CS410J Project 1: Designing a Phone Bill Application (7 points[1])

In this project you will create the fundamental `PhoneBill` and `PhoneCall` classes that you will work with for the duration of the course.

Goals: Extend classes that you did not write and perform more complex command line parsing

The `edu.pdx.cs410J` package contains two abstract classes, `AbstractPhoneBill` and `AbstractPhoneCall`. For this project you will write two concrete classes in your `edu.pdx.cs410J.`*login* package: `PhoneBill` that extends `AbstractPhoneBill` and `PhoneCall` that extends `AbstractPhoneCall`[2]. Each of your classes must implement all of the abstract methods of its superclass.

A `PhoneBill` has a customer name[3] and consists of multiple `PhoneCall`s. A `PhoneCall` is initiated by a person with a given phone number at a given time[4], is received by a person with a given phone number, and terminates at a given time. For this assignment, all of this data should be modeled with `String`s. Additionally, you may ignore the `getStartTime` and `getEndTime` methods.

You should also create a `Project1` class that contains a `main` method that parses the command line, creates an `PhoneBill` and a `PhoneCall` as specified by the command line, adds the `PhoneCall` to the `PhoneBill`, and optionally prints a description of the `PhoneCall` returned by its `toString` method[5]. Your `Project1` class should have the following command line interface[6]:

```
usage: java edu.pdx.cs410J.<login-id>.Project1 [options] <args>
  args are (in this order):
    customer                Person whose phone bill we're modeling
    callerNumber            Phone number of caller
    calleeNumber            Phone number of person who was called
    startTime               Date and time call began (24-hour time)
    endTime                 Date and time call ended (24-hour time)
  options are (options may appear in any order):
    -print                  Prints a description of the new phone call
    -README                 Prints a README for this project and exits
  Date and time should be in the format: mm/dd/yyyy hh:mm
```

Note that multi-word arguments should be delimited by double quotes. For instance the `customer` argument could be `"Brian Griffin"`. However, dates and times should **not** be quoted. (they are two separate command line arguments) The following dates and times are valid: `1/15/2018 19:39` and `01/2/2018 1:03`[7]. Phone numbers have the form `nnn-nnn-nnnn` where `n` is a number `0-9`.

[1] 6 for code, 1 for POA

[2] Be aware that you should **not** modify any of my code. When I test your code I will use my version of the code, not yours. In fact, the `Submit` program will not allow you to submit my code. Remember that the `Submit` program can submit more than one file at a time.

[3] Customer names can contain any character include numbers.

[4] Your program should accept times and dates that have already occurred as well as ones that occur in the future.

[5] Note that `PhoneCall`'s `toString` method is inherited from `AbstractPhoneCall`. You do not need to override it.

[6] You can learn more about the `README` option in the "Documenting Your Code for CS410J" handout on the course's website.

[7] That is, the month and the day can be expressed as either 1 or 2 digits. The year should always be four digits.

Error handling: Your program should exit "gracefully" with a user-friendly error message under all reasonable error conditions. Examples of such conditions include

- Something is missing from the command line or there are extraneous command line arguments
- The format of the day or time is incorrect or the phone number contains characters other than - that are non-numeric

The class files for classes in the `edu.pdx.cs410J` package can be found in `/u/whitlock/jars/cs410J.jar`

You should submit `Project1.java`, `PhoneCall.java`, and `PhoneBill.java` using the submit program. You can learn more about the `Submit` program in the "Instructions for submitting projects for CS410J" handout on the course's website.

To get you started with the project, there is a Maven archetype for the Phonebill project.

Before you can generate the archetype, however, you must configure Maven to look for archetypes hosted in my Maven repository. This is done by adding the following in your `settings.xml` file in the `.m2` directory in your home directory.

```
<settings xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      http://maven.apache.org/xsd/settings-1.0.0.xsd">

  <profiles>
    <profile>
      <id>davidwhitlock-bintray</id>
      <repositories>
        <repository>
          <id>archetype</id>
          <url>https://dl.bintray.com/davidwhitlock/maven/</url>
          <releases>
            <enabled>true</enabled>
            <checksumPolicy>fail</checksumPolicy>
          </releases>
          <snapshots>
            <enabled>true</enabled>
            <checksumPolicy>warn</checksumPolicy>
          </snapshots>
        </repository>
      </repositories>
    </profile>
  </profiles>

  <activeProfiles>
    <activeProfile>davidwhitlock-bintray</activeProfile>
  </activeProfiles>

</settings>
```

```
$ mvn archetype:generate \
  -DarchetypeGroupId=edu.pdx.cs410J \
  -DarchetypeArtifactId=phonebill-archetype
Define value for groupId: : edu.pdx.cs410J.<login-id>
Define value for artifactId: : phonebill
Define value for version:  1.0-SNAPSHOT: :
Define value for package:  edu.pdx.cs410J.<login-id>: :
Confirm properties configuration:
groupId: edu.pdx.cs410J.<login-id>
artifactId: phonebill
version: 1.0-SNAPSHOT
package: edu.pdx.cs410J.<login-id>
 Y: : Y
```

The archetype creates the `Project1` class and a class for testing it, `Project1Test`

```
+- phonebill/
   +- pom.xml  (Dependencies and reporting configuration)
   +- src/
      +- main/  (program source code)
         +- java/
            +- edu/pdx/cs410J/login-id/
               +- PhoneCall.java
               +- Project1.java
         +- javadoc/ (files for JavaDoc)
            +- edu/pdx/cs410J/login-id/
               +- package.html
      +- test/  (unit tests)
         +- java/
            +- edu/pdx/cs410J/login-id/
               +- PhoneCallTest.java
         +- javadoc/ (files for test JavaDoc)
            +- edu/pdx/cs410J/login-id/
               +- package.html
      +- it/  (integration tests)
         +- java/
            +- edu/pdx/cs410J/login-id/
               +- Project1IT.java
```

The project should compile and run out-of-the-box. The ‘verify‘ phase compiles all of the source code, runs the unit tests, creates the jar file, and runs the integration tests.

```
$ mvn verify
```

The archetype configures a bunch of cool reports to run against your project.

```
$ mvn site
```

Open `target/site/index.html` and view the reports generated for your project.
The jar file created by the archetype is an “executable jar” that runs your `Project1` main class.

```
$ java -jar target/phonebill-1.0-SNAPSHOT.jar -README
```