



**Capstone Courseware, LLC**

33 Boylston Street  
Jamaica Plain, MA 02130

877-227-2477  
capstonecourseware.com

# Java Foundation Classes

**Roman V. Snytsar  
Will Provost**

## *Instructor's Guide*

**Revision 5.0**



### Revision Notes

**Revision 5.0** updates the course for Java 5.0. That is, all examples have been re-tested on the new SDK, and where the resulting GUIs appear substantially different, screenshots have been replaced. Not all code has been refactored to take full advantage of Java-5 language features, since this one version is meant to be appropriate for both Java-1.4 and Java-5.0 programmers.

**Revision 1.2** introduces support for Linux systems. The book is unchanged, and the pre-existing Windows installer includes fixes for a few bugs encountered during Linux testing.

**Revision 1.1** adds the third module, "Advanced GUI Design with JFC".

**Revision 1.0** is the initial revision.





## Course Overview and Philosophy

This course develops knowledge about building the GUI interfaces in Java. It is designed for students who already understand the basics of the Java programming language, and who desirably but not necessary are already familiar with AWT. Students may be coming to this course from our Java GUIs module (part of the Java Programming course), or they may just have basic experience in Java language.

The first module starts with an introductory chapter that offers some perspective on the Java Foundation Classes and the Swing library in particular. The first chapter also gives a laconic introduction to the AWT architecture. Chapters 2 and 3 present the most common Swing components as well as the Swing top-level containers. These chapters give the student enough information to build a simple GUI using the Swing library. Chapter 4 goes beyond the basics and presents a theory of Design Patterns and its applications to the Swing and AWT design.

This first module is lab-intensive, roughly as much as other Capstone courseware, although there is a bit more emphasis here on demonstration and a bit less on lab work. Instructors are encouraged to let students dwell on the labs in the later chapters, and to develop real working solutions. Many of the labs can also be expanded easily with a little in-class invention.

The second module delves into the intricacies of the JFC tree and table controls. These are of a similar structure and complexity, each requires an iterative approach, working first from the most basic practical uses and then into more complex techniques such as customizing model implementation, cell rendering or in-place editing. One chapter each is devoted to JTree and JTable, and then a third chapter goes deeper into the capabilities of the separate model implementation. This last chapter allows the students to investigate some more difficult practical problems and introduces more advanced techniques for solving them, such as lazy evaluation and the evictor pattern.

The third module expands the student's knowledge about building complex user interfaces, with whatever control types. Where more information is to be presented than might fit on a given window, splitters, scrollers, and tabbed windows can be used to organize the interface. Popup elements such as message and dialog boxes, file and color choosers, popup menus, and tooltips are studied and implemented in labs. Finally, JFC's data transfer framework is explored, and students implement clipboard operations and drag-and-drop features in lab exercises.





## Timeline

### Day 1

JFC Intro, Chapter 1	Introduction to JFC
JFC Intro, Chapter 2	Basic JFC Application Design

### Day 2

JFC Intro, Chapter 3	Swing Components
JFC Intro, Chapter 4	JFC Architectural Patterns

### Day 3

Trees and Tables, Chapter 1 Hierarchical Data and JTree  
Trees and Tables, Chapter 2 Tabular Data and JTable (spans days 3 and 4)

### Day 4

Trees and Tables, Chapter 3	Managing the Model
Advanced GUI, Chapter 1	Organizing Application Windows

### Day 5

Advanced GUI, Chapter 2	Popup GUI Elements
Advanced GUI, Chapter 3	Data Transfer

The first two chapters of the second module follow similar paths. Each lays out one of the more sophisticated JFC controls in good detail, and each has a recommended two hours of labs. Each should take roughly three quarters of a day, leaving a half-day for the final chapter. This last chapter is more adventurous theoretically, but it is assumed that students will only work the first of the three labs, so time should not be a big problem.

If there is extra time, one or both of the optional labs in chapter 3 can be quite rewarding for dedicated students, but neither of these is for the faint of heart. Leading a review of the answer code to these labs might be a good follow-up to each of the corresponding **Inventory** demos, showing similar techniques applied to different problems and JFC controls. Code review only should add no more than an hour to the class time.

The third module offers a lot of simple techniques, more in cookbook style. It can be covered in a single day if two or three labs are skipped, and this is not likely to be too harmful given the nature of the material.





Although not recommended by Capstone, a possible four-day timeline for strong students who already have AWT programming experience is as follows. Chapters 1 and 2 of the first module may be covered completely but at a faster pace. Chapters 3 and 4 should still be covered completely. In the second module, cut the third of three labs in each of Chapters 1 and 2, and skip the two optional labs in the last chapter. In the third module, skip labs 1B, 2A, and 3B. It is possible that even with these cuts three days will not be enough; next to go would be the second module, last chapter, first lab.





