MusicPhone

MusicPhone is an application that runs on a GPS-enabled, MP3-capable mobile phone. MusicPhone will make recommendations for artists the user may like and find upcoming concert events for artists using data gathered from the Last.FM website. The goal of the following tasks is to implement the necessary logic to enable MusicPhone to make recommendations and plan a travel itinerary to concert events.

1. You are free to browse the web to look up information.
2. You may *not* add any extra add-ins to Eclipse
3. Use JUnit 4 testing framework for unit tests.
4. Use test-driven development to implement the tasks.

Task A: Ramp-up

A1. Run the project and see three UI windows appear. The Player and GPS UIs are complete, but you will need to bound the buttons to the correct functionality
The Recommender window is just a skeleton. You will implement the required functionality in the `Recommender` class.
Run SmokeTests and check the sample tests to get familiar with the application.

A2. Take 5 minutes to review the information in the file **Documentation.doc**.

Task B: Compute distance to a concert

The user will see how far away upcoming concert events are for a particular artist based on the user's current GPS position and the position of the concert's venue. Implement a public method with the signature `double computeDistance (GeoPoint, GeoPoint, String)` to calculate the great-circle distance between two geo-coded positions. A geo-coded position is represented by a `GeoPoint` object that specifies a longitude and latitude *in degrees*. The method computes the great-circle distance between two points in either kilometres ("km") or miles ("mi") as specified by the third parameter.

Examples:

GeoPoint$_1$		*GeoPoint*$_2$		*units*	**Great-circle distance** (in *units*)
LonR$_1$ LonD$_1$	*LatR*$_1$ LatD$_1$	*LonR*$_2$ LonD$_2$	*LatR*$_2$ LatD$_2$		
0	*0*	*6.283*	*6.283*	km	**0**
0	0	360	360		
0	*0*	*1.047*	*0*	km	**6671.70**
0	0	60	0		
0	*0*	*6.283*	*6.283*	mi	**0**
0	0	360	360		
0	*0*	*1.047*	*0*	mi	**4145.60**
0	0	60	0		
0.630	*-1.513*	*0.592*	*-2.066*	mi	**1793.55**
36.12	-86.67	33.94	-118.40		

Numbers in roman type are in degrees.
Numbers in italics are in radians.

The formula for the great-circle distance is:

LonD = longitude in degrees;
LatD = latitude in degrees;
$LonR$ = longitude in radians;
$LatR$ = latitude in radians;

$$radians = (degrees \times \pi)/180 ; \qquad a = \sqrt{\sin^2\left(\frac{\Delta LatR}{2}\right) + \cos(LatR_1)\times\cos(LatR_2)\times\sin^2\left(\frac{\Delta LonR}{2}\right)} ;$$

$\text{Great Circle Distance} = 2\times\arcsin(\min(1.0, a))\times Radius$; where *Radius* is the Earth's radius in kilometers (6371.01) or in miles (3958.76).,

Task C: Find concerts for an artist

Implement the `getDestinationsForArtist` method of the `Recommender` class. When the user clicks on an artist's name in the list of recommended artists, the UI calls the `getDestinationsForArtist` method to obtain a list of destinations, where each destination contains the info of an upcoming concert together with the distance to the concert's location from the user's current position. The user's current position is provided by the GPS.

Task D: Recommend artists

Implement the `getRecommendations` method of the `Recommender` class. MusicPhone recommends artists based on the favourite artists of Last.FM users who are the top fans of the currently-playing artist on the user's player. MusicPhone recommends the 20 most-frequently-appearing artists among those fans' top artists (as in "people who like this artist also like these other artists…"). `getRecommendations` should return a list of Recommendation objects. Each `Recommendation` object consists of an artist's name (`artist`) and the frequency of occurrence (`fanCount`) of that artist. You will need to use `IConnector's getTopFansForArtist and getTopArtistsByFan` methods to implement this method.

Example: Suppose the currently playing artist is *Cher*. Suppose *John*, *Sally*, and *Tom* are the only top fans of *Cher*. If *Tom* and *John* both like *U2* as well, then `getRecommendations` should return a list that includes a recommendation containing *U2* with a `FanCount` of 2.

Task E: Compute an itinerary for concerts

E0. Implement the `buildItineraryForArtists` method of the `Recommender` class. In the UI, the user can click on buttons that will add or remove a selected artist to a list of artists who the user would like to see in concert. Whenever this list is changed, the application recalculates an itinerary of upcoming concerts by calling this method. An itinerary is a list of `Destinations`, where each destination contains the concert information and the distance from the previous destination. The first destination's distance is the distance from the user's current position. The itinerary must be chonologically ordered according the the start date of the concerts in it. Satisfy the following constraints in any order, but tackling them *one by one:*

E1. The itinerary starts with the concert having the earliest start date for any of the selected artists.
E2. An artist has at most one concert in the itinerary (some artists may have no concerts, however).
E3. No two concerts in the itinerary may take place on the same day.
E4. If multiple artists have concerts on the same day,the itinerary includes only the concert that is closest in distance to the previous destination.
E5. If the same artist has multiple concerts on the same day, the itinerary inlcudes only the concert that is cloest in distance to the previous destination.