Language Assignment #4: Go

Issued: Thursday, November 13
Due: Tuesday, December 2

Purpose

This assignment allows you to program in a language introduced in lecture: Go. It was designed by Robert Griesemer, Rob Pike, and Ken Thompson, at Google, in 2009.

Why Go?

Go is a garbage-collected, concurrent, object-oriented, and imperative language. It has been considered a competitor of the systems-programming duo of C and C++, but it has garbage collection. Rust is also a competitor in this arena, without garbage collection.

Documentation

Go lecture slides are at:

```
pub/slides/slides-go.pdf
```

Go is demonstrated by:

```
pub/sum/go
pub/etc/sleepsort
```

Go is too new to be described in our textbook.

Links to programming-language documentation can be found at:

`http://csweb.boisestate.edu/~buff/pl.html`

Assignment

Begin, by porting the simple banking application at:

`pub/la4`

from Java to Go. Model your Go solution on the Java solution. Thus, you will have multiple Go packages. Use the patterns we saw in:

`pub/etc/student`

Then, reimplement the `Accrue` functions, to use goroutines. This is a bit silly, since interest-accrual is too simple to deserve separate threads. In any event, use a channel, so the bank's `Accrue` function can sum the interest, and print the total. Use the patterns we saw in:

`pub/etc/sleepsort`

Hints and Advice

- An abstract class/method can be approximated with an interface that declares a function that has no definition.
- Your bank can represent its set of accounts as a map from interface pointers to interface values. An interface value is essentially an object reference:

  ```
  accounts map[*IAccount]IAccount
  ```

  If `b` is a `Bank`, you can iterate over the accounts like this:

  ```
  for _,a:=range b.accounts {
      ...
  }
  ```