MODULE 1

Q1 a) what is mobile applications? Explain the importance of mobile application in today's scenario. (8 marks)

A mobile application, most commonly referred to as an app, is a type of application software designed to run on a mobile device, such as a smartphone or tablet computer.

As it stands, there are really four major development targets. Each of the native frameworks comes with certain expectations and a user base. BlackBerry is often used in education and government, whereas the iPhone and Android user base is far more widespread. Windows Phone 7 being the newcomer is used primarily by developers and hasn't necessarily hit its stride yet.

iOS, the technology that is run on Apple mobile devices, has benefits and limitations specific to its development cycle. The base language is Objective-C, with Cocoa Touch as the interface layer. At this time iOS can be developed only using Apple's XCode, which can run only on a Macintosh.

The Android framework, on the other hand, is written in Java, and can be developed using any Java tools. The specific tooling recommended by Google and the Android community is Eclipse with the Android toolkit. Unlike iOS, it can be developed on PC, Mac, or Linux. Like Android, the BlackBerry device framework is also written in Java; however, it is limited in that the Emulator and Distribution tools run only on Windows at this time.

The newest native framework on the market is Windows Phone 7 and its framework sits on top of the Microsoft's .NET Framework. The language of choice is C# and the framework lies in a subset of Silverlight, Microsoft's multiplatform web technology. It also has the limitation that the Microsoft Windows Phone tools run only on Windows.

Q1 b) what are the development process of mobile application in business world (8 marks)

• If potential customers cannot reach your services, they are lost potential customers. Smartphones, tablets, and other nontraditional devices are pervasive in the market.

• The onus of responsibility is on developers to help customers get a product anywhere.

• Whether you're a content provider, Product Company, or service company, expanding product reach is necessary.

• And one of the most effective ways to reach farther is to simplify a message so that it can be delivered to a wider audience.

• As of September 2011, Nielsen reports that 40 percent of all mobile consumers in the United States over the age of 18 have smartphones: http://blog.nielsen.com/nielsenwire/online_mobile/40- percent-of-u-s-mobile-users-ownsmartphones-40-percent-are-android/.

• Wired states as of November 2011 that global smartphone usage has reached 30 percent: www.wired .com/gadgetlab/2011/11/smartphones-feature-phones/.

Q2 a) What do you understand by mobile information design and mobile platform? Explain it.

1 **Android**

- Android has a diverse ecosystem, with fewer institutionalized restrictions and a wider variety of mobile devices than other popular systems.
- They are strong competitor in the mobile market, but the flexibility of Android design can introduce new issues.
- Development of the Android operating system is led by Google http://developer.android.com/guide/practices/ui_guidelines/index.html

Interface Tips

– Get started on Android application design with these hints.

- Android convention is to place view-control tabs across the top, and not the bottom, of the screen.
- Use the main application icon for temporal, hierarchical navigation, instead of a "back" button and main icon link to the home screen.
- Don't mimic user interface elements or recycle icons from other platforms.
- For instance, list items should not use carets to indicate deeper content.
- Parallax scrolling is common in Android applications.
- Android development can extend to home-screen "widget" tools.

Accessibility

- Google provides guidelines and recommendations, such as testing with the often-preinstalled and always-free Talkback. Accessibility design guidelines are listed on the Android Developer website (http://developer.android.com/guide/topics/ui/accessibility/index.html), and further discussed by the Google "Eyes Free" project (http://eyesfree.googlecode.com/svn/trunk/documentation/android_access/index.html).

2 iOS

- Apple maintains strict design standards, which are detailed and updated online. iOS
- interface documentation and general mobile design strategies are available from Apple, including design strategies and case studies, at http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html

Interface Tips

- -Apple can reject an application from the official App Store because of design problems. Follow the current guidelines closely, starting with these tips:
- iPhone users generally hold from the bottom of the device, so main navigation items should be in reach of user thumbs.
- Target areas for controls should be a minimum of 44 x 44 points.
- Support standard iOS gestures, such as swiping down from the top to reveal the Notification Center.
- Larger iPad screens are great for custom multi-finger gestures, but nonstandard gestures should never be the only way to reach and use important features.

