com1101 Lab 8

Greg Pettyjohn

February 22, 2003

Part 1 Iterating With For Loops

In this lab we will use the IRange interface to iterate over collections.

Open the `IRangeTest.java` file and study the code for the methods `containsBlueEyedPerson` and `allOverEighteen`.

1. `IRangeTest.java` contains several definitions of data structures to help speed up testing. Add a few more definitions of your own.

2. Design the method `containsPersonWithEyecolor` which consumes an `IRange` collection and a `Color` object and determines if there is a `Person` in the collection with that eyecolor.

3. Design the method `allOlderThan` which consumes an `IRange` collection and a number of years and determines whether all the `Person`s in the collection are older than the given age.

4. Rewrite `containsBlueEyedPerson` and `allOverEighteen` to use `containsPersonWithEyecolor` and `allOlderThan` as helper functions. Reuse the tests!

Part 2 Writing andMap and orMap

We can write versions of the `andMap` and `orMap` methods that will work with `IRange` collections.

1. Open the file `IPerson2Bool.java` and study the `IPerson2Bool` interface. Write a class that implements `IPerson2Bool` and can be used to determine whether a person has blue eyes. Also write a class that implements `IPerson2Bool` and can be used to determine whether a `Person` is over 18 year old.

2. The header for `andMap` can be found in `IRangeTests.java`. Write `andMap`, which consumes an `IPerson2Bool` object and an `IRange` collection and determines if all of the `Person`s in the collection satisfy the `IPerson2Bool` object.

3. The header for `orMap` can be found in `IRangeTests.java`. Write `orMap`, which determines whether at least one `Person` in a `IRange` collection satisfies an `IPerson2Bool` object.

4. Rewrite `containsBlueEyedPerson` and `allOverEighteen` to use `orMap` and `andMap`. Use the classes from step 1. Reuse tests!

5. Create examples that use `andMap` and `orMap` with anonymous inner classes.

Part 3 Using the IRange Interface in a Recursion

Here is an example of a function, `totalYears` that sums up the ages of all the `Person`s in a collection. In the function I use an extra pair of {'s to introduce a local variable `newTotal`. Why must I use the local variable?

```
// totalYears
// To sum up all the ages of the People in the given IRangeCollection
int totalYears(IRange i, int total) {
    if (i.hasMore()) {
        {
            int newTotal = total + ((Person)i.current()).age;
            i.next();
            return totalYears(i, newTotal);
        }
    }
    else
        return total;
}
```

1. Design `listPersons` which creates a `AListPerson` object using all the `Person`s in an `IRange` collection.

2. How about sort?

3. Can you rewrite these methods using a `for` loop?