More Java basics

Java syntax

Input and Output

September 22, 2003

What we've done so far

- Seen the basics of objects and Object-Oriented programming
- Looked at a few examples of Java programs
- Picked out some of the parts of Java programs
- Learned about two special methods, `main` and the constructor
- Written our own method for the Account class
- Seen different types of comments
- Figured out how to navigate around Linux in the lab

Today

- A bit more about classes, methods, and variables
  - some of this will be review from last week
- Input and output: 2 methods
  - screen ("console")
  - windows (JOptionPane and JFrame)

Classes, objects, methods, and variables

For the next few slides, we will refer to the programs SimpleShapes.java and Account.java
Classes: 2 types

- **Standard classes**
  - a.k.a predefined classes
  - the ones that Java provides
  - e.g. System.out, JFrame

- **Programmer-defined classes**
  - the classes that we write
  - e.g., Account, SimpleShapes

Classes can contain objects of other classes

- Example: SimpleShapes contains several objects from other classes
  JFrame win, Graphics g

- SimpleShapes is *dependent* on JFrame and Graphics, because it uses its methods
  - JFrame: setSize(), setLocation(), setVisible(), ...
  - Graphics: setColor(), fillRect(), fillOval(), ...
  - Wu: SimpleShapes "sends messages to" JFrame and Graphics

Class name vs. object names

- **Class names**: Start with a capital letter
  - Good: Account, BankAccount, CheckingAccount, SavingsAccount, RetirementAccount
  - Bad: 2Account, #$%@Java, bankaccount, bank*account

- **Object names**: Start with a lowercase letter
  - Good: account, acct, bankAcct, checkingAccount
  - Bad: BankAccount, #&haha, 2good, bankaccount

- **Remember that Java is case sensitive**

Creating an object from a class

- **Use the new keyword**
  - account = new Account("Jane Doe", 2000);
  - win = new JFrame("Simple Shapes Example");

- **Calls the constructor for the class**
Methods

- Method = action
  - method = message sent to a class (Wu)
- Syntax: `<object>..<method>(<arguments>)`
  - `win.setSize(500,500);`
  - `g.getColor();`
  - `acct.deposit(50);`
- “perform <method> action in <object> class with <arguments> this data”
- You can tell a method from a variable because it will contain parentheses

Methods (cont.)

- `<arguments>` could be blank
  - `g.getColor();`
- Multiple arguments are separated by commas
- Methods will sometimes return a value, but not always
  - more on this later!
- Method names: start with a lowercase letter (same rules as for object names)

Variables

- Data used by an object
  - double balance;
  - String name;
- Consists of `<type> <name>`
- Variable names: same rules as for method names and object names
- More in the next chapter

A brief diversion: Strings

- String `string1 = “This is a string”;`
- Stored as an array
  - position 0: T
  - position 1: h
  - position 2: i
  - ...
  - position 15: g
- Length = 16 characters
String methods you should know

- **length()**: returns the number of characters in the string (spaces included)
- **substring(x,y)**: returns all the string's characters between position x and position y in the array
- **indexOf(“foo”)**: returns the position in the array of the first character of “foo” (case-sensitive!)
- **Concatenation**: assemble a string from other strings

Examples

- `string1 = “bah humbug!”, string2 = “too much homework!”;
- Value of `string1.length()` is 9
- Value of `string2.length()` is 18
- `string3 = string1 + “ “ + string2;
  - string3 is now
    “bah humbug! too much homework!”
- Value of `string1.substring(4,9)` is “humbug”
- Value of `string2.indexOf(“homework”)` is 9

Input and output

- Several ways to do input and output
  - screen (console)
  - graphical windows
  - files
  - ...
- For now, we'll concentrate on the first 2
  - files much later

Console output

- We saw this in the Account class:
  - `System.out.println(“Account holder: ”+acct.name);
  - `System.out.println(“Account balance: $”+acct.balance);
- “console” = “command window” (Windows), “terminal window” (Linux, Mac)
- We'll use the term **standard output (stdout)** to refer to this type of output
Writing to stdout

