt();
```

- A run-time error occurs if the user enters the wrong type of input
- For now, you can assume the user input will always be valid