EspressOS

A message to all students about posting on Ed

If you have a question to ask on ED please search before asking.

If you post a question, please use the test name as your title of your thread. This would help significantly for anyone else looking.

If you find someone who didn't include the test name in the title, please suggest them to do so.

Make life easy to search for a problems/solutions with a testname. Keep Australia clean!

Due date

11:59PM Sunday 04 November 2018 AEST

You are tasked with writing a phone OS for EspressOS Mobile that will be installed on their products. You are required to write the software using the Java programming language. EspressOS OS must support the following features:

- Battery life
- Network connection
- Signal Strength
- Charging a phone
- Manage contact data
- Being able to copy contacts
- Delete contacts
- Add contacts
- Update contact details
- Copy contacts
- Manage messages
- Add messages
- Clear messages
- Get latest and oldest messages
- Run applications

The given properties of the class cannot be removed and their data type and modifiers cannot be changed.

You are provided with a scaffold and comments that describe the methods required to implement. Each method's comments desribes the process necessary to implement.

Factory Defaults

Every phone manufactured and installed with EspressOS OS. The default factory settings are:

- Phone is off
- Phone has battery life (25)
- Not connected to a network
- No Signal (signal strength is 0)

Owner contact factory default on the device:

- First Name: EspressOS
- Last Name: Incorporated
- Phone Number: 180076237867
- One message should be included under this contact: `"Thank you for choosing EspressOS products".`
    - Stored as `"EspressOS: Thank you for choosing EspressOS products"`
- The contact list should not contain ANY other contacts on first boot.

If the factory defaults do not match it can be suspected that the hardware contains a fault and requires inspection. By ensuring the software clearly adheres to the correctness of what is specified, this can be ruled out.

Battery and Charging

The OS needs to keep track of the battery level and implement functions related to battery changing, charging and status. The battery level is represented as an integer between 0 and 100 inclusive. **[0, 100]**

- isPhoneOn
- getBatteryLife
- changeBattery
- chargePhone
- usePhone
- setPhoneOn

isPhoneOn

This method checks if the phone is on or not

getBatteryLife

Retrieves the battery life which is represented as value between 0 and 100.

changeBattery

Changes the battery and therefore changing the battery level. The phone is switched to the off state after this operation and the battery life is updated. If the new battery's level is outside of the range accepted (**n** < 0 OR **n** > 100) then it should be rejected and no update should occur.

chargePhone

The phone is charged and battery life increases by 10. In the event that the battery life exceeds 100, the charge becomes 100. A charge would not occur and the method should return false in the event that nothing has changed.

usePhone

will reduce the battery level by **k** units of battery level. The phone will turn off if the use causes the battery level to reach 0.

setPhoneOn

Turning the phone on will reduce the battery level by 5, if the battery level is < 6 the phone should not power on.

Network Connectivity and Signal

A baseline feature to a phone is determining the network connectivity status and updating it.

The network status has two parts. Network is connected and Signal strength. Signal strength is represented as a range between 0 and 5 inclusive **[0,5]**. 0 representing that the phone is not connected to a network while all numbers > 0 infer that the phone is connected to a network.

- isConnectedNetwork
- disconnectNetwork
- connectNetwork
- getSignalStrength

- setSignalStrength
- changeAntenna

isConnectedNetwork

Reports if the phone is connected to a network

connectNetwork

Connects to a network if needed otherwise does nothing. When connecting to network, sets the signal strength to 1 if the signal strength is currently set to 0. Sets the signal strength to the last known value of signal strength if it is **not** currently set to 0. If the network needs to connect, this process will reduce the battery life by 2.

disconnectNetwork

Disconnects from a network and sets the signal strength to 0

getSignalStrength

Returns an integer value between 0 and 5 **[0,5]**, that represents the signalStrength

setSignalStrength

Sets the signal strength to n, where n must be in the range of **[0,5]**. If n is inside of the range of **[0,5]** the method will be successful.

changeAntenna

Changes the antenna with different antenna object. This will be used to replace broken antennas. Signal strength starts at **0** and is disconnected from a network.

If the phone is not connected to a network and n > 0, it will connect to a network and reduce the battery life by **2**.

If the phone is connected to a network, the signal strength value is updated. If the signal strength is zero, it will disconnect the network, while a signal strength of > 0 will not change the network connected status.

If n is outside of the range of **[0,5]**, or the phone is off, this method should not affect the mobile and specify that it did not successfully update.

Contact Management

The OS allows for the user to manage contacts by being able to search, remove and add contacts. The maximum number of contacts that can be stored on the device is 10 plus the owner contact.

The following methods require to be implemented:

- searchContact
- addContact
- removeContact
- getCopyOfOwnerContact
- getNumberOfContacts

searchContact

A user would want to find contacts that are stored on their phone. Given a name a user could use an input, the OS should check to see if the contact's first name or last name match the given input.

The method can return more than one result if the string is matched multiple times. If the phone is off, the method should not proceed to execute and instead return no entries.

addContact

Given a EspressOSContact, the OS should add this contact to the contact list. Only when there is enough space to do so. If phone is off, the method should not add a contact and return that adding the contact failed.

removeContact

Given a EspressOSContact, the OS should remove this contact from the contact list. It is successful if the contact was found and removed. Otherwise failed. If phone is off, the method should not remove a contact and return that adding the contact failed. Invalid contact, such as null, will result in fail.

getNumberOfContacts

Return the number of active contacts. This is possibly less than the maximum.