## The Eclipse Overlay

Some students and instructors may prefer to install an IDE and use it in working through the course exercises. Capstone Courseware provides an optional package of workspaces and project files for Eclipse 3.0.2 for this course. (See the course Setup Guide for download URLs.) Instructors, use this package on your own initiative and at your own risk. You should have experience yourself with Eclipse before using the overlay package in the classroom. The workspaces and projects have been tested lightly with the course but are not part of the standard product.

That said, this overlay should save a good deal of work for those who wish to use Eclipse instead of the text editor and command-line tools that are standard for the course. See the file `c:\Capstone\JFCIntro\Eclipse\ReadMe.html` for general notes on how to use the Eclipse overlays for Capstone courses. Be prepared to walk students through the first few exercises in Eclipse; the notes in this file are for experienced Eclipse users, and will not be clear to many students on their own.

One foible of Eclipse is that workspaces cannot be arbitrarily relocated without confusing the workbench a bit. To wit, if you choose not to install the workspaces to the default, `c:\Capstone`, you will probably find that projects either don't build when they should, or that when you try to run them the Eclipse launcher "can't find the main class." The bottom line is the projects all need to be refreshed if they are opened from a path other than the one at which they were last open. So in this case the best process is to have everyone begin by opening, refreshing, and then closing all projects; this should clean up any odd behavior due to the "surprise" location.

For this course, some specific notes:

- Many of the applications load images, and they do so by looking for image files as class resources. The primary lab image holds these files in the class path under `c:\Capstone\Classes`, so the Eclipse projects have an **Images** folder linked to that location, and their classpaths include GIF files from there. This should be transparent in practice, but students may wonder how the image loading succeeds when running from this different class path.
- A few applications in the **JFCTree** module load large data files, and they do so using a relative path – the **Inventory** and **Revolution** examples, and hence Labs 3A, 3B and 3C as well. In these exercises, the working directory is significant, in a way it usually isn't for our labs. Running from Eclipse right out of the box, these applications will fail. See the files **EclipseErrata.html** in the Eclipse projects themselves for suggested workarounds, but we recommend just building and running these from the command line, for the sake of simplicity.

Again, Capstone Courseware can only offer complete technical support on the standard course, and while we hope this overlay is convenient, it is not as thoroughly tested as the core lab image at this time. If a given exercise is giving trouble, please be certain to build



and run it from the command line, using the SDK tools as prescribed in the student guide, before contacting Capstone.

We always welcome feedback on our courseware, and especially with this new undertaking we would appreciate whatever comments and criticism you might have. Eclipse, like all IDEs, tries to promote ease of use by providing many different ways of doing the same thing; this is convenient for users but does leave a lot of questions as to what's best practice. At this time we believe consensus is still forming around a number of basic practices, and we'd like to hear how you use Eclipse for training situations and how this overlay works out for you. Please contact Will Provost at **[provost@capstonecourseware.com](mailto:provost@capstonecourseware.com)**.





## Module 1: Introduction to JFC

### Chapter 1:

This chapter sets the stage for the rest of the module. It introduces the Java Foundation Classes (JFC), a collection of APIs that allows developers to build full-featured GUI applications, and its major parts: AWT and Swing. When augmented with Appendix B, the chapter may also serve as a brief introduction to AWT, covering the Component-Container pattern, layout management, and the AWT event model.

### Chapter 2:

In this chapter students get their first detailed look at Swing. Here the students build their first simple application that has all basic JFC features. The chapter presents the architecture of the Swing top-level containers. The Root Pane - Content Pane hierarchy is presented in detail.

### Chapter 3:

This chapter is the real flesh of the module. Here we learn how to use the most common Swing components. The chapter covers buttons, menus, and text components. Assuming there is enough time, the instructor may want to dig deeper into the Swing text management framework before Lab 3A and into data transfer before Lab 3B.

### Chapter 4:

Here we add some theory. This chapter introduces some of the primary Design Patterns used in JFC. We look how the Design Patterns are applied to the Swing components design, and to application designs that use Swing. Lab 4A is also a good point to talk about the Swing components as Java Beans.





## Module 2: JFC Trees and Tables

### Chapter 1:

This chapter focuses on the JTree control. This chapter will likely be a bit of a change of pace for students as they come off of the previous module. To use JTree and JTable effectively requires deeper knowledge than that needed for the simpler controls considered so far. There may well be a bit of resistance to this depth; some students will want more of a quick start in plugging trees and tables into interfaces, or will want to progress through these two as quickly as, say, listboxes and comboboxes. There's plenty of practical material here, but we're also hoping that students will walk out of the course ready for some real-life challenges, most of which will require that they've spent some time under the hood, so to speak. Conversely, students with some design pattern knowledge may really warm up to this material.