Accessibility

- Accessible touch and gestural controls are available on the iPad and later generation iPhones;
- Screen magnification and colour contrast adjustments are also available.

3 BlackBerry OS

- BlackBerry OS is often the mobile device of choice in or corporate environments.
- BlackBerry includes native support of corporate emails; and runs on many devices with hard keypads. Which is favoured by users with accessibility issues as well as late adopters to touch-screen interfaces.

Interface Tips

- Link common tasks to the BlackBerry track pad according to standard actions:

- Press the track pad: Default option, like revealing the menu
- Press and hold track pad: Activate available pop-up box
- Press track pad and select Shift: Highlight content
- Press track pad and select Alt: Zoom
- Move finger along track pad: Cursor or mouse will move accordingly

Accessibility

- BlackBerry mobile devices include text-based push-delivery messages, closed captions on multimedia content, and hearing-aid compatibility for hearing accessibility issues. Low-vision users can use the Clarity theme and other screen adjustments, and benefit from tactile keyboards. Predictive text and AutoText aid users with mobility and cognitive issues. Best practices and device capabilities are maintained online at http://docs.blackberry.com/en/

4 Windows Phone7

- Developed by Microsoft, Windows Phone 7 (WP7) is a currently smaller contender, focused on consumer markets.
- Using the "Metro" theme, features are divided into "Live Tiles" that link to applications.
- Six dedicated hardware buttons (back, start, search, camera, power, and volume), at least 4 GB of Flash memory, and Assisted GPS.

Interface Tips

- Windows Phone 7 interfaces are minimalist,
- Using empty space to lend clarity to the applicationWP7 uses movement over gradients for on-screen elements to immerse users in the application experience. Users will enter a WP7 application from a "tile," which can display dynamic and real-time information.
- Tile images should be in the PNG format, 173 pixels 173 pixels at 256dpi.eaving focus. Do not use a "back" button to navigate back the page stack. Panorama controls slide horizontally through panes, and pivot controls list panes users can visit.
- Uniform Page Shuffle presents non-hierarchical information users can shuffle through;
   "leaf-blowing turn" flips content area into focus,
   Scatteringand tilting tiles leaving focus.

Accessibility

- WP7 devices include many standard accessibility features, such as color and
- Contrast adjustment to themes for low-vision users. Many, but not all, devices
- are compatible with TTY, TDD, and hearing aids.
- Learn more about the basics of WP7 accessibility at
- http://www.microsoft.com/

5 Mobile Web Browsers

- If a mobile application sends users to a website, that website should be optimized for mobile browsers.
- Similarly, mobile web applications should follow key mobile design methods.
- A great resource for design best practices for mobile web browsers is published by the W3C.

Interface Tips

Few quick tips to get started:

- Test for a consistent experience: when websites are accessed from a variety of mobile browsers.
- Provide minimal navigation: at the top of the page, and use consistent navigation mechanisms.
- Do not change or refresh the current window: or cause pop-ups, without informing the user and providing the means to stop it.
- Limit content: what the user has requested, and what the user's device can display by avoiding large image files.
- Specify default input formats : when possible, provide preselected defaults

Accessibility

- The W3C Web Accessibility Initiative provides introductions, solutions, and further resources to create accessible mobile websites and mobile web applications

Q2 b) Explain the effective use of screen real estate and features of mobile application (8 marks)

The first step to use the smaller interfaces of mobile devices effectively is to know the context of use. Who are the users, what do they need and why, and how, when, and where will they access and use information?

Mobile design is difficult, as developers try to elegantly display a telescoped view of almost limitless information. But user experience issues are amplified on mobile interfaces.

Cognitive load increases while attention is diverted by the needs to navigate, remember what was seen, and re-find original context.

Cognitive load is the mental effort to comprehend and use an application, whatever the inherent task complexity or information structure may be.

Effectively use screen real estate by embracing minimalism, maintaining a clear visual hierarchy, and staying focused.

