Inner Classes, Anonymous Classes, and Nested Classes

In Lab 4 we will make substantial use of classes inside of other classes. Java allows several kinds of such classes:

- A *nested class* is a static element of another class:

```
public class A {
        int x;
        private static class B {
                ...
        }
        ...
}
```

This is useful when class B doesn't need to make use of any of the properties of class A. If class A has a generic type such as MyLinkedList<E> then class B can't refer to type E.

- *Local classes* are short classes defined inside a method. You aren't allowed to give visibility attributes because they are only visible within the method where they are declared. You might use a local class if inside a method you need to create a comparator class so you can sort an array of objects.

Neither nested classes nor local are much help with Lab 4.

- *Inner classes* are non-static classes defined within other classes. An inner class is allowed to access the properties of its surrounding class. You probably used an inner class to define the Node type in your linked implementation of queues in Lab 3.

Anonymous classes are, of course, classes to which no name is given. They are only useful in situations where there is only one point in the code at which you will construct a new object of the class and where the point of the class is to make concrete an abstract class or interface. For example, suppose you have an interface Inter. With a named class you would say

```
class MyInter implements Inter {
        ...
}
```

and then when you need to construct an object of my inter you might say

```
Inter foobar = new MyInter();
```

With anonymous classes you would skip the declaration of class MyInter and where you create a new object say

```
Inter foobar = new Inter() {
        ...
}
```

Between the braces you would put the entire body (everything but the header) of the MyInter class declaration. Since there is no name for the anonymous class there is no way to make a constructor for it. The instance variables for the anonymous class can be initialized when they are declared.

I find anonymous classes hard to read but a lot of people use them so you should at least understand how they work. Whether you use them in Lab 4 is up to you.