Lecture 9 – Swing Interfaces

08-671

Java Programming for App Developers

September 29, 2015

Jeffrey L. Eppinger & Terry Lee

08-671 Lecture Topics

(subject to change – but only a little bit)

#1 Intro	#8 File & Network I/O
#2 Primitive Types	#9 Swing Interfaces
#3 Java Classes	#10 Swing Actions
#4 Reference Types	#11 Threads
#5 Loops & Arrays	#12 Exceptions
#6 Methods & Classes	#13 Functional Programming
#7 Lists & Maps	#14 In-class Written Exam

* Final Exam – this will be a 3-hour programming problem

Homework Plan

- Homework #6
  - Released today
  - Due on Monday (10/5)
- Homework #7
  - Released next Tuesday (10/6)
  - Due following Monday (10/12)

Exam Plan

There are two exams in the course

- Written Exam
  - In-class on Oct 15th or Oct 16th (Thursday or Friday)
  - We need a larger room (or rooms)
  - We will confirm date and location(s)
  - Plan: multiple choice & fill-in the blank
- Programming Exam
  - October 22nd from 6pm to 9pm
  - DH 2210 & DH 2315
  - Plan: same as HW#6, but different

Outline

- ✓ Course Plan
- ⇢ Questions
    - Packages => Dates
- Swing Interfaces
- Homework
- Questions

Question for You

- What was the Y2K problem?

Question for You

- How do you get the time in Java?

Java Time

- Java represents time in milliseconds since January 1, 1970 12:00am GMT
- Original Unix represented time as number of seconds since 1/1/70 00:00 GMT
  – Stored in an int or unsigned int
    - Int-based apps will have problems starting in 2037
    - Unsigned int apps will have problems sometime after 2100
- Java uses a long
  – Java-based apps can represent dates ±292 million years

Example

```
public class Now {
  public static void main(String[] args) {
        long now = System.currentTimeMillis();
        System.out.println(now);
    }
}
```

Question for You

- How would you format this in a useful way?

Example

```
public class Now2 {
  public static void main(String[] args) {
      long now = System.currentTimeMillis();
      java.text.DecimalFormat df;
      df = new java.text.DecimalFormat("#,###");
      System.out.println(df.format(now));
  }
}
```

Example

```
public class Now3 {
  public static void main(String[] args) {
      java.util.Date d = new java.util.Date();
      System.out.println(d);
  }
}
```

Example

```
import java.util.Calendar;

public class Now4 {
   public static void main(String[] args) {
       Calendar c = Calendar.getInstance();
       int year   = c.get(Calendar.YEAR);
       int month  = c.get(Calendar.MONTH);
       int day    = c.get(Calendar.DAY_OF_MONTH);
       int dow    = c.get(Calendar.DAY_OF_WEEK);
       System.out.println("year = "+year);
       System.out.println("mon  = "+month);
       System.out.println("day  = "+day);
       System.out.println("dow  = "+dow);
   }
}
```

Example

```
import java.text.SimpleDateFormat;
import java.util.Date;

public class Now5 {
  public static void main(String[] args) {
      SimpleDateFormat sdf =
            new SimpleDateFormat("dd-MMM-yy @ HH:mm");
      Date d = new Date();
      System.out.println(sdf.format(d));
  }
}
```

Example

```
import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

public class Now6 {
   public static void main(String[] args) {
        String languageCode = args[0];
        Locale locale = new Locale(languageCode);
        DateFormat df = DateFormat.getDateTimeInstance(
                                      DateFormat.MEDIUM,
                                      DateFormat.MEDIUM,
                                      locale);
        Date d = new Date();
        System.out.println(df.format(d));
   }
}
```

Example

```
import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

public class Now7 {
   public static void main(String[] args) {
        String languageCode = args[0];
        Locale locale = new Locale(languageCode);
        DateFormat df = DateFormat.getDateTimeInstance(
                                    DateFormat.LONG,
                                    DateFormat.LONG,
                                    locale);
        Date d = new Date();
        System.out.println(df.format(d));
   }
}
```