Embrace Minimalism

• Limit the features available on each screen, and use small, targeted design features.

• Content on the screen can have a secondary use within an application, but the application designer should be able to explain why that feature is taking up screen space.

• Banners, graphics, and bars should all have a purpose.

Use a Visual Hierarchy

• Help users fight cognitive distractions with a clear information hierarchy.

• Draw attention to the most important content with visual emphasis.

• Users will be drawn to larger items, more intense colors, or elements that are called out with bullets or arrows; people tend to scan more quickly through lighter color contrast, less intense shades, smaller items, and text-heavy paragraphs.

• A consistent hierarchy means consistent usability; mobile application creators can create a hierarchy with position, form, size, shape, color, and contrast.

Stay Focused

• Start with a focused strategy, and keep making decisions to stay focused throughout development.

• A smaller file size is a good indicator of how fast an application will load, so the benefits of fighting feature creep extend beyond in-application user experience.

• Focused content means users won't leave because it takes too long for the overwhelming amount of images per screen to load.

• And users won't be frustrated with the number of links that must be cycled through to complete a task. Text-heavy pages reduce engagement as eyes glaze over and users switch to another application.

• If people have taken the time to install and open an application, there is a need these users hope to meet.

• Be methodical about cutting back to user necessities. Build just enough for what users need, and knowing what users need comes from understanding users

As it stands, there are really four major development targets. Each of the native frameworks comes with certain expectations and a user base. BlackBerry is often used in education and government, whereas the iPhone and Android user base is far more widespread. Windows Phone 7 being the newcomer is used primarily by developers and hasn't necessarily hit its stride yet.

iOS, the technology that is run on Apple mobile devices, has benefits and limitations specific to its development cycle. The base language is Objective-C, with Cocoa Touch as the interface layer. At this time iOS can be developed only using Apple's XCode, which can run only on a Macintosh.

The Android framework, on the other hand, is written in Java, and can be developed using any Java tools. The specific tooling recommended by Google and the Android community is Eclipse with the Android toolkit. Unlike iOS, it can be developed on PC, Mac, or Linux. Like Android, the BlackBerry device framework is also written in Java; however, it is limited in that the Emulator and Distribution tools run only on Windows at this time.

The newest native framework on the market is Windows Phone 7 and its framework sits on top of the Microsoft's .NET Framework. The language of choice is C# and the framework lies in a subset of Silverlight, Microsoft's multiplatform web technology. It also has the limitation that the Microsoft Windows Phone tools run only on Windows.

MODULE 2

Q3 a) Explain the fundamentals of developing android application.

A developer may go for mobile app development because of following reasons − Your competitors have mobile apps, but you don't. So, to improve the business and to sustain in the market, one may like to go for having a mobile-app. Mobile apps make good business sense. One can give good quality service in a faster time frame. Your services would add value to a user's mobile experience but your website isn't mobile friendly. Do you need a mobile application or a mobile website?

COST OF DEVELOPMENT

Mobile app development costs related to hardware and software. The developer team needs devices to test the software on. And if you want to deploy your application to any public market, then your company will need accounts on the various markets (these often renew annually).

Hardware

To develop good mobile apps:

a. Need an intel based Mac:

- Can run Windows on them either virtually or on the bare metal.
- Expect to spend between $800 to $1600

b. Need multiple monitors

- When debugging any application requires 3 machines
- Emulator/simulator
- My Dev Tool (IDE)
- Web browser
- Having access to all of this information at once prevents context switching for a developer.

c. Examples of devices you can use to test the various platforms instead of emulator:

- Android 2.2 (Froyo): Motorola Droid 2
- Android 3.0 Tablet: Samsung Galaxy Tablet
- Apple iPod Touch: iPod Touch 3rd Generation
- Apple iPhone (versions 3.x and 4.x) (cell service): iPhone 3GS
- Apple iPhone (versions 4 and greater) (cell service): iPhone 4
- Apple iPad (WiFi or 3G for cell service testing): iPad 1
- Apple iPad (with camera): iPad 2 or iPad 3
- Windows Phone 7: Samsung Focus

