Java Swing Startup

- Java's Swing framework is the platform's "official" toolkit for creating graphical user interfaces (GUIs)

- It's not the only one — there's AWT and SWT

- Follows the JavaBeans and Event specifications: this means that parts of the API have a common "feel"

- Just like Java, aims to be "write once, run anywhere," but with some caveats because it involves GUIs

Key Swing Concepts

- Prerequisite: you must be comfortable with many Java language features, particularly with interfaces, abstract classes, semantics of public/protected/private, packages, and, later on — believe it or not — threads and memory management

-

- Pluggable look-and-feel — in Java 1.5, it is actually a "skinnable" look-and-feel

- Self-sizing components

- Extensible via the usual object-oriented mechanisms

- Generally follows model-view-controller (MVC), but frequently combines VC within a single class
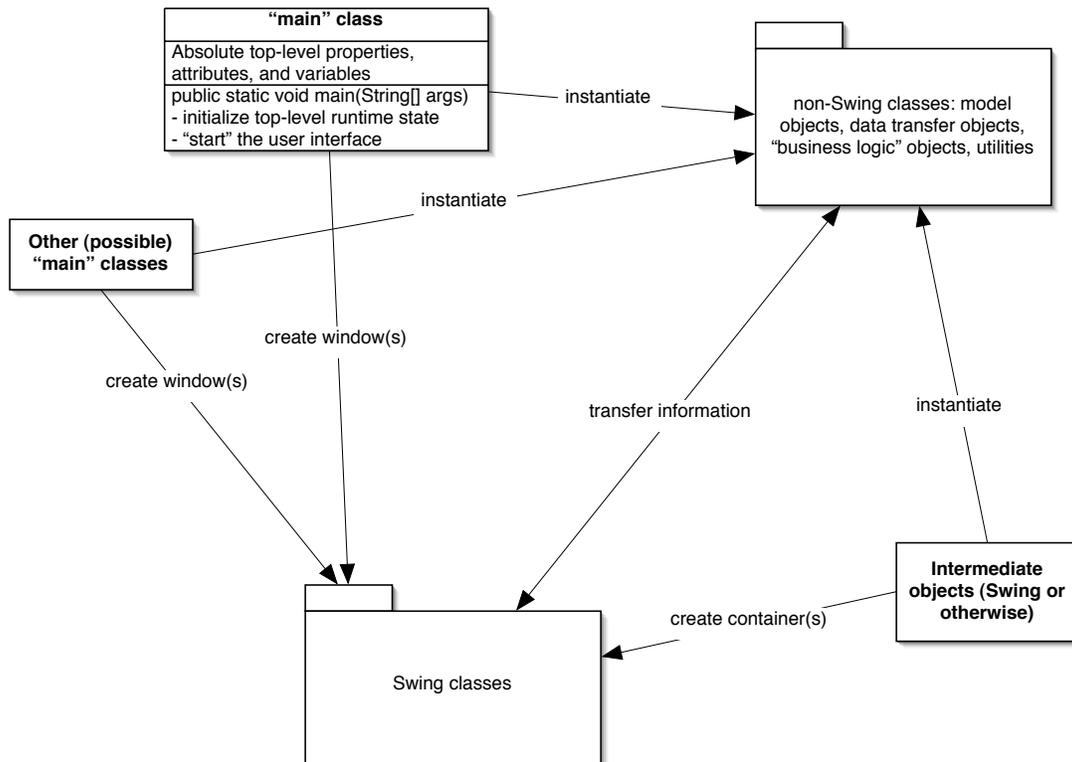
Learning Swing

- Lots of stuff on the Web — if you know how to assimilate these, you don't need a book

- The official Swing tutorial:

    http://java.sun.com/docs/books/tutorial/uiswing/index.html

- Java API reference:

    http://java.sun.com/j2se/1.4.2/docs/api/index.html

    http://java.sun.com/j2se/1.5.0/docs/api/index.html

- The Swing Connection

    http://java.sun.com/products/jfc/tsc/index.html

- Tons of non-Sun Swing resources

Our Approach in Class

- Focus on "best practices," either from my own experience or others — how we do certain things, and why we do it that way

