Lecture 10 – Swing Actions

08-671
Java for Application Programmers

February 11, 2016

Terry Lee
Assistant Teaching Professor
School of Computer Science

08-671 Lecture Topics

(subject to change – but only a little bit)

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

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

Exam Plan

- Written Exam
  - In-class on Feb 25th (Thursday)
  - Location: BH A51 (Giant Eagle Auditorium)
  - Plan: multiple choice & fill-in the blank, etc.
    - Closed everything. Pencils and erasers
- Programming Exam
  - Date and Time: 5:30pm on Mar 1st (Tuesday)
  - Location: BH A51 (Giant Eagle Auditorium)
  - Plan: same as HW#6, but different
    - Need your laptop. Don't forget your power adapter

Outline

✓ Administrative Issues

⇢ Questions

Swing Actions

Nested & Anonymous Classes

Enumerations

Sample Final Exam Questions

What are SwingConstants?

- Interface (`javax.swing.SwingConstants`) where you can find:
  - a collection of constants generally used for positioning and orienting components on the screen.
  - Notice that they are all ints.

Details about constant field values:

https://docs.oracle.com/javase/8/docs/api/constant-values.html

Also, remember `java.util.Calendar` class in Lecture 9?

Check out constant fields for `java.awt.Font` too

Outline

- ✓ Administrative Issues
- ✓ Questions
- ⇢ Swing Actions
- Nested & Anonymous Classes
- Enumerations
- Sample Final Exam Questions

Events in Swing

- An event is when something changes
  - Button clicked, scrolling, mouse movement, keys typed, etc.
- Swing (actually AWT) generates an event
- To do something, you need to implement a Listener Interface and register interest with a component

Event Listeners

Swing has lots of event listeners interfaces:

- ActionListener
- AdjustmentListener
- FocusListener
- ItemListener
- KeyListener
- MouseListener
- TreeExpansionListener
- TextListener
- WindowListener
- …and on and on…

`ActionListener` Interface

- Events for JButtons, JTextFields, etc
  - The things we are using
- Implement `ActionListener` Interface
  - Provide actionPerformed method
- In `actionPerformed()` method
  - Can use event.getSource() or event.getActionCommand() to determine which button was clicked, etc.
- Register the ActionListener
  - Step to connect ActionListener object with GUI component

Example

- Let’s write QuoteGUIAction

`this` keyword

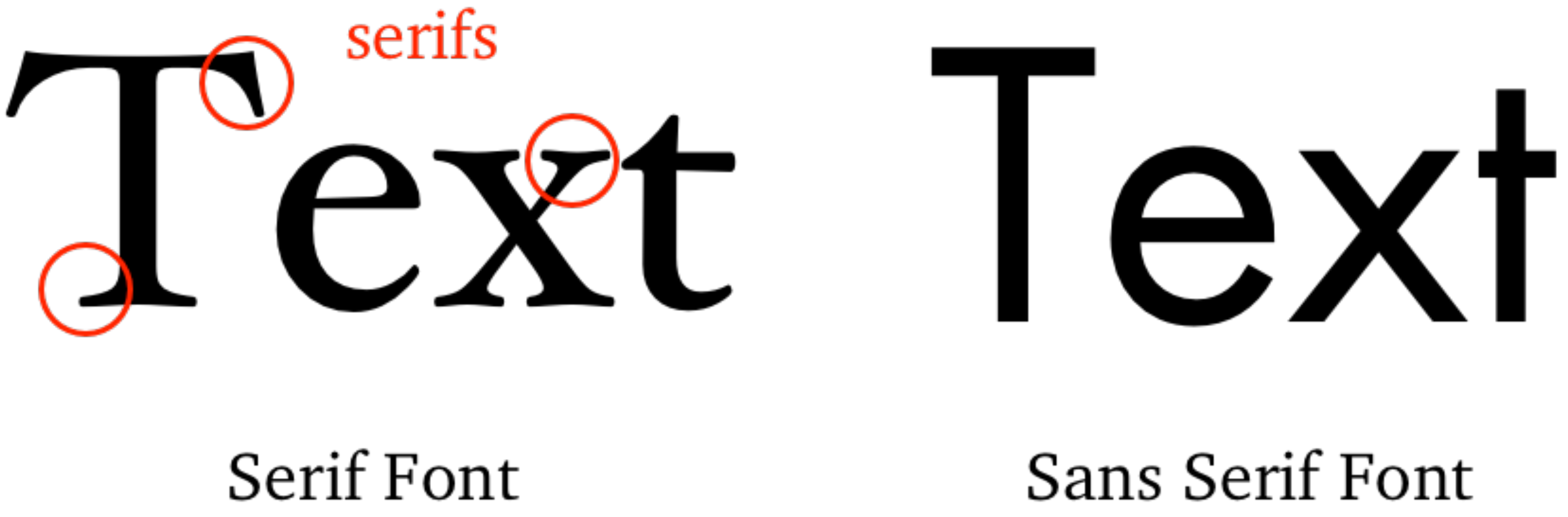
- A **reference to the current object**
- Because a field is shadowed by a method or constructor parameter

```
public class Point {
    private int x = 0;
    public Point(int x) { this.x = x; }
}
```