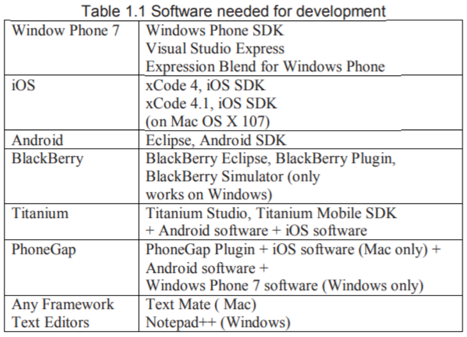
Software

When developing mobile applications, there are few overlaps when it comes to software.

To develop for iOS you need a Mac. To develop for BlackBerry you need Windows. For Java-based frameworks use Eclipse. Building HTML for PhoneGap can be done in your

text editor of choice. Table 1.1 shows software needed for development.

Table 1.1 Software needed for development



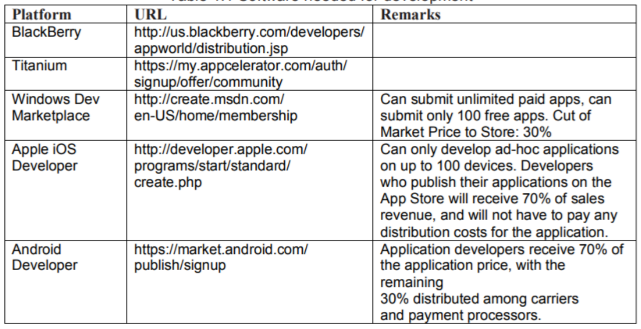
Window Phone 7	Windows Phone SDK Visual Studio Express Expression Blend for Windows Phone
iOS	xCode 4, iOS SDK xCode 4.1, iOS SDK (on Mac OS X 107)
Android	Eclipse, Android SDK
BlackBerry	BlackBerry Eclipse, BlackBerry Plugin, BlackBerry Simulator (only works on Windows)
Titanium	Titanium Studio, Titanium Mobile SDK + Android software + iOS software
PhoneGap	PhoneGap Plugin + iOS software (Mac only) + Android software + Windows Phone 7 software (Windows only)
Any Framework Text Editors	Text Mate (Mac) Notepad++ (Windows)

Licenses and Developer Accounts

Table 1.2 shows the information regarding various accounts necessary to develop mobile

app. One has to pay some amount for developer accounts per year.

Table 1.1 Software needed for development



Platform	URL	Remarks
BlackBerry	http://us.blackberry.com/developers/ appworld/distribution.jsp	
Titanium	https://my.appcelerator.com/auth/ signup/offer/community	
Windows Dev Marketplace	http://create.msdn.com/ en-US/home/membership	Can submit unlimited paid apps, can submit only 100 free apps. Cut of Market Price to Store: 30%
Apple iOS Developer	http://developer.apple.com/ programs/start/standard/ create.php	Can only develop ad-hoc applications on up to 100 devices. Developers who publish their applications on the App Store will receive 70% of sales revenue, and will not have to pay any distribution costs for the application.
Android Developer	https://market.android.com/ publish/signup	Application developers receive 70% of the application price, with the remaining 30% distributed among carriers and payment processors.

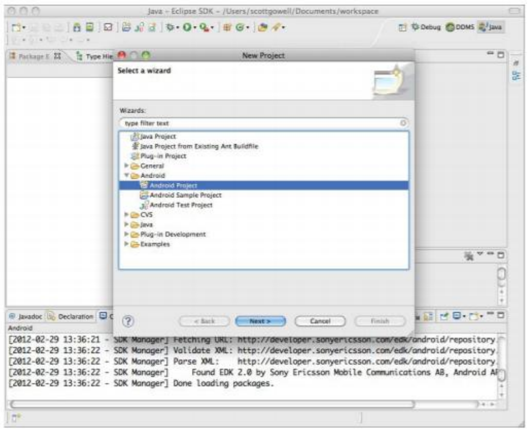
Documentation and APIs

Following are links to the respective technologies' online documentation and APIs. This will be the location for the latest information in the respective technology

