# COSC 1P03 Lab 2 Jan 22-26, Winter 2024

In this lab we will be considering searching an array in the context of a data processing problem.

## Over the Horizon

Over the Horizon Utilities is an electric utility that provides electricity to homeowners. At the end of each month, it does billing for each customer. The amount charged the customer is the consumption (in kilowatt hours (kwh)) times the current rate ($1.525/kwh). The customer data is maintained as an `ASCIIDataFile`. For each customer there is an account number (`String`), customer name (`String`), a previous meter reading (from the last billing period, `double`) and a current meter reading (`double`). The program processes the customer records producing a report (`ReportPrinter`) listing the relevant customer billing data and the total billed as summary such as shown in Figure 1:

```
                    Over The Horizon Utilities
                          Billing Report

Account #          Name          Previous Current    Charge
111111     Fred Flintstone          123.4    234.5    $169.43
222222     Barney Rubble            234.5    333.3    $150.67
333333     Homer Simpson            345.6  1,345.6  $1,525.00
444444     Wile E Coyote              0.0      1.0      $1.52
           Total Billed                             $1,846.62
```

Figure 1 Billing Report

At the end of each billing period and prior to running the billing program, the utility sends meter readers to read the electricity meters. The meter reader has a tablet that runs a program to enter new current meter readings for accounts in the account file. The program presents the form shown in Figure 2.

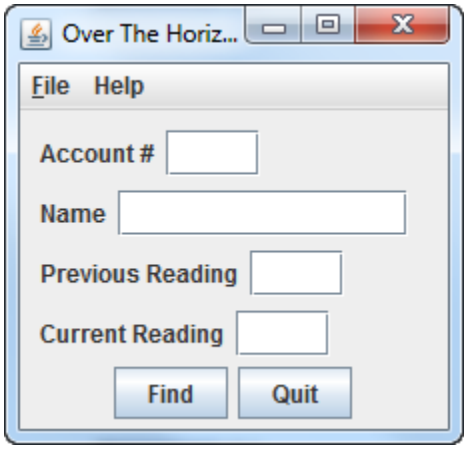


Figure 2 Meter Reading Form

The reader enters the account number for the customer (the number is on the meter) and presses `Find`. If the account number is found, the account details are displayed (Figure 3) and the reader enters the new current reading and presses `Update` and the cycle continues until the meter reader presses `Quit`.

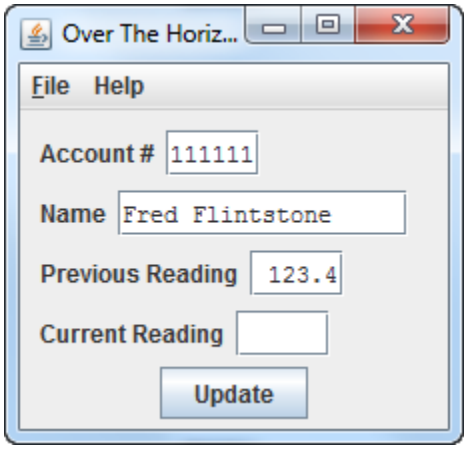


Figure 3 Meter Form with Details

When all the meters have been read, the billing program is run to prepare the billing information for the billing period.

# Exercise 1
# Account Records

*Estimated time: 30 minutes*

Download and extract the `Lab 2` folder from Brightspace onto your LabExcercise folder (make a folder). It contains a package folder `Billing` containing the Billing project which has a Billing package inside for this exercise.

As part of the `Billing` package, write the `Account` class.

It should provide a constructor:

```
public Account ( ASCIIDataFile from )
```

that reads a line containing the account information (number, name, previous reading, current reading) from a text file.

The class should provide appropriate accessor methods:

```
public String getAcctNum ( )
public String getName ( )
public double getPrevReading ( )
public double getCurrReading ( )
```

and two other methods:

```
public void takeReading ( double reading )
```

that updates the account, setting the *current reading* to the parameter `reading` leaving the *previous reading* unchanged; and:

```
public double billForUsage ( )
```

that computes the billing amount as the difference between the current and previous meter readings (consumption) times the rate and updates the *previous reading* to be the *current reading* in the account record (after `billForUsage` previous and current readings are equal).

Finally, it should include a method:

```
public void write ( ASCIIOutputFile to )
```

that writes the account information to an `ASCIIOutputFile` in the same order that the constructor reads the account information.

A skeleton of the `Account` class is found in the file `Account.java`. An implementation of the `Billing` class is included in the `Billing` package supplied. Implement and compile your `Account` class in the `Billing` package and run it using the supplied data file: `accounts1.txt` producing a new accounts file `newAccounts1.txt` and a report `report1.pdf`. The report should be the same as the example on page 1.

# Exercise 2
# Update

*Estimated time: 45 minutes*

As part of the `Billing` package, write a main class `Update`, to be run by the meter reader, that performs account update (i.e. meter readings).

The `Account` objects are stored in the file (`ASCIIDataFile`) in account number order. Since the reader will not be reading the meters in account number order, the program will have to load all account objects into an array and search the array for the `Account` with the account number as entered by the meter reader (as in Figure 2). If the account is found, it will present the information from the account with the current reading empty (as in Figure 3).

When the reader presses `Update`, it will update the `Account` object by calling the `takeReading` method.

When the reader presses `Quit`, the program will write the updated `Account` objects to a new `ASCIIOutputFile`.

A skeleton of the `Update` class is provided as part of the `Billing` package. The `setUpForm` and `fillForm` methods are supplied. Complete the implementation in 2 phases:

**Phase 1**

Write the method `loadAccts` which creates the array for the accounts, fills the array with the account objects read from the `ASCIIDataFile` and counts the number of accounts.

Write the method `writeAccts` which writes the account objects from the array to the `ASCIIOutputFile`.

Write an appropriate constructor to use `loadAccts` and `writeAccts` to read and write the accounts.

Run the program using `accounts2.txt` as input and producing `accounts3.txt` as output. The two files should be identical.

**Phase 2**

Write the method `findAccount` that searches the account array for the indicted account number returning a reference to the desired account object or `null` if there is no such object.

Modify the constructor to load the account array and then, until the user presses `Quit`, presents the meter reading form with all fields empty (Figure 2). When the user enters an account number and presses `Find`, it locates account object and displays the account information in the meter reading form (Figure 3). When the user enters the current reading and presses `Update`, it updates the account object using the `takeReading` method. When the user presses `Quit`, it writes the updated account records to the `ASCIIOutput` file.

Run the program using the file `accounts2.txt` as input producing the updated accounts file `updatedAccounts2.txt`. During execution, update the records in arbitrary order (see Figure 1 for values) and then quit. The `updatedAccounts2.txt` file written should be the same as `accounts1.txt`.

Run the `Billing` class to produce the billing report using `updatedAccounts2.txt` as input and producing `newAccounts2.txt` as the output file and `report2.pdf` as the report. The report should be the same as shown in Figure 1 and `newAccounts2.txt` should be the same as `newAccounts1.txt` produced in Exercise 1.