AP Computer Science Homework 11

Due date: Sunday, December 4, 2016
Instructor: Mr. Alwin Tareen

Part A: Create a `CellPhone` Class

- Write a class called `CellPhone` that will simulate a Simple Message System(SMS) text message inbox, similar to that on a typical cellphone.
- The text message inbox will consist of an **array** data structure, which is declared as an instance variable called `inbox`.
- Each text message contains a **message** and a **sender name**, in the form of a `String`. This information is encapsulated in the `TextMessage` class, which means that each element in the `inbox` array is an object of data type `TextMessage`.
- The `TextMessage` class has been provided for you. Be sure to look over it, to understand the various methods that are available.
- *Hint:* This assignment requires extensive use of the operator `null`. It signifies that no object is present at that particular memory location. Think of it as a kind of "zero", but especially intended for use with objects.
- The `CellPhone` class provides a management system for all of the text messages in its inbox. Text messages can be added, retrieved, searched or deleted. The class should have the following properties:
- Instance variables:
  - `private TextMessage[] inbox;` This is the text message inbox. It has been provided for you.
- The constructor: `public CellPhone()`
  - This sets up the `inbox` to be of type `TextMessage` and of size `8`. It has been provided for you.
- Mutator methods:
  - `public void insertTextMessage(int i, TextMessage text)` This places the text message into the array `inbox` at position `i`.
  - `public void deleteMessage(int i)` This deletes the text message at position `i` of the array `inbox`.
  - `public void clearMessages()` This deletes **all** of the messages in the array `inbox`.
- Accessor methods:
  - `public int messageCount()` This returns the numbers of text messages that are present in the array `inbox`.
  - `public String getMessage(int i)` This retrieves the contents of the text message that is at position `i` of the array `inbox`.
  - `public boolean searchSender(String name)` This searches for the presence of `name` among all of the senders in the `inbox`. If there is a match, this method returns `true`, otherwise, it returns `false`.

- Other methods:
  - `public String toString()` This returns a string containing the object's data. It has been provided for you.
  - `public TextMessage getTM(int i)` This has been included for testing purposes.
  - `public TextMessage[] getInbox()` This has been included for testing purposes.
- You are provided with the files `TextMessage.java`, `CellPhone.java`, `CellPhoneTest.java`, and `CellPhoneJUnitTest.java` to develop this program.
- Write your code in the file `CellPhone.java`, in the area indicated by `// YOUR CODE HERE`.
- When you have finished writing the `CellPhone` class, you may run the `CellPhoneTest.java` test bench. Your output should look like the following:

```
Quantity of messages: 6

Message at index 2: Meet me for lunch

Search for Deandra Miller: true
Search for Max Riley: false

Position: 0
Message: Order some pizza
Sender: Allen Preston

Position: 1
Message: Send for coffee
Sender: Larry Craig

Position: 2
Message: Meet me for lunch
Sender: Tristan Ruben

Position: 3
Message: Gym session tonight
Sender: Mallory Jones

Position: 4
Message: Bring math textbook
Sender: Zoe Smith

Position: 5
Message: Buy concert tickets
Sender: Deandra Miller


Deleting message at index 4.

Position: 0
Message: Order some pizza
Sender: Allen Preston
```

```
Position: 1
Message: Send for coffee
Sender: Larry Craig

Position: 2
Message: Meet me for lunch
Sender: Tristan Ruben

Position: 3
Message: Gym session tonight
Sender: Mallory Jones

Position: 5
Message: Buy concert tickets
Sender: Deandra Miller


Clearing all messages.
```

- On your BlueJ project window, you should see a button labelled Run Tests. Press this button to run the JUnit tests.
- You should see a BlueJ: Test Results window pop up. If everything is correct, you should see a green bar that indicates that your code has passed the JUnit tests. If your program is incorrect, you will see a red bar. You can click on the method name to get more information about the problem. Otherwise, just click on the Close button, and you can go ahead and upload this program to Web-CAT.

Part B: Submission

- Submit your Java program CellPhone.java by uploading it to the Web-CAT automated grading platform.