CS61B Lab #6

1. Iterators and Iterables (Background)

Many kinds of object that you define are either intended to be used as collections of values or to consist in part of a collection of values. An iterator is an object whose purpose is to traverse the values in one of these collections, yielding them one at a time to the iterator's client. The standard Java library defines a standard interface that iterators can implemnt:

```
    package java.util;

    public interface Iterator<Value> {
        /** Returns true iff this iterator has another value to deliver. */
        boolean hasNext();
        /** Returns the next value from this iterator, and advances to
         *  the next.
         *  Precondition: hasNext() must be true.
         */
         Value next();
         /** Remove the last value delivered by next() from the
          *  underlying collection of values being iterated over.
          *  Optional method.  May be called at most once per call to
          *  to next(). */
         void remove();
    }
```

Classes that wish to function as collections of values can implement a method that returns an appropriate Iterator, which can then be handled in a generic way. For example, if L is a List<String>, you can write

```
    for (Iterator<String> i = L.iterator(); i.hasNext();) {
        String value = i.next();
        System.out.print(value + " ");
    }
```

or, if you prefer,

```
    Iterator<String> i = L.iterator();
    while (i.hasNext()) {
        String value = i.next();
        System.out.print(value + " ");
    }
```

This idiom is sufficiently common that Java introduced a special syntax for it. If L has a type that implements Iterable<String> (as List<String> does), then the loops above may be written

```
   for (String value: L) {
       System.out.print(value + " ");
   }
```

The Iterable interface looks like this:

```
    package java.lang;
    public Iterable<Value> {
        /** Returns an iterator over my values. */
        Iterator<Value> iterator();
    }
```

2. Creating Filters

The purpose of this exercise is to give you a chance to exercise the mechanics of using Java's object-oriented machinery. It's best to work it through in lab, where you can ask the TAs for help.

After reading through the two classes utils.Predicate and utils.Filter, try filling in the FIXME places in the other files to get the program to compile (with the 'make' command) and work. The main program is in the class Iterating.

3. Project.

If you have time, feel free to have the TAs help you with the project.