### Chapter 2:

Now it's JTable's turn. The chapter approaches tables by much the same path as was used for trees, starting with the most basic possible uses and then delving into models, renderers, editors, selection models, etc. Thanks to their work on the previous chapter, students may work through this one a little more quickly, even though it's pretty much the same scope and depth. The labs are arguably a little more demanding, so it should even out.





## Chapter 3:

This chapter takes a step back from the tree and table controls and classes themselves. Most of the chapter focuses on managing lots of data in a GUI. This can be very interesting in itself, but it's important to assert the real theme here, which is a more in-depth study of the use of models behind JFC presentation classes. After the simple Chess example, all the problems presented will involve great quantities of data and tuning applications for performance under some stress. These are just some of the problems that good decomposition, specifically of presentation and model, can solve in JFC implementations, but they are enough of a piece that they should serve as good puzzles to exemplify a broader design approach.

Installation of labs that require huge data sets presented a bit of a problem given that we want to be able to deploy the labs easily. Rather than packing up big databases or requiring Internet connectivity to our own servers in-class, we've built, with each of the example and lab tracks for the chapter, Java software that will generate the data onsite. So, the **Inventory** example comes with an **InventoryGenerator** class that builds data files of a given size. This generator is run three times by the **BuildInventories** script, creating the files **inventory\_small.dat**, **inventory\_medium.dat**, and **inventory\_huge.dat**, which are referenced in the student guide. If there are problems during installation or these files are mysteriously missing, try compiling and running the generator class/application by hand. It takes a single parameter, which is a number of items to generate. The resulting file size will be this number times ten plus four, so the three files are generated with five hundred, five hundred thousand, and five million items, respectively. If the student machines are very memory-starved or very memory-rich, you may find that you have to generate files of different sizes for the demonstrations to be effective. It would be wise to test this thoroughly on the instructor machine or on a typical student machine prior to covering the chapter.

Similarly, the **Revolution** lab track requires a database of text files. This too should be generated automatically on installation. If not, the application class is **StartARevolution** and it takes a single parameter as well. Since sheer size is not as much the focus of the labs, just run this with a parameter of 1. Make sure to run the **StartARevolution** script in each new workspace (example step or lab working directory) before doing coding and testing.





## Module 3: Advanced GUI Design with JFC

The beginning of this module will again be a change of pace and focus for students. Where the first and second modules concentrated mostly on developing knowledge of the control set, this module moves in a new direction by considering the problem of how to organize many controls, whether those controls are simple buttons or complex tables. A scattering of techniques is presented in these three chapters, and most or all of them stand alone nicely.

### Chapter 1:

The material here is fairly simple. Understanding the relationship between the scroller and the scrollee (scroll pane and scrollable, that is) is the key to the scrolling material. Once students have that, the individual methods and implementations should be easy enough. Encourage students to consider the practical application of each of the pane classes: which might be best used where, and how and when might they be combined effectively?

### Chapter 2:

Where chapter 1 discusses compression of broader UIs into smaller spaces, this chapter offers means of UI expansion and growth, via popup elements. Mostly the prepared elements in the Swing API are considered: option pane and its various dialog types, file chooser, color chooser. Students may be curious about more custom development working directly from JDialog, and this makes for a good add-on discussion. Again, the labs here are fairly simple. Most of the work is in orienting students within a larger body of starting code, especially in the Meals application. Instructors are advised to become familiar with this one so as to be ready for the usual onslaught of "where is" questions.

### Chapter 3:

This chapter holds the only material with any internal dependencies, and as such it's a little harder to skip around, if skipping is what you want to do. Data transfer must be covered before clipboard, and typically drag-and-drop is that much harder to understand without having first seen transferable objects in use in the simpler clipboard context. Also, the drag-and-drop material is probably the least practical as of this writing, since the actual JRE support for it is still somewhat shaky. You may choose to skip the optional lab, and focus on the theory for now. Students may notice anomalies in the behavior of the lab code, such as incorrect cursors, inability to transfer to/from other applications (compare to the clipboard lab), etc. Again, stick to theory as much as possible here, until Sun works out the kinks.





## Feedback

We truly do welcome feedback, both of a specific nature (pointing out mistakes) and general suggestions. For the former sending email with a numbered list of corrections would be most helpful.

Please send feedback to:

Will Provost  
Capstone Courseware  
<mailto:provost@capstonecourseware.com>  
[www.capstonecourseware.com](http://www.capstonecourseware.com)

