CS1004: Intro to CS in Java, Spring 2005

Lecture #7: Java expressions II, GUIs

Janak J Parekh
janak@cs.columbia.edu

Administrivia

- HW#1 due now
- HW#2 out this afternoon
  - It will have three "mini" programming assignments

String concatenation, revisited

- Note that the + operator is used for *both* addition and concatenation
- 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
- So what happens when you write "I am " + 10 + 10 + " years old"?

Increment and Decrement

- Also turns out that adding or subtracting one is extremely common, so much so there are special one-operand operators for these tasks
- The *increment operator* (`++`) adds 1 to its operand
- The *decrement operator* (`--`) subtracts 1 from its operand
- The statement

```
count++;
```

  is functionally equivalent to

```
count = count + 1;
```

Increment and Decrement

- The increment and decrement operators can be applied in *postfix form*:

```
count++
```

- or *prefix form*:

```
++count
```

- When used as part of a larger expression, the two forms can have different effects
- Because of their subtleties, the increment and decrement operators should be used with care

Data Conversion

- Sometimes it is convenient to convert data from one type to another
- For example, in a particular situation we may want to treat an integer as a floating point value
- These conversions do not change the type of a variable or the value that's stored in it – they only convert a value as part of a computation

Data Conversion (II)

- Conversions must be handled carefully to avoid losing information
- *Widening conversions* are safest because they tend to go from a small data type to a larger one (such as a `short` to an `int`)
- *Narrowing conversions* can lose information because they tend to go from a large data type to a smaller one (such as an `int` to a `short`)

Data Conversion (III)

- In Java, data conversions can occur in three ways:
  - assignment conversion
  - promotion
  - casting
- *Assignment conversion* occurs when a value of one type is assigned to a variable of another
  - If `money` is a `float` variable and `dollars` is an `int` variable, the following assignment converts the value in `dollars` to a `float`

```
money = dollars
```

  - Only widening conversions can happen via assignment; attempts to narrow trigger a compilation error
  - Note that the value or type of `dollars` did not change

Promotion

- *Promotion* happens automatically when operators in expressions convert their operands
- For example, if `sum` is a `float` and `count` is an `int`, the value of `count` is converted to a floating point value to perform the following calculation:

```
result = sum / count;
```

- Also happens when you concatenate numbers with Strings

Casting

- *Casting* is explicit conversion
- Both widening and narrowing conversions can be accomplished by explicitly casting a value
- To cast, the type is put in parentheses in front of the value being converted
  - Higher precedence than operators
- For example, if `total` and `count` are integers, but we want a floating point result when dividing them, we can cast `total`:

```
result = (float)total / count;
```

- When in doubt, cast!

Interactive Programs

- Programs generally need input on which to operate
- The `Scanner` class provides convenient methods for reading input values of various types
- A `Scanner` object can be set up to read input from various sources, including the user typing values on the keyboard
- Keyboard input is represented by the `System.in` object

Reading Input

- The following line creates a Scanner object that reads from the keyboard:

```
Scanner scan = new Scanner(System.in);
```

- The `new` operator *creates* the `Scanner` object
  - We'll learn much more about *new* soon
- Once created, the `Scanner` object can be used to invoke various input methods, such as:

```
String answer = scan.nextLine();
```

Reading Input

- The `Scanner` class is part of the `java.util` class library, and must be *imported* into a program to be used
  - `import java.util.Scanner;` at the top of your code
- The `nextLine` method reads all of the input until the end of the line is found
- The details of object creation and class libraries are discussed further in Chapter 3

Input Tokens

- What if you want to input multiple values into separate variables?
- Unless specified otherwise, *white space* is used to separate the elements (called *tokens*) of the input
- White space includes space characters, tabs, new line characters
- The `next` method of the `Scanner` class reads the next input token and returns it as a string
- Methods such as `nextInt` and `nextDouble` read data of particular types

Let's put it all together

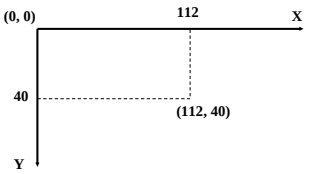
- Simplest example: write a Fahrenheit-to-Celsius converter
- Steps?
  - Create two variables, one to hold the temperature in F and another to hold it in C
  - Get temperature input from user
  - Do the math and store the result in the second variable
  - Print out the result
- You tell me what to write

Introduction to Graphics

- Book reinforces some of the concepts through graphics examples at the end of each chapter
- A picture is made up of *pixels* (picture elements), and each pixel is stored separately
  - The number of pixels used to represent a picture is called the *picture resolution*
  - The number of pixels that can be displayed by a monitor is called the *monitor resolution*
- Each pixel can be identified using a two-dimensional coordinate system

Coordinate Systems

- When we use a coordinate system with the origin in the top-left corner



Representing Color

- A black and white picture could be stored using one bit per pixel (0 = white and 1 = black)
- A colored picture requires more information; there are several techniques for representing colors
  - For example, every color can be represented as a mixture of the three additive primary colors Red, Green, and Blue
  - Each color is represented by three numbers between 0 and 255 that collectively are called an *RGB value*

The Color Class

- A color in a Java program is represented as an object created from the `Color` class
- The `Color` class also contains several predefined colors, including the following:

Object	RGB Value
`Color.black`	`0, 0, 0`
`Color.blue`	`0, 0, 255`
`Color.cyan`	`0, 255, 255`
`Color.orange`	`255, 200, 0`
`Color.white`	`255, 255, 255`
`Color.yellow`	`255, 255, 0`

Applets

- All the programs we've written so far are Java *applications*
- A Java *applet* is a program that is intended to be transported over the Web and executed using a web browser
  - An applet also can be executed using the **appletviewer** tool in the JDK
- An applet doesn't have a `main` method
  - Instead, there are several special methods that serve specific purposes

Applets

- The `paint` method, for instance, is executed automatically and is used to draw the applet's contents
- The `paint` method accepts a parameter that is an object of the `Graphics` class
- A `Graphics` object defines a *graphics context* on which we can draw shapes and text
- The `Graphics` class has several methods for drawing shapes

Why applets?

- You can write programs your friends can access without installing the full JDK or having a CUNIX account
  - *Although* they may need the Java plug-in (available from www.java.com)
  - If you install the JDK on your home computer, it installs this automatically
- Easy to set up graphical programs
  - You can create a graphical Java application, but it turns out to be more work
- Applets have limitations to prevent security problems
  - Won't be a problem for this homework

How to create an Applet

- Your class must *extend* the `Applet` class
  - This makes use of *inheritance* (Chapter 8)
  - You don't need to know how this works in order to write applets
- Next, embed the applet into an HTML file using a tag that references the class file of the applet
- View the HTML file using a web browser or appletviewer
  - The web browser can automatically download the .class file like an image

Next time

- Finish applets
- Basic circuit design and computer architecture