A JADE-based Android Application

In this assignment, you will learn how to create an agent oriented application based on JADE using the Android OS. **If there are any changes and/or clarifications, please watch the update section below.**

UPDATES

Updates override other information specified on this page.

- 01-22-15: Task 2 has been included. Please check below for the same.
- 01-24-15: Task 3 (the last task) has been included.
- 01-24-15: Grading rubric has been added.

Android Development Environment

If you are new to Android programming environment, Anroid Dev Guide is a good place to start. You should at least read "What is Android?" and "Application Fundamentals" sections.

Before you can begin writing your first android application you need to configure your environment. Follow instruction in Android Quick Start, to get started with Android SDK. You may also run the "Hello Android" program as a warm up.

JADE Programming for Android

JADE on Android Tutorial describes how to create JADE-based applications on the Android OS. If you are not familiar with JADE, you may want to start with JADE Programming Tutorial.

Application Summary

The goal of the project is for you to understand the basic on how to create a JADE-based Android application. For this project, you are to add several features to a minimal Chat platform which is described in JADE on Android tutorial and its source code is available for download here. **You don't need a physical device for this project.** The entire application will function within a device emulator that is packaged with the android SDK.

Before you begin the implementation, **please read chapter 3 in the tutorial carefully and follow the instruction there to setup the Eclipse project to compile the code of the Chat client application**.

Using the following values when you set up the project: (you can also find the information in the tutorial)

- Project Name: chatClient
- Package Name: chat.client.gui
- Build Target: Android 2.3.3 (API level 10) (Note: If the API doesn't show up, you have not installed it. Use the Android SDK Manager to install the required API)
- Activity: unflag the "Create Activity" checkbox
- Min SDK: 10

- **Task 1: Sending announcement message to the chat room**

In this part, your task is to send a chat message that announces that someone has joined or left the chat room when some people enters or leaves the room. To do so, you will need to modify the **ChatClientAgent.java** file and upon success the activity should display the announcement message which looks similar to this.

- ## Task 2: Integrating contacts on the device

In this part, you will integrate user's contacts information on the device into the Chat platform. To make this worthwhile you will first need to add some friends to your emulator's contact manager. To do this you will need to start the emulator and follow these steps.....

1. Click on the home button on the device. This should bring you to the "desktop" of the phone.
2. Open the "Contacts" application.
3. From the contacts menu click on the menu button on the device. Click on the "new contact" option.
4. In the top box set the first and last name to "Andy ", then scroll to the bottom and click save.
5. Repeat steps 3 and 4 for the following names...
    - Bobby
    - Charles
    - David
    - Elizabeth
    - Fred
    - Greg
    - Heidi

You should now have 8 contacts stored in the emulator. These contacts will remain in the emulator even after rebooting so you shouldn't need to do this step again. Android uses a service-oriented architecture by making most of the phone's features available via **ContentProviders**. We will use the existing "Contacts" content provider to programmatically get a list of all of your friends. Information on how to retrieve data from a content provider can be found in this document.

In this task, for each current participant (please refer to the tutorial to see how to check participants) in the chat room, you need to check if he/she is in your contact list or not. If he/she is in your contact list, you should put "**[Y]**" next to his/her name, otherwise, you put **"[N]**" next to his/her name. Further, for those participants who are not in your contact list, you may want to add them to your contact list. You can simply click on the participant's name to add him/her to your list. To achieve the task, you need to

1. Load the contacts into your application.
2. Check for each participant and put "**[Y]**" next to the name of the participant who is in your list and **"[N]**" next to the name of the one who isn't. Upon success, your screen should look like this.
3. Click the name in the participants list to add the person in your contact list. You can repeat the operation to add more. You can check your contact list later to make sure your application functions properly.
4. You will add your code to the **ParticipantsActivity.java** file.

Note: Android applications need appropriate permissions to access protected features of the phone. You need to figure out what permission is necessary for reading and writing contacts and include it in the AndroidManifest.xml. Read Android Permissions to know how to include permissions.