- To call another constructor in the same class

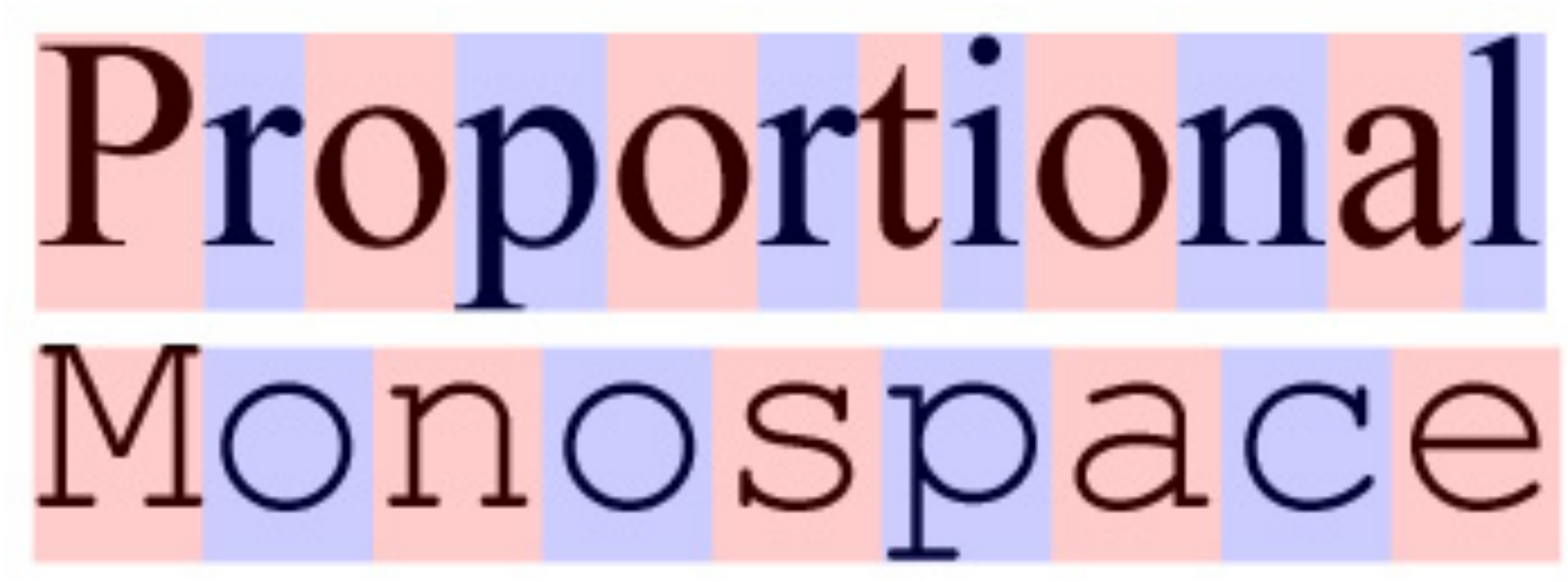
```
public Rectangle() {
    this(1, 1);
}
public Rectangle(int width, int height) {
    …
}
```

Serif vs. Sans Serif



Source: http://drmarkwomack.com/a-writing-handbook/style/typography/

Proportional vs. Monospaced



Source: https://en.wikipedia.org/wiki/Typeface

Organizational Tips

- Declare references to components you'll be manipulating as instance variables

```
private JButton obamaButton;
private JButton trumpButton;
```

- Put the code that performs the actions in private "helper" methods. (Keeps things neat) or use other recipes

