Background for Graphics

The "wheels" package hides a lot of detail about how graphics programming is done in Java. This very brief, very incomplete writeup is intended to prepare you to start looking at some of the details.

Some Problems

Let’s get these out in the open right away. Being aware of them can save you a lot of time and frustration.

- There are almost always many very different ways to achieve the same thing. Different programmers have different favorite ways to do things. Reading examples from many different sources will probably confuse you at first since one person will say, “do it this way” and another will say, “no, do it that way.” Just find a way that makes sense for you and stick with it. Try to seek out examples that adopt the same approach.

  I personally like the approach taken by Fred Swartz (see list of resources, below).

- There is a wealth of material on the World Wide Web about Java programming, but be careful — much of the material and many of the examples are many years old and reflect practices that either no longer work or, at best, fail to take advantage of newer language features. I’ve listed some sources below that I’ve checked on — they should give you reasonably up-to-date information. I will also put these on the course Web site. Part of this problem is owing to the rapid development of Java and the tendency of old Web sites to linger long after they have been abandoned by their creators. See section on versions, below.

- Graphics programming is inherently complex. Don’t expect it to be trivial to make a simple animation of a circle going across the screen.

Some Useful Terms

You’ll encounter these in your reading:

1. **Java SE, JRE, JDK** — SE means “Standard Edition.” There are several pieces to Java SE. The JRE, or “Java Runtime Environment,” enables users to run Java programs — for example, Java applets over the Web. However, in order to create your own Java programs you also need a set of developer’s tools (such as `javac`, `java`, and several other commands). You can get both of these — the JRE and the developer’s tools — through something called the JDK, or “Java Developer’s Toolkit.” Go to [java.sun.com](http://java.sun.com) for more.
The current version of Java SE is Java SE 6. Our labs have Java SE 5, which is almost the same (the differences are not things we would notice in a CMPSC 111 course). There is a lot of confusion about version numbers. For instance, Java SE 5 is also known as J2SE 5.0 or J2SE 1.5.

As long as the version of Java you are using is 1.5 or 5.0 or higher, you’re probably in good shape. Version 1.4 does not have things like the `Scanner` class and other features that we’ll be using, but otherwise you should be able to learn from sample programs and tutorials that deal with that version of Java. I suggest you avoid tutorials and examples that are clearly identified as being for earlier versions of Java. They will probably still work, but they may use outdated methods or approaches. One way (but unfortunately not a foolproof way) to tell that you are using a program for an outdated version of Java is to receive a compiler warning about using “deprecated” features. Features that have been “deprecated” often still work, but they are not recommended.

2. **GUI** — graphical user interface. Any piece of software that lets the user communicate with the computer by means of graphical (as opposed to pure text-only) input and output. GUIs usually involve one or more of the following: buttons, icons, mouse-clicks, menus, pop-ups, etc. Much of the material on graphical programming in Java falls under the heading of “GUI programming.”

3. **awt** — the Abstract Window Toolkit. A collection of tools (buttons, frames, checkboxes, etc.) for constructing GUIs in Java.

4. **Swing** — a newer graphical toolkit that builds on and greatly expands the tools available through **awt**. If you are looking at tutorials or examples online that use classes like “Button” or “Panel” or “Frame”, you are probably looking at old **awt** examples. The newer, better Swing versions of these are named “JButton”, “JPanel”, “JFrame”, etc.

Some Useful Resources

- First and foremost is your textbook. Chapters 6 and 7 give an introduction to graphics that is consistent with the notation you have already learned, and the programs in the book can be downloaded from the class Web site. Start here!

- [http://leepoint.net/notes-java/index.html](http://leepoint.net/notes-java/index.html). This collection of notes is not yet complete, but what’s there is very good. The author, Fred Swartz, is on the faculty of University of Maryland University College Europe. He is also working on a set of notes called “Java Basics” (see [http://www.leepoint.net/JavaBasics/index.html](http://www.leepoint.net/JavaBasics/index.html)), but unfortunately he hasn’t gotten far yet.

- [http://java.sun.com/docs/books/tutorial/java/TOC.html](http://java.sun.com/docs/books/tutorial/java/TOC.html) — an introduction to the Java language from Sun Microsystems (the company that invented Java). This covers the basics about classes, objects, etc. — no graphics in here, but it might be of interest to you as a complement to our textbook.

- [http://java.sun.com/developer/onlineTraining/new2java/divelog/](http://java.sun.com/developer/onlineTraining/new2java/divelog/) — “Building an application, part 1”. Walks you through a graphical application step by step. This is dated
2003, so some aspects of Java have been updated since then, but it still seems like a good
tutorial.

tinuation of the above. There are three more parts (same URL, but use “part3”, “part4”,
and “part5” as the last part of the address). These get pretty advanced.