- MSDN Library: http://msdn.microsoft.com/enus/library/ff402535(v=vs.92).aspx
- iOS Documentation: http://developer.apple.com/devcenter/ios/index.action
- BlackBerry Documentation: http://docs.blackberry.com/en/developers/?userType=21
- Android SDK Documentation: http://developer.android.com/reference/packages.html
- and http://developer.android.com/guide/index.html
- PhoneGap Documentation: http://docs.phonegap.com/
- Titanium API Documentation: http://developer.appcelerator.com/apidoc/mobile/latest

Q3 b) Explain how to build derby app in android.

Creating a New Project First things first — you need to create a new Android project. The line highlighted in Figure 6-7 is the type of project you want.

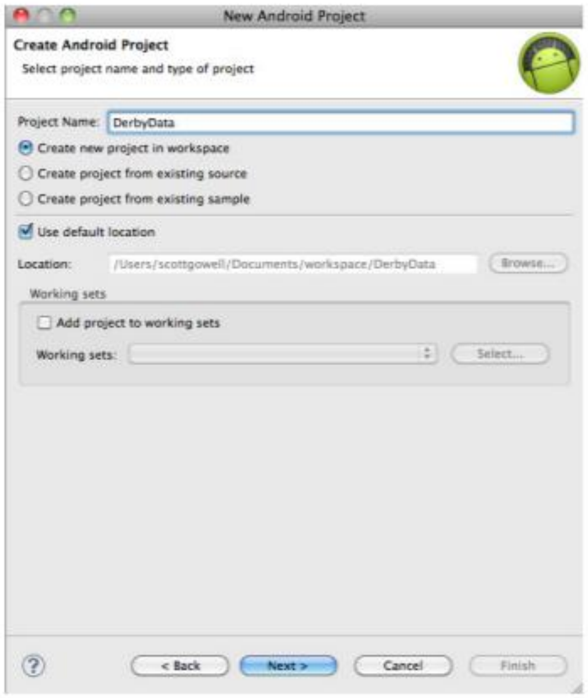


```
Android
[2012-02-29 13:36:21 - SDK Manager] Fetching URL: http://developer.sonyericsson.com/edk/android/repository.
[2012-02-29 13:36:22 - SDK Manager] Validate XML: http://developer.sonyericsson.com/edk/android/repository.
[2012-02-29 13:36:22 - SDK Manager] Parse XML:    http://developer.sonyericsson.com/edk/android/repository.
[2012-02-29 13:36:22 - SDK Manager]     Found EDK 2.0 by Sony Ericsson Mobile Communications AB, Android AP
[2012-02-29 13:36:22 - SDK Manager] Done loading packages.
```

FIGURE 6-7 Creating a new Android project

First you need to name your application and add it to a workspace. Think of a workspace as the folder in which your application resides. Figure 6-8 illustrates what the New Android Project screen looks like. After you have named your application you will need to give it a package name, set the minimum SDK required to run your application, and name the initial Activity that will run when your application runs. If you want to add a test project to your workspace, you can do so at this time. Figure 6-9 shows a completed Application Info step in the new project wizard. An important note at this point: Make sure that your package name is unique. The standard format for package names is com.companyName.applicationName. This must be unique because that is how it is known on the Android device and in the Android Market. When you make updates you can make them only within your package name. If you change your package name there will be no upgrade path between versions.

The minimum SDK required is generally set when you are leveraging a permission or piece of functionality that did not exist when the core Android version was released, or if you want to target a specific type of device (tablet versus handset). The major jumps are between 1.6 and 2.1, 2.3 and 3.x, and 3.x and 4.x. Figure 6-10 shows you all of the SDKs that you have installed that you can target when creating your application. Please note that the reason this screen is full of SDKs is because I took the time to download them all for demonstration purposes.



New Android Project

Create Android Project

Select project name and type of project

Project Name: DerbyData

[✓ Create new project in workspace]

Create project from existing source

Create project from existing sample

[✓] Use default location

