Today

- Lambda Functions, Cont.
- Method References.
- Event-Driven Programming.
- JavaFX Installation and Setup.

Today, Cont.

- Second technique: TestFilter2.java – use an anonymous class instead.
  - This is the best version, so far!

Lambda Functions, An Example

- Suppose you want to have a method that only displays certain members of a collection, depending on a criteria that is specified outside the method and provided as an argument.
- You don’t want to hard code the criteria inside the display method.
- First technique: Supply an object implementing an interface that has a “filter” method that returns a true or false. See TestFilter1.java

Lambda Functions, An Example, Cont.

- Second technique: TestFilter2.java – use an anonymous class instead.
  - This is the best version, so far!

Lambda Functions, Cont.

- Filter is an example of a Functional Interface.
- These interfaces can only contain a single abstract method.
- Lambdas can only be created using Functional Interfaces.
- You can use the @Functionall Interface annotation to make sure your interface is OK.

Pre-Defined Functional Interfaces

- Turns out the java.util.function package has many pre-defined generic functional interfaces.
  - See the API, svp.
  - The one that matches our check function signature is called Predicate<T>. It has the abstract method signature:
    ```java
    boolean test(T)
    ```
  - See TestFilter4.java.
  - Even better than the last best one!
Method References

- ArrayList.sort() accepts a Comparator<T> object that can specify the desired algorithm for comparison of objects of type T.
- Turns out Comparator is a Functional Interface, so you can build a lambda function for a comparator.
- See TestSorting.java.

Method References, Cont.

- But suppose our Person class has a method that matches the Comparator interface:
  ```java
  public static int compareByAge(Person p1, Person p2) {
    return p1.age - p2.age;
  }
  ```
- In this case you can supply a Method Reference instead of building the lambda function.
- See TestSortingAgain.java.

Lambda Functions, Cont.

- You can have multiple lines of code in a lambda, but don’t make them too long.
- You can use as many lambdas for arguments as you wish when invoking a function, as long as they each match up to some functional interface.
- In GUI programming the most common event handler interface, EventHandler<ActionEvent> is a functional interface, so lambdas can be used to attach event code to handlers.

Event-Driven Programming

- So far, for assignments, you have written single threaded “batch” programs – the coder (you!) controls the flow of execution.
- For event-driven programs, the user controls the flow by initiating events.
- A GUI interface consists of components contained within a frame (or “window”). Components and even the frame itself can respond to events.
Events, Cont.

- Most events you ignore – your interface does not have to respond to every possible keyboard and mouse initiated event – that would be crazy!!

- To respond to an event in code, you attach an Event Handler object to a component. When an event occurs to a component that has the focus, the handler receives an object which contains information about the source of the event (which mouse button, etc.)

- You decide which events are of interest and what you want your program to do when these events occur.

GUI Construction

- Construction of a Graphical User Interface involves:
  - Creating the design of the window – choosing components (buttons, labels, text boxes, etc.) and where they are positioned in the window.
  - Changing the properties, including appearance properties, of the components.
  - Adding and linking functionality to events, where needed.
  - Repeating these steps for each window!
  - Connecting the program flow between windows.

History of GUI Components in Java

- Next slide has a hierarchy of Historical Interest only:

- These classes are part of the Component hierarchy in javax.swing

- Swing was a modern improvement of the older AWT (“Abstract Windows Toolkit”) classes.

- The following diagram is a simplified summary of the structure:

JavaFX

- JavaFX classes are no longer part of this hierarchy.
- JavaFX is a bold, relatively new world of GUI coding!

- Swing API code will no longer be updated. Any improvements will be in JavaFX packages only.

- Library code was bundled with the API but recently was changed back to being a separate download (See the Resources page.)

- And you need to configure Eclipse to program in JavaFX.
JavaFX, Cont.

- Swing was developed mostly for enterprise/business use, not for personal use and certainly not for mobile devices.
- JavaFX is available in the API since Java 7.
- Jazzier controls and more support for portable devices.

“Client Technologies”

- For lots of links on JavaFX and Swing see:
  http://docs.oracle.com/javase/8/javase-clienttechnologies.htm

Getting Started Using JavaFX - Topics

- Overview / Installation.
- Start by looking at the Scene/Stage model.
- What is a Scene Graph?
- How components are laid out using Pane-based layout managers.
- Look at the use of CSS style sheets.
- Attach events to components.
- Look at the use of FXML files and SceneBuilder.

JavaFX Overview

- You can separate style and layout from functional code using *.css and *.fxml files. Very modern!
- Contains a rich set of tools for:
  - Web browsing.
  - Media playing.
  - 2D and 3D drawing and animation. Including many visual effects.
  - Taking advantage of hardware accelerated rendering.

JavaFX Overview, Cont.

- Already contains separate threads for:
  - The main application.
  - Rendering.
  - Media playback.
- Over 50 controls, built-in.
- A dozen built-in chart types.
- 2D and 3D transformations as well as many other visual effects can be applied to most controls.

JavaFX - Installation

- You need to add e(fx)clipse to eclipse in order to have the libraries and tools you need to write JavaFX code.
- You don't need SceneBuilder but might like to play with it later.
- See the Resources page on the course web site.
- The homepage for e(fx)clipse is:
  http://www.eclipse.org/efxclipse
**e(f)xclipse Version**

- Currently 3.5.0… (Even though the Eclipse Marketplace says it is version 3.6.0.)

- Follow the instructions on the course “Resources” page.

**SceneBuilder Version**

- Latest installer build linked to the Gluon site is 11.

- A drag and drop visual editor that edits an *.fxml file in the background.

- SceneBuilder can be invoked from within eclipse, but essentially runs outside of eclipse.

- More later…

**New JavaFX Project Wizard**

- Sets up quite an elaborate structure for you!

- Don’t have more than one JavaFX project per eclipse project.

- Since the JavaFX API is no longer bundled with the rest of the API, you need to add the libraries to your project. And you need to add to the “Run” VM command line. More details in the Resources page. The current JavaFX library version is 14, the Resources page refers to version 13.

- Let’s have a look at the Wizard and how to configure the project.