PLAIN HTML VERSION: lecture10.161.txt.html
Xcode Project Archive: lecture10.161.zip

```
/*
file:    lecture10.161.swift
author:  Danny Abesdris
date:    April 6, 2016
purpose: MAP523AA/DPS923AA lecture #10

         Swift programming language topics:
         Introduction to MapKit / Core Location
*/

/*
Core Location Framework in Swift:

iOS devices are able to employ a number of different techniques for obtaining informat
about the current geographical location of the device. These mechanisms include GPS, c
tower triangulation and finally (and least accurately), by using the IP address of
available Wi-Fi connections. The mechanism that is used by iOS to detect location
information is, however, largely transparent to the application developer and the syst
will automatically use the most accurate solution available at any given time. In fact
that is needed to integrate location based information into an iOS application is an
understanding of how to use the Core Location Framework.

The Core Location Manager:
The key classes contained within the Core Location Framework are CLLocationManager and
CLLocation. An instance of the CLLocationManager class can be created using the follow
Swift code:
*/

var locationManager: CLLocationManager = CLLocationManager( )

/*
Once a location manager instance has been created, it must seek permission from the us
to collect location information before it can begin to track data.

Requesting Location Access Authorization:
Before any application can begin to track location data it must first seek permission
do so from the user. This can be achieved by making a call to one of two methods on th
CLLocationManager instance depending on the specific requirement. If the application o
needs to track location information when the application is in the foreground then a
call should be made to the requestWhenInUseAuthorization method of the location manage
instance. For example:
*/

locationManager.requestWhenInUseAuthorization( )

// In the event that tracking is also required when the application is running in the
// background, the requestAlwaysAuthorization method should be used instead:

locationManager.requestAlwaysAuthorization( )

/*
```

```
Each method call requires that a specific key-value pair be added to the Information
Property List dictionary contained within the applications Info.plist file. The
value takes the form of a string and must describe the reason why the application needs
access to the users current location.
The keys associated with these values are as follows:

NSLocationWhenInUseUsageDescription  - A string value describing to the user why the
                                       application needs access to the current location
                                       when running in the foreground.
NSLocationAlwaysUsageDescription     - A string describing to the user why the applicat
                                       needs access to the current location data all th
                                       time that the application is running
                                       (including when it is in the background).

Configuring the Desired Location Accuracy:
The level of accuracy to which location information is to be tracked is specified via
the desiredAccuracy property of the CLLocationManager object. It is important to keep
in mind when configuring this property that the greater the level of accuracy selected
the greater the drain on the device battery. An application should, therefore, never
request a greater level of accuracy than is actually needed.

A number of predefined constant values are available for use when configuring this prop

kCLLocationAccuracyBestForNavigation  - Uses the highest possible level of accuracy
                                        augmented by additional sensor data. This accur
                                        level is intended solely for use when the devic
                                        connected to an external power supply.
kCLLocationAccuracyBest               - The highest recommended level of accuracy for d
                                        running on battery power.
kCLLocationAccuracyNearestTenMeters   - Accurate to within 10 meters.
kCLLocationAccuracyHundredMeters      - Accurate to within 100 meters.
kCLLocationAccuracyKilometer          - Accurate to within one kilometer.
kCLLocationAccuracyThreeKilometers    - Accurate to within three kilometers.
*/

// The following code, for example, sets the level of accuracy for a location manager
// instance to best accuracy:

locationManager.desiredAccuracy = kCLLocationAccuracyBest

/*
Configuring the Distance Filter:
The default configuration for the location manager is to report updates whenever any
changes are detected in the location of the device. The distanceFilter property of the
location manager allows applications to specify the amount of distance the device
location must change before an update is triggered. If, for example, the distance filte
is set to 1000 meters the application will only receive a location update when the
device travels 1000 meters or more from the location of the last update. For example,
to specify a distance filter of 1500 meters:
*/
locationManager.distanceFilter = 1500.0

// The distance filter may be cancelled, thereby returning to the default setting, usin
// the kCLDistanceFilterNone constant:

locationManager.distanceFilter = kCLDistanceFilterNone

/*
The Location Manager Delegate:
Location manager updates and errors result in calls to two delegate methods defined wit
the CLLocationManagerDelegate protocol. Templates for the two delegate methods that mus
```

```
be implemented to comply with this protocol are as follows:
*/


func locationManager(manager: CLLocationManager!, didUpdateLocations locations: [AnyOb
   // Handle location updates here
}

func locationManager(manager: CLLocationManager!, didFailWithError error: NSError!)  {
   // Handle errors here
}


/*
Each time the location changes, the didUpdateLocations delegate method is called and
passed as an argument an array of CLLocation objects with the last object in the array
containing the most recent location data.

Changes to the location tracking authorization status of an application are reported
via a call to the optional didChangeAuthorizationStatus delegate method:
*/


func locationManager(manager: CLLocationManager!,
    didChangeAuthorizationStatus status: CLAuthorizationStatus) {

        // App may no longer be authorized to obtain location
        //information. Check status here and respond accordingly.

}

// Once a class has been configured to act as the delegate for the location manager, tl
// object must be assigned to the location manager instance. In most cases, the delegat
// will be the same view controller class in which the location manager resides, for ex

locationManager.delegate = self

/*
Starting Location Updates:
Once suitably configured and authorized, the location manager can then be instructed to
start tracking location information:
*/

locationManager.startUpdatingLocation( )

/*
With each location update, the didUpdateToLocation delegate method is called by the loc
manager and passed information about the current location.

Obtaining Location Information from CLLocation Objects:
Location information is passed through to the didUpdateLocation delegate method in the
CLLocation objects. A CLLocation object encapsulates the following data:

Latitude
Longitude
Horizontal Accuracy
Altitude
Altitude Accuracy
Longitude and Latitude

Longitude and latitude values are stored as type CLLocationDegrees and may be obtained
```

```
from a CLLocation object as follows:
*/


let currentLatitude: CLLocationDistance = location.coordinate.latitude

let currentLongitude: CLLocationDistance = location.coordinate.longitude


// Accuracy:
// Horizontal and vertical accuracy are stored in meters as CLLocationAccuracy values a
// may be accessed as follows:


let verticalAccuracy: CLLocationAccuracy = location.verticalAccuracy

let horizontalAccuracy: CLLocationAccuracy = location.horizontalAccuracy



// Altitude:
// The altitude value is stored in meters as a type CLLocationDistance value and may be
// from a CLLocation object as follows:

let altitude: CLLocationDistance = location.altitude

/*
Calculating Distances:
The distance between two CLLocation points may be calculated by calling the distanceFro
method of the end location and passing through the start location as an argument. For e
the following code calculates the distance between the points specified by startLocatio
endLocation:
*/

var distance: CLLocationDistance = endLocation.distanceFromLocation(startLocation)

/*
Location Information and Multitasking:
Location based iOS applications are one of the three categories of application that are
to continue executing when placed into the background (for a detailed description of mu
refer to iOS Multitasking, Background Transfer Service and Fetching). If location updat
required when the application is in the background state it is strongly recommended tha
desired accuracy setting be reduced within the applicationDidEnterBackground method by
a call to the startMonitoringSignificantLocationChanges method of the location manager
This will ensure that the application is only notified of significant changes to the lo
of the device thereby reducing the load on the battery.
*/
```