Java Programming Week 3
Introducing Applets

Orange Coast College
Computer Science 170
Spring 2004

The Plan For Week 3

- Object-Oriented Vocabulary: how OOP works
- Applet basics: the structure of a Java program
- Method basics: how methods are defined
- Talking to objects: how methods are invoked or called
- JavaDocs: how to locate and read the documentation
- Homework 3 & Quiz 3

Object-Oriented Programming (OOP)

- OOP History
  - Simula and early simulation
- OOP Fundamentals
  - Objects: “Variables” that contain both data, and the functions that operate on that data
  - Classes: “Blueprints” used to describe objects.
  - Encapsulation: Hiding data inside an object
  - Inheritance: Creating new classes from old
  - Polymorphism: Different objects, same message
Structured Programming
- Effective for linear, assembly-line type problems
- Not as helpful for building "reactive" software

Different methods of organization
- Procedural: concentrates on actions
- Object-oriented: cooperating objects
- Consider your Web browser
- Let's take a look at some OO vocabulary terms

Objects

What are objects?
- Components used to build OO computer programs
  - User-defined variables used in OO programming
  - Represent visible objects, concepts, relationships
- All objects have three properties
  - Identity: who the object is
  - State: the object's characteristics
  - Behavior: what the object can do
- Let's look at each of these properties

Object Properties

Identity: the object's "name"
- Your program can have several Button objects
- Refer to each by using it's name
- One object can have more than one name

State: values stored in an object's data fields
- One Button object may say "OK", another "Cancel"
- One Gauge object may say "Full", another "Empty"

Behavior: things the object can do [methods]
- A Window may be scrolled
- A spreadsheet cell may "know" how to recalculate
What Are Classes?
- Patterns or blueprints that describe the common characteristics of one or more objects

Patterns:
- SmallCar
- Parts:
  - 4 Wheels
  - 1 Body
  - 1 Small Engine
  - 4 Seats
  - A Small Car Can
    - Go Slow
    - Stop
    - Turn

Classes
- Objects

A Class Definition
- A class definition consists of both
  - Attributes
    - Object Data
  - Methods
    - Object Actions

Methods and Messages
- When you define a class, you specify
  - The data attributes that hold an object’s state
  - The methods that define object behavior

- When you create an object
  - You create an instance of a class
  - These instances are called objects
  - This is also called instantiation

- You send messages to objects
  - Each message invokes a particular method
Creating a new class with existing class as a starting point

- Newly created class is called a subclass
  - Think of it as a “child” class
- The “parent” class is called the superclass
- Subclass usually has new attributes or capabilities
  - “Inherits” attributes and capabilities of its parent

Exercise 3A

What is Inheritance?

Applet Basics

- Let’s start by examining the structure of a Java program
  - We’ll begin by looking at Hello.java from page 15
  - We’ll look at the basic structure or program organization
- The class header which introduces the class
  - Name of the class and keywords versus identifiers
  - Naming conventions and naming rules
  - What extends Applet means
  - The import statement

Exercise 3B - Applet Basics

Basic Program Organization

- Every Java program is a class definition
  - Class definition is a blueprint to construct objects
  - The basic class structure looks like this:

```java
... some stuff
public class SomeClassName
{
  ... some more stuff
}
```

- Must be stored in SomeClassName.java
- Only one public class per file
Class definitions have two parts: header & body

- Header for class Hello looks like this

```java
public class Hello extends Applet
```

- Starts with **declaration**: public class
  - Announces to the compiler what is to follow
- Both of these are **keywords** or **reserved words**.
  - "Built-in" vocabulary (about 50 words)
  - Always in lowercase, cannot be used as identifiers

The Class Header

The name of this class is: Hello
- This is an **identifier**, not a keyword
  - An identifier is a name that you make up
  - You'll create names for classes, methods, fields, etc.
- Java language **syntax rules** for identifiers
  - Case sensitive
  - Contains letters, digits, $, and underscore
  - Avoid using the $, Java uses it internally
  - Cannot start with a digit
  - The character set is Unicode, not ASCII