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == obamaButton) {
        doSomething();
    }
    if (e.getSource() == trumpButton) {
        doSomethingElse();
    }
}
```

Three Different Recipes (from Lecture 9)

- QuoteGUI1.java
  - Builds GUI in main method
  - Not recommended, it's hard to implement actions
- QuoteGUI2.java
  - The Head First Java recipe
  - Builds GUI in constructor of new class
- QuoteGUI3.java
  - The Teach Yourself Java in 21 Days recipe
  - Used in ToDoSwingGUI.java example
  - Builds GUI in constructor of JFrame subclass
  - Does demonstrate inheritance

Updated Different Swing Recipes (Including ActionListener options)

1) Don't subclass anything
   - a) Your GUI class implements ActionListener
   - b) private nested class implements ActionListener
   - c) private static nested class implements ActionListener
   - d) Anonymous class implements ActionListener

2) Subclass JFrame (`extends JFrame`)
   - a) Your GUI class implements ActionListener
   - b) private nested class implements ActionListener
   - c) private static nested class implements ActionListener
   - d) Anonymous class implements ActionListener

Teach Yourself Java in 21 Days uses recipe 2a

Head First Java uses recipe 1b

Nested Classes

- You can declare a class inside a class

```
public class Outer {
    private class Inner {
        …
    }
}
```

  - Usually, these are private classes
  - Non-static nested classes (also called inner classes)
    - have access to enclosing class's methods & variables
    - but must be instantiated in a non-static context
  - Static nested classes
    - will NOT prevent instances of enclosing class from being garbage collected

Anonymous Classes

- Un-named class that implements an interface or extends another class
  - Used when a class is so simple and to be instantiated only once in your code
  - No need to use `implements` or `extends` keywords
- Code for the class is provided inline

* Will discuss this further in lecture 13

Updated Different Swing Recipes (Including ActionListener options)

1) Don't subclass anything
    - a) Your GUI class implements ActionListener
    - b) private class implements ActionListener
    - c) private static class implements ActionListener
    - d) Anonymous class implements ActionListener

2) Subclass JFrame (`extends JFrame`)
    - a) Your GUI class implements ActionListener
    - b) private class implements ActionListener
    - c) private static class implements ActionListener
    - d) Anonymous class implements ActionListener

Teach Yourself Java in 21 Days uses recipe 2a

Head First Java uses recipe 1b

For QuoteGUIAction.java, I would use 1d

Outline

- ✓ Administrative Issues
- ✓ Questions
- ✓ Swing Actions
- ✓ Nested & Anonymous Classes
- ⇢ Enumerations
- Sample Final Exam Questions

Revisit: What are SwingConstants?

- Interface (javax.swing.SwingConstants) where you can find:
  - a collection of constants generally used for positioning and orienting components on the screen.
  - Notice that they are all ints.

Details about constant field values:

https://docs.oracle.com/javase/8/docs/api/constant-values.html

Check out constant fields for java.awt.Font too

Swing & Other Constants

- A list of constants that are used as parameters to various Swing methods and others:
  - E.g., `javax.swing.SwingConstants`, `java.awt.Font`
  - Also saw many constants in `java.util.Calendar`
- Notice these constants are ints and Strings (mostly)
  - You can easily use incorrect values from the wrong list of constants

Enumerations

- Define class that is a list of all possible values
- Values are constants
- Leverage Java Class mechanism to ensure type checking at **compile time**

Enum Examples

```java
public enum Month {
    JANUARY,
    FEBRUARY,
    MARCH,
    APRIL,
    MAY,
    JUNE,
    JULY,
    AUGUST,
    SEPTEMBER,
    OCTOBER,
    NOVEMBER,
    DECEMBER
}
```

```java
public enum DayOfWeek {
    SUNDAY, MONDAY, TUESDAY,
    WEDNESDAY, THURSDAY,
    FRIDAY, SATURDAY;

    @Override
    public String toString() {
        switch (this) {
        case SUNDAY:
        case SATURDAY:
            return "Weekend :-)";
        default:
            return "Weekday :-(";
        }
    }
}
```

Example to Run

- `EnumTest.java`
  - `Month.java`
  - `DayOfWeek.java`

Enumerations are Often Nested Classes

- `EnumTest2.java`

Check Out `java.time` package

- New in Java 8
  - Immutable Classes
  - Separate classes for Date and Time, etc.

Outline

✓ Administrative Issues

✓ Questions

✓ Swing Actions

✓ Nested & Anonymous Classes

✓ Enumerations

⇢ Sample Final Exam Questions

Sample Final Exam Questions

In Java:

- What is an abstract class?
- What is an interface?
- Why do we need both interfaces and abstract classes?
- What are the advantages of using enums rather than constant ints & Strings?

Next Week

- Threads
- Exceptions
- More Swing Practice
- Last Homework!

Read Head First Java Chapters 11 & 15