Chapter 5: Summary

Summary

It may not seem like it, but we've rushed through the topic of inheritance, interfaces, and abstract classes very quickly. We have learned the basics about using them, and there's still a lot more to learn about them. But we'll save that for another course.

We've learned the basic essentials in this lesson:

- A Java interface defines (not surprisingly) an interface for a group of similar concrete classes. It sets up the methods that a programmer calls when using objects of that type. In our case, the objects are lists.
- An abstract class defines any methods that don't depend on the final implementation. That eliminates duplicate code in the concrete classes. We were able to define our size() and toString() methods independently, without caring whether the concrete class used an array or an ArrayList.
- A concrete class implements the remaining methods using the data structure that is most efficient for a specific process. A programmer can use that class to create objects in an application.

Next time, we'll start down an entirely different path. We've been learning how to do a number of things in Java, but we've done everything in text-based console applications. In the next lesson, we'll start learning how to use Java to write stand-alone desktop applications using a *graphical user interface*, usually called a *GUI*. That means we'll be using windows, menus, the mouse, and all that fun stuff!

Don't forget to do the assignment. I wouldn't want you to miss the fun of actually using the two concrete classes we developed in this lesson.

If you have any questions, please let me know in the Discussion Area.

Until next time, then.