- ## Task 3: Getting and marking your current location from the GPS provider

Android SDK provides an API for applications to access the GPS sensors on a device making it easy to develop location based services (LBS). In this part you will learn how to use the API to get a device's current location. Review the **AlertMeActivity.java** to understand how to retrieve and display a device's current position. To test it, follow these instructions –

1. Create a new Android project with the following values...

    - Project Name: sample
    - Build Target: Google APIs, API level 10
    - Application Name: sample
    - Package Name: edu.ncsu.sample
    - Activity Name: AlertMeActivity
    - Min SDK: 10 (Google API)

2. Copy the following files into the specified directories.

    - main.xml under the res/layout folder (replaces existing file)
    - AlertMeActivity.java in the edu.ncsu.sample package under the src folder (replaces existing file)

Your application needs the **ACCESS_FINE_LOCATION** permission to access the GPS sensors. Now you should know how to add the **ACCESS_FINE_LOCATION** permission to the application manifest.

1. Before launching your application, you need to create an Android Virtual Device (AVD). Use the Android AVD Manager inside eclipse to create one (make sure the target of the AVD is Google APIs, API level 10).
2. Launch your project on the AVD you just created.
3. You can emulate a GPS sensor on the AVD using the DDMS perspective in Eclipse which looks like this.
4. Your job in this part includes:
    1) Complete the icode in AlertMeActivity.java to display the GPS co-ordinates you manually send from the DDMS perspective. The activity should display the location which looks similar to this. [Hint: You need to add a LocationListener to request automatic location updates. Refer to the dev guide to figure out how to do it.]

    2) Integrate the location information with the chat application. Specifically, each time when a user sends a chat message, his/her current location should be included in the message he/she sends next to his/her name. Further, each time when you set a new GPS co-ordinates, the new location should be reflected in the next chat message the user sends. Your application should display the location which looks like this. [Hint: You need to extract the useful code snippet from AlertMeActivity.java and integrate it into **ChatActivity.java**.]

    3) Further, modify the ChatActivity.java file such that each time a place is included in the message, your code needs to check for three pre-identified locations. If some pre-identified location is less than 50m from the current position, that pre-identified location has to be displayed in the next chat message the user sends like this.
5. The function to find distance between two lat, long pairs is given below.

```
private static double getDistance(double lat1, double lon1, double lat2, double lon2) {

    final double Radius = 6371 * 1E3; //Earth's mean radius
    double dLat = Math.toRadians(lat2-lat1);
    double dLon = Math.toRadians(lon2-lon1);
    double a = Math.sin(dLat/2) * Math.sin(dLat/2) +
    Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2)) *
            Math.sin(dLon/2) * Math.sin(dLon/2);
    double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
    return Radius * c;

    }
```

6. Three pre-identified locations:

EBII @ Centennial NCSU:	longitude -78.67385	latitude 35.77198
Washington DC:	longitude -77.03687	latitude 38.90719
Paris:	longitude 2.35222	latitude 48.85661

Deliverables

- A README.txt file containing the project member's names, unity id's, section numbers, and any special instructions for your submission. In the README.txt, **you need to note down all the java files modified and new java files added (if any) for your implementation**.
- A zip file containing the project and the APK file (in Eclipse, right click on the project and export as archive file).

Testing and grading

Your application will be tested for correct functionality. Roughly, the following will be the rubric.

- Your application should launch successfully. If it force closes or throws an ANR error, it will not be graded.
- (20 points) Your application sends announcement messages properly. We will join and quit the chat multiple times using different user names to test your program.
- (30 points) When we choose participants menu item, your application shows a list of participants along with either [Y] or [N] indicating the person is or not in your contact list. Further, when we click on some participant with [N] adjacent to his/her name, your application needs to add the person in your contact list. So, when we see participants list again, his/her name should be marked as [Y]. We will add names different from the ones given above to test your program.
- (50 points) We will run two chat agents indicating there are two people there. In between the chats, we'll send a pair of random GPS coordinates from the DDMS perspective for both agents. We will verify if your application shows the chat message appropriately. We will test the pre-identified locations to see if your application shows the correct location information e.g., This is EBII @ Centennial.