- The object `System.out` handles the task of writing the output to the screen
  - Automatically created by Java
- `System.out` has two possible methods:
  - `print()`: prints text to the screen with no newline at the end
  - `println()`: prints text to the screen with a newline at the end

System.out.println()'s arguments

- Can be a string
  - `System.out.println("Hello, Newman!");`
- Can be a non-string (e.g., number) and/or variable
  - `int x = 56; System.out.println(x);`
  - Java automatically converts non-strings to strings in this case
- Can be a combination of the two
  - `System.out.println("Account balance: "+acct.balance);`

Console input

- *Standard input (stdin)*
- `System.in` is the Java object that handles reading in data from the console
  - automatically created
- Usage is a bit more complicated than for stdout
  - `System.in` reads in one byte at a time
  - we want to read in more than one byte at a time, typically

Reading a line from stdin

- Step 1: Create an `InputStreamReader` object
  - reads in one character at a time (4 bytes)
- Step 2: Create a `BufferedReader` object
  - reads in a single line of text (up to a carriage return)
- Step 3: Use the `readLine()` method to read text from stdin
  - store this text in a variable
Example

```java
import java.io.*;
...
public static void main(String[] args) {
    InputStreamReader in = new InputStreamReader(System.in);
    BufferedReader br = new BufferedReader(in);
    String name, ageStr;
    System.out.println("What is your name?");
    name = br.readLine();
    System.out.println("What is your age?");
    ageStr = br.readLine();
}
```

Reading in numerical data

- In the example, we read in a number (age)
- Everything is read in as a string
- We need to convert numbers “manually”
- For integers, we use `Integer.parseInt(...)`
- e.g., add to the end of the previous example
  ```java
  int age = Integer.parseInt(ageStr);
  ```

One more thing to add

- Sometimes, reading in data from stdin could result in an error
- In Java, we need to “catch” errors where they might occur
  - lets the program continue to execute, even if something goes wrong
  - called exception handling
  - will talk about this much more later
- The error that could occur here is called an `IOException`

Our code should look like this

```java
import java.io.*;
...
public static void main(String[] args) throws IOException {
    InputStreamReader in = new InputStreamReader(System.in);
    BufferedReader br = new BufferedReader(in);
    String name, ageStr;
    System.out.println("What is your name?");
    name = br.readLine();
    System.out.println("What is your age?");
    ageStr = br.readLine();
    int age = Integer.parseInt(ageStr);
}
```
What did we just do?

- When we throw an exception, we pass the responsibility of handling the error to the next level up
  - in this case, to the system on which we're running
- We could also catch the exception, and just handle the error here
  - we'll do this later

Graphical output

- We'll call this window output
- Send output to a pop-up window on the screen
- There are several ways to do this:
  - use a JFrame (as in SimpleShapes.java)
  - use a JDialog (opens a “dialog window”)
- We'll use the second method for now

Example

```java
import javax.swing.*;
...
public static void main(String[] args) {
    JOptionPane.showMessageDialog(null, "This is a dialog window in the center of the screen");
}
```

Another example

```java
import javax.swing.*;
...
public static void main(String[] args) {
    JFrame win = new JFrame();
    win.setSize(200, 200);
    win.setVisible(true);
    JOptionPane.showMessageDialog(win, "This is a dialog window at a specific screen location");
}
```
Window input

- Retrieve input from a pop-up window on the screen
  - very similar to window output
- Use a dialog window
  - JOptionPane.showInputDialog()
  - (other methods too)

Example

```java
import javax.swing.*;
...
public static void main(String[] args) {
    String name, ageStr;
    name = JOptionPane.showInputDialog(null, "What is your name?");
    ageStr = JOptionPane.showInputDialog(null, "What is your age?");
    int age = Integer.parseInt(ageStr);
}
```