- "Reusable until proven proprietary" — generally structure our code with an assumption that we may use it again in other programs or contexts, until we explicitly decide that we really *are* writing something that is truly application-specific

- We'll try to avoid going over information that is otherwise well-specified on the Web

Anatomy of a Swing Application ("My" Way)

- No official "template" for using Swing

- Two-edged sword — allows for great flexibility but sometimes unclear on what is the "best" or "right" way to do things

- Stuff I'll give you reflects much of my own philosophy based on specific lessons and experiences I've learned with Swing

"main" class

Absolute top-level properties, attributes, and variables
public static void main(String[] args)
- initialize top-level runtime state
- "start" the user interface

Other (possible) "main" classes

non-Swing classes: model objects, data transfer objects, "business logic" objects, utilities

instantiate

instantiate

create window(s)

create window(s)

transfer information

instantiate

Intermediate objects (Swing or otherwise)

create container(s)

Swing classes

The Blank Slate Composite: JPanel

- JPanels compose other Swing components into self-contained, reusable units

- Use JPanels liberally — they help you to structure your user interface

- Two primary phases of JPanel construction:

  1. construct persistent components (e.g. components that you'll want to "get to" later on)

  2. arrange these components within the JPanel

JPanel Parts

- *layout* — the panel's layout manager; this is the overall "philosophy" by which the panel's children are arranged

  1. Create, configure, then set the layout manager

  2. Build the panel by adding child components; the layout manager and the order in which they're added determines where they ultimately end up

- *border* — the panel's border; what decorates or frames the panel?

Layout Managers

- Layout managers hold the "intelligence" or "heuristics" that determines how a panel's child components are arranged within the panel

- Mastery of layout managers, and knowing which one to use, is a key step toward Swing proficiency

- A sampling: *BorderLayout*, *FlowLayout*, *BoxLayout*, *GridLayout*, *GridBagLayout* (and if no combination of these will do, you can roll your own)

- Key information: an ordered list of components to lay out, plus an optional "constraint" for each component

- Layout managers can do their work because of Swing's triplicate notion of component size:

  - *minimum size* — the absolute smallest area that a component will occupy

  - *preferred size* — the area that a component would ideally like to have

  - *maximum size* — the absolute largest area that a component will occupy

- One can think of a layout manager's "semantics" in terms of how it uses the minimum, preferred, and maximum sizes of its components

Borders

- A border is any kind of "decoration" that surrounds a Swing component, including white space

- A Swing component's border can be assigned by calling *setBorder()* with a Border object as the argument

- As with layout managers, individual classes represent specific border types: *EmptyBorder*, *LineBorder*, *EtchedBorder*, *TitledBorder*, to name a few

- A "very special" border is *CompoundBorder* — it wraps multiple borders into a single composite border

Border Advice

- Instead of constructing borders individually, ask *BorderFactory* to make them for you — allows for *shared* border objects

- Note how a border is independent of a component's true content; thus, one can view a border as being the responsibility of the component's *container* (e.g. let the container decide spacing, titling, etc.)

- Spacing issues can be addressed using either empty borders or layout managers — use good software design to decide which is best for your program

Child Components

- Once you've set up a panel's layout manager, it's time to add the components that make up that panel

- Another large part of Swing proficiency is knowing the available components and how to use them properly

- Of course, you can put panels within panels, whether for layout purposes or software reuse

- Key decision: which of your child components should be "persistent" — that is, which ones should you "hold on to" as instance variables?

- In general, if you will need access to a component later on in the program — typically to *get* or *set* the information it is displaying — then you assign it to an instance variable

- If a component's role is purely display-oriented — e.g. static labels, scroll panes, tab views — then you can probably create it on the fly, add it to your panel, then forget about it

- Now, all that remains is getting to know the Swing components and learning to use them properly…here's a good resource for that:

http://java.sun.com/products/jfc/tsc/articles/component_gallery/index.html