Example

```
import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

public class Now8 {
   public static void main(String[] args) {
        String languageCode = args[0];
        String countryCode  = args[1];
        Locale locale = new Locale(languageCode, countryCode);
        DateFormat df = DateFormat.getDateTimeInstance(
                                    DateFormat.LONG,
                                    DateFormat.LONG,
                                    locale);
        Date d = new Date();
        System.out.println(df.format(d));
   }
}
```

Another Question for You

- When you use a class in Java, how does Java know:
  - Whether you wrote it?
  - Whether it's in the class library?
  - What methods, etc, the class offers?

CLASSPATH & Import Statements

- Java maps the class name into a file name
- Java searches for this file down the CLASSPATH
  - If not specified, checks current dir & location of the JVM
- Java Archives (JAR files) can also be searched
- Java code is collected into packages
  - If you don't want to type the package name over and over again, use an import statement
    - E.g., use java.util.Calendar **or**
      import java.util.Calendar **and**
      just use Calendar in the rest of the file

Last Question For You

- What is cookbook programming?
- Cookbook programming
  – (The book calls this using a Framework)
  – Let’s you follow a recipe for writing your programs
  – All cakes are different, but there are a few basic recipes and everything else is a slight variation
    - Add some cinnamon
    - Substitute chocolate chips instead of nuts

Cookbook Programming

- You have a template for your program
- You change things around, but you don't mess with the overall structure
- Examples:

```
public static void main(String[] args) { … }
for (int i=0; i<args.length; i++) { … }
```

- Many people consider Swing development to be cookbook programming

A Little History

In the beginning…

- There was Java, it was like C++, it was good
- Then came HotJava, it was a Java-based browser
  - You could run hunks of Java code called Applets
  - It was cool → Netscape & then IE added Java support
- But Applets were a pain
  - Browsers had out of date JVMs
  - Used the AWT (lots of platform-based non-Java code)
  - Didn't have the look and feel of the rest of the platform
  - Couldn't run as a standalone program with a GUI

Swing

- A new user interface environment
  - Implemented in Java
    - More consistent across implementations
  - Offers different “look and feel” options
    - Windows, Unix, and other (Metal)
  - Can be a main method or a JApplet
- Still uses AWT for event handling, fonts, etc.
  - BTW – Swing portability is still the subject of complaint
  - SWT – The standard widget toolkit (from Eclipse) is supposed to address this

Simplest Structure

- You make a Window
- Make a container
- Add your Buttons, Boxes, etc to the container
- Set up the window to display the container

Components

Swing has lots of components:

- JLabel
- JButton
- JCheckBox
- JChoice
- JRadioButton
- JTextField
- JTextArea
- JList
- JScrollBar
- … and more

JFrame & JPanel

- JFrame is the Swing Window
- JPanel (aka a pane) is the container to which you add your components (or other containers)

Layout Managers

- The default Layout Manager is FlowLayout
  - Place items in the container from left to right
  - When a line is full, FlowLayout goes to the next

More Layout Options

- GridLayout
- GridBagLayout
- BorderLayout
- Explicit Placement

Example

- QuoteGUI.java
- Let’s do it in Eclipse

Several Different Recipes

- QuoteGUI0.java
  - Builds GUI in main method
  - Not recommended, it’s hard to implement actions
- QuoteGUI1.java
  - The Head First Java recipe
  - Builds GUI in constructor of new class
  - **I prefer this way**
- QuoteGUI2.java
  - The Java in 21 Days recipe
  - Builds GUI in constructor of JFrame subclass
  - Does demonstrate inheritance
  - But is more complicated and causes UID warning in Eclipse

Next Lecture

- Making the Swing GUI respond to actions
- More on laying out your GUI

Homework #6

- Will be released after class
- Due on Monday
- You’ll need to create a Swing GUI to drive your Directory from HW#5
- You can start to layout the GUI
- We’ll do actions on Thursday
- In recitation, we can discuss more techniques to improve layout