Inheritance rocks! So let’s do some problems that revolve around this concept.

**Problem #1:** Integer is a class that has already been written and is part of the Java API. Suppose you attempt to extend the Integer class and add a new method that returns the integer as a String that is written in base-32. Recall base-2 is binary, base-8 is octal, base-16 is hexadecimal, etc. Explain why your code cannot succeed.

**Problem #2:** Random is a class that has already been written and is part of the Java API. It has various methods to generate random integers, random doubles, etc. However, Random does not have a method that will generate a random integer between two specified integers (like 0 and 10). That would be very handy! Using inheritance, write a class called ImprovedRandom that can do *everything* that Random can do, but also add this handy method. Now write a tester for your new class.

Note: Your class is supposed to do *everything* that Random can do, so pay careful attention to what constructors should be possible. i.e., if Random can be instantiated with a “seed” parameter, can yours? I hope so!

**Problem #3:** StringTokenizer is a class that has already been written and is part of the Java API. This class takes a string filled with words, and each time the method nextToken() is called, it returns the next word in the string. I describe StringTokenizer in my online “Introduction to Programming” notes in the section on “File I/O”. Create a new class that
acts just like a StringTokenizer but has a new method that returns all of the words in a single array.

For example, if the string is “This class rocks”, then your new method will return an array where the 0th element is “This”, the 1st element is “class”, and the 2nd element is “rocks”.

Note: Your class is supposed to do everything that StringTokenizer can do, so pay careful attention to what constructors should be possible.

Problem #4: Your code should always be well commented. For example, in problem #3, your new method should have a comment that sounds like the description for nextInt(int n) in Random. (See the API!) For this problem, I would like you to use javadoc to generate an appropriate API for problem 3. Turn in the html files associated with your classes (e.g., ImprovedRandom.html), but don’t bother to send me the rest of the API documents.

Hint: Do the reading assigned above!

Lab problem: Let’s start to use our robots! We’ll begin with a simple problem this week so that you can learn the process of communicating with your robot. I want you to write a program that plays a short, simple song of your choice (for example, “Mary Had a Little Lamb”). To do this you will need to

(1) Install Java on your NXT brick. Read attached instructions. We have already installed LeJOS on all of the lab computers so you can skip that part of the instructions. (If you turn on your NXT brick and it says “LEJOS”, then someone has already installed Java for you. Aren’t you lucky!)

(2) Read the LeJOS API. It is installed on the lab computers at C:\lejos_nxj\docs\apidocs\index.html. These are the classes that are available for you to use on your robot. A lot of classes from the Java API are missing – that’s because the NXT brick has limited memory and can’t hold everything.
(3) Test your installation with the “hello world” program in the instructions.

Note that LCD class displays strings. You can read about it in the API.

(4) Now go to my online notes and download the package of robot code that I have pre-written for you. In particular, look at the Tone class. This class has a single method that plays musical notes. The method will require you to specify the duration of the note in centiseconds. So a “1” means 1/100 of a second, and a “100” means 1 second.

(5) Using the Tone class, write code that plays a simple song, and put it on the NXT brick just like you did with the “hello world” program.

(6) Warning: Writing and debugging code for small portable hardware devices can be a process of “trial and error”. So leave plenty of time, and be prepared for annoying and time-consuming quirks due to the hardware (failing batteries, failing USB ports, etc.).

Please turn in your code and put your name on a piece of paper near your NXT brick so I can play your tune.