Contacts

A baseline feature that is required to be implemented is contact management. Each contact has a:

- First name

- Last Name
- Phone Number, cannot be less than 6 digits or greater than 14 digits.
- and Chat History

The fields First Name, Last Name and Phone Number can be updated by the user. A first and last name can be of any length and cannot be set to null. Each contact will have these methods associated with it:

- getFirstName
- getLastName
- getPhoneNumber
- updateFirstName
- updateLastName
- updatePhoneNumber
- copy

getFirstName, getLastName, getPhoneNumber

Retrieves the respective properties associated with the method name.

updateFirstName, updateLastName, updatePhoneNumber

Allows updating/changing the properties associated with the method name.

copy

This method allows a contact to be duplicated, this would be used if a contact has two phone numbers and the user would like to duplicate the user and update the phone number on one of them. This method should create a copy of **EspressOSContact** object.

Messaging

Messages are stored for each contact on the phone. Each contact can contain a maximum of 20 messages and once messages exceed that limit it will overwrite existing messages.

There are 4 methods that are required to be implemented:

- addChatMessage
- getLastMessage
- getOldestMessage
- clearChatHistory

addChatMessage

When a message is sent to the phone, the EspressOS needs to store it and be able to retrieve it based on the contact.

The message format of a chat message when stored in the chat history is `whoSaidIt + ": " + message`. When two contacts communicate, **the first name** is `whoSaidIt`.

getLastMessage

This method should retrieve the last message from a contact, if this contact has no messages, the method should return null.

getOldestMessage

This method should retrieve the oldest message in the chat history for a contact.

clearChatHistory

This method should clears the chat history for a contact.

Application (Feature)

EspressOS needs to support third party applications. You will need to implement the **Apps** feature that will allow a variety of third party applications to run on the system. Apps can implement different behaviour that will allow them to interact with different resources of the system. Currently only two behaviours are to be implemented, **Background** and **Notify**.

Apps have the following methods

- start
  This method starts the execution of an app, this method does not return any values or accept
- exit
  Will quit the process with an exit code.

EspressOS will need to implement the following methods to support **Apps**.

- install
  Installs an app on the operating system. If the object passed is null then an app will not be installed. If the app has been installed already, then the app will not be installed. The method returns the value true if the app has been successfully installed, otherwise false.

- uninstall
  Given a name of an app, it will find the app and remove it from the operating system. If the app exists and has been uninstalled it will return true, otherwise the method returns false.
- getRunningApps
  Returns a **List** of all running applications on the operating system.
- getInstalledApps
  Returns a **List** of all installed applications on the operating system.
- getBackgroundApps
  Returns a **List** of all **Background** applications on the operating system.
- getNotificationApp
  Returns a **List** of all **Notify** applications on the operating system.
- getNotifications
  Returns a **List** of all notifications that have been created and sent to the operating systems by **Notify** apps.
- run
  Given an application name, it will find the application and invoke the **.start()** method. If the application exists, the method will invoke the **.start()** method and return **true**. Otherwise the the method returns false.
- close
  Given an application name, it will find the application that is currently running and invoke the **.exit()** method. This method is commonly associated with **Background** apps as they will have asynchronous execution.

Background Apps

Background apps require implementation of two methods and utilisation of the **BackgroundThread** class that has been provided. Background apps will run in a loop until specified. They can exit by calling the exit() method associated with the **BackgroundThread** object which will allow it to end on its next loop.

- backgroundStart
  .start() is typically invoked by operating system. Since the .start() methd will be reserved for setting a **BackgroundThread,** your application's start method will be in .backgroundStart().
- getData
  Background applications provide an interface that allows other applications and the operating system to extract data from. Your background app will provide a getData method that will return an **Object** and allows an **Object** to be passed to it.

Notify Apps

Applications can utilise generate notifications and will interact with the EspressOS's own notification collection. Notify apps will create a notification that the user will be able to read.

- notifyOS
  This method will create a notification that will be sent to the operating system. Notifications can be retrieved using getNotifications method.

As part of development of these features, you will need to produce a set of test cases that will ensure that the feature has been implemented correctly. Consider test case coverage and the variety of cases that may arise with these apps.

About tests

Tests for the assignment will be released progressively.

Evaluation

This assessment contributes 10% to your final grade and is broken down into the following components.

- Automatic Test Cases (6%)
- Manual Marking (4%)
  - Implementation of the application feature (2%)
    - Create appropriate classes and interfaces for your implementation.
    - Utilise object oriented principles such as inheritance and generics.
  - Test cases (2%)
    - Test cases for Antenna and Battery.
    - Test cases for the App feature.

Academic Declaration

By submitting this assignment you declare the following:

I declare that I have read and understood the University of Sydney Academic Dishonesty and Plagiarism in Coursework Policy, and except where specifically acknowledged, the work contained in this assignment or project is my own work, and has not been copied from other sources or been previously submitted for award or assessment.

I understand that failure to comply with the the Academic Dishonesty and Plagiarism in Coursework Policy, can lead to severe penalties as outlined under Chapter 8 of the University of Sydney By-Law 1999 (as amended). These

penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgement from other sources, including published works, the internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

I realise that I may be asked to identify those portions of the work contributed by me and required to demonstrate my knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

I acknowledge that the School of Information Technologies, in assessing this assignment, may reproduce it entirely, may provide a copy to another member of faculty, and or communicate a copy of this assignment to a plagiarism checking service or in house computer program, and that a copy of the assignment may be maintained by the service or the School of IT for the purpose of future plagiarism checking.

***Warning:** Any attempts to deceive or disrupt the marking system will result in an immediate zero for the entire assignment. Negative marks can be assigned if you do not properly follow the assignment specification, or your code is unnecessarily or deliberately obfuscated.*

Description
BackgroundApp.java
EspressOSMobile.java
AppTest.java
PhoneAntenna.java
PhoneBattery.java
EspressOSApp.java
BatteryTest.java
NormalApp.java
BackgroundThread.java
Battery.java
Antenna.java
App.java
BackgroundNotifyApp.java
AntennaTest.java