The Name of the Class

Other Naming Rules

- Common-sense rules for identifier names
  - Try to make your names understandable
  - Should accurately describe what the identifier does

```java
quarterlySales = quarterlySales + monthlySales;
x = x + x2;
```

- Conventions: agreed upon naming standards
  - Class names: **CapitalizeEveryWord**
  - Method and field names: **startLowThenCaps**
  - Constants: **CAPS_WITH_UNDERSCORES**
What Does "extends Applet" Mean?
- Says that the class Hello
- Adds something to, or builds upon the Applet class
- Means that Hello is a subclass of the superclass Applet
- Where Does Applet Come From?
  - Not a keyword
  - Isn't all lowercase
  - One of the supplied Java API classes
  - Full name is java.applet.Applet

Packages and import
- A "fully-qualified" name. Why does it contain periods?
  - java.applet is package where class is stored
  - Why not use the full name (java.applet.Applet)?
- Instead, normally use the import statement

import java.applet.*;
import java.awt.*;

- Allows access to all classes in package
- Does not allow access to packages in package
- Complete Exercise 3B

Method Basics
- Class body is delimited by braces { } (begin...end)
- Inside the class body, you'll define two things:
  - Attributes -- the data fields used to store object state
  - Methods -- where the actions take place in an object
- We'll cover attributes next week; methods this week
- Methods are used in two places in your programs
  - When you define a method, specify instructions to perform
  - When you use, invoke or call a method
What is a Method?

- A method is some code that performs an action.
- Methods are usually attached to a particular object.
- You “tell” the object to perform the action like this:
  
  ```java
  myButton.setFont(bigFont);
  ```

- This is often called “sending a message” to the object.
- The message is `setFont()`.
- Message is sent to the Button object named `myButton`.
- The Font named `bigFont` is passed as an argument.
- Also called invoking or calling a method.

Kinds of Methods

- There are two basic kinds of methods.
- Some methods perform a calculation and return a value.
  - These are called functions in some languages.
- Some methods just carry out an action.
  - These are called procedures in some languages.
- A method may require extra information to do its job.
  - You specify type of information when you define the method.
  - You supply values (arguments) when calling the method.

Defining Methods

- An applet’s methods are where actions occur.
- Hello is a “child” of Applet, so it inherits several different methods from its parent, and even one from its grandparent.
- These five methods are automatically called:
  - `init()`, `start()`, `stop()`, and `destroy()`, from Applet.
  - The `paint()` method, inherited from Component.
- The inherited versions don’t do anything interesting.
- We can override or redefine these methods as desired.
- The Hello applet overrides the `paint()` method.
Defining a Method

- Methods are defined inside a class
- Consist of a header and body, just like a class

```java
import java.applet.*;
import java.awt.*;

public class Hello extends Applet {
    public void paint(Graphics g) {
        g.drawString("Hi!", 20, 10);
    }
}
```

The Method Header

- The method header contains these four pieces
  - `public`: the access specifier determines who can invoke this method. If it is `public`, anyone can call it. If it is `private`, then it can only be called from other methods in this class.
  - `void`: the method return type. Methods that produce no value, like `paint()`, put `void` here.
  - `paint`: the name of the method, used to invoke it.
  - `(Graphics g)`: the argument list, consisting of a set of parentheses and a formal argument used to customize the behavior of the method. Each argument has a type, `Graphics`, and a name `g`.

The Method Body

- Delimited by braces `{ }` (begin...end) like a class
- Inside the body you'll put Java statements
  - Think of a statement as a “sentence” in a natural language
  - Every statement ends with a semicolon `;`
- The `Hello` applet `paint()` method contains one statement:
  - Draws “Hi” on the screen using the `drawString()` method
  - 20 pixels in from the left and ten pixels down from the top
### Style & Punctuation
- **Style and Punctuation Issues**
  - Java is a freeform language; use space to add readability
  - Try to group related statements into "paragraphs"
  - Span lines if statement is large
  - Don't normally put multiple statements on one line
  - Indent the statements inside class and method bodies
- **See the CS 170 Programming Style Guide**

### Comments
- **Comments help human readers understand your code**
- Comments are ignored by the compiler
- Java has three kinds of comments
  - Single line comments, start with `//` and end at the end of a line
  - Multi-line comments, delimited by `/*` and `*/`
  - JavaDoc comments, started with `/**`, ends with `*/`
  - Use special tags like `@author` and `@version`
  - See the style guide for more info on required tags
- **Complete Exercise 3C**

### Talking to Objects
- **In this portion you'll learn how to "talk" to objects**
- We'll start by learning how to "send messages"
  - Meet the who, what, and "wit-what" of calling a method
  - How do you call the `drawString()` method
- Then, we'll spend some time with the `Graphics` class
  - The coordinate system used by `Graphics`
  - The painting process, (who does what?)
  - The responsibilities of the AWT and the programming
- We'll also look at the `Graphics` environment
How to Send a Message I

- The paint() method is passed a Graphics object named g
- Communicate with objects by sending messages
- Let's send a message to g, asking it to "say Hi!"
- 1. Always start by saying "who" should get the message

```java
public void paint(Graphics g) {
    g;
}
```

This is called the "receiver"

How to Send a Message II

- 2. Next, tell the object what to do

```java
public void paint(Graphics g) {
    g.drawString;
}
```

This is called the "request"

Separate receiver and request with a "dot"

How to Send a Message III

- 3. Add parentheses, and pass any arguments

```java
public void paint(Graphics g) {
    g.drawString("Hi Mom", 10, 15);
}
```

Add parentheses and arguments

- What arguments? Read the JavaDocs! (Coming up!)
- Arguments, receiver, must match documentation
The `drawString()` Method

- The `drawString()` method takes three arguments:
  - A `String` containing the message to display
  - A `String` is "text between double-quotes"
  - Two `int` containing the horizontal and vertical position
    - An `int` is just a plain whole number
  - Coordinates start in upper-left position (0,0) called `origin`
  - `x` increases to right
  - `y` increases down

Message Recap

- Receiver: who the message goes to
- Request: what you want the object to do
- Arguments: additional information as required

```
someObject.doSomething( someInfo );
```

Painting and the Graphics Class

- What is the purpose of the `Graphics` class?
  - Single-tasking O/S had complete control of screen
  - Multi-tasking applications must now share display
  - AWT and Graphics class act as a "traffic cop"
- Three responsibilities of the AWT
  - Provides a "clipping region" when you paint
    - Keeps your stuff from "spilling over"
  - Monitors state of your drawing surface
  - Schedules painting operations behind the scenes
The Graphics Class

- The "default" Graphics "environment"
  - Font, used when you call drawString()
  - 2. Painting color
  - 3. Painting mode
  - 4. Coordinate system
- Some values stored as part of Component
  - Color, font are properties of component
  - Can change either permanent or temporary copy
    ```java
g.setColor(Color.red);
myButton.setForeground(Color.red);
```
- Complete exercise 3D

Reading the JavaDocs

- Open JCreator, press Ctrl+F1 or click JDK Help button
  - Locate package, then class
- Type name of class, press Ctrl+F1
- Reading the docs
  - The class summary and the inheritance hierarchy
- How to locate and read the method documentation
  - The return type and arguments
    - int, String, double

Week 3 Wrap Up

- Homework 3: A Dash of Nash
  - Due by Tuesday, February 24
  - Pick an Ogden Nash Poem
    - At least 4 lines, no more than six lines
    - Use an applet size of 300 wide and 200 high
    - Center each line in the poem
    - You'll have to use trial and error to position your text
    - Put your name in the lower-right corner of the applet
- Quiz 3: Deadline is Tuesday, February 17