Assignment 1

Due: January 22nd, 2017 before 11:59pm

Objectives

- Practice implementing classes
- Review arrays and Strings
- Practice reading and understanding specifications
- Exposure to more complete testing

Introduction

In this assignment you will implement classes that could be used by a Contact Manager like those found on many smart phones.

You will create a `Contact`, which will store the contact name and a list of phone numbers associated with the contact. Our phone numbers will store both the digits associated with the phone number, but also a label like "Work" or "Home" or "Cell".

Quick Start:

1. Download `a1tester.java, PhoneNumber.java, PhoneNumberList.java, and Contact.java`
2. Read the comments in `PhoneNumber.java`
3. Implement one of the methods in `PhoneNumber.java, PhoneNumberList.java, or Contact.java`
4. Compile and run the test program `a1tester.java`
5. If no errors reported by test program see Grading section of this document
6. goto 3

You should implement your solution in this order: `PhoneNumber.java`, then `PhoneNumberList.java` and finally `Contact.java`.

Understanding the test program: a1tester.java

You've been given a program that will test your implementation of `PhoneNumber`, `PhoneNumberList` and `Contact`.

One of the first things you should do after downloading the source code files is to run the test program.

Compile the test program by typing:

```
javac a1tester.java
```

Run the test program by typing:

```
java a1tester
```

You should see the following output:

```
PhoneNumber testing.
Failed test: 0 at line 51
```

The tester is reporting that your implementation is failing the very first test. This is hardly surprising, since you haven't written any code yet!

There are 32 tests and your goal for the assignment is to pass them all.

What to do

You should start working in `PhoneNumber.java`

The comments above each method to be implemented are supposed to provide an unambiguous description of exactly what the method is to accomplish. Read the comments carefully before implementing the method.

First, implement the constructor that takes a single parameter:

```
     public PhoneNumber (String theDigits)
```

Then implement `getDigits` and `getLabel`.

Now, run the tester, you should have passed tests 0 and 1.

Then implement the second constructor:

```
     public PhoneNumber (String theDigits, String theLabel)
```

Run the tester again and you should have passed tests 2 and 3.

Now implement `setDigits` and `getDigits`. Run the tester and you will have passed test 4.

Now implement `setLabel` and `getLabel`. Run the tester and you should have passed test 5.

Now implement the `toString` method. Run the tester and you should have passed test 6.

Finally, implement the `equals` method. Remember that we don't consider the label when comparing phone numbers, only the digits.

Run the tester and you should have passed tests 8 and 9. Congratulations you've completed about 1/3 of the assignment!

Now move on to `PhoneNumberList.java`. I suggest you implement the methods in the following order: `constructor, size, add, get, find`, and finally `remove`.

Finally, work on `Contact.java`. This is the easiest to implement, since you will primarily be calling methods on the instance of `PhoneNumberList` you allocate in the constructor. You shouldn't have to write more than a few lines of code in each method.

Submission

Submit your `PhoneNumber.java PhoneNumberList.java and Contact.java` using Connex. **Please be sure you submit your assignment, not just save a draft**.

A reminder that it is OK to talk about your assignment with your classmates, and you are encouraged to design solutions together, but each student must implement their own solution.

We will be using plagiarism detection software on your assignment submissions.

Grading

If you submit something that does not compile, you will receive a grade of 0 for the assignment. It is your responsibility to make sure you submit the correct files.

NOTE: Test cases are numbered from 0. The first test is test 0, the last test is test 31.

Requirement	**Marks**
You submit something that compiles	1
1 mark for each test case you pass	Up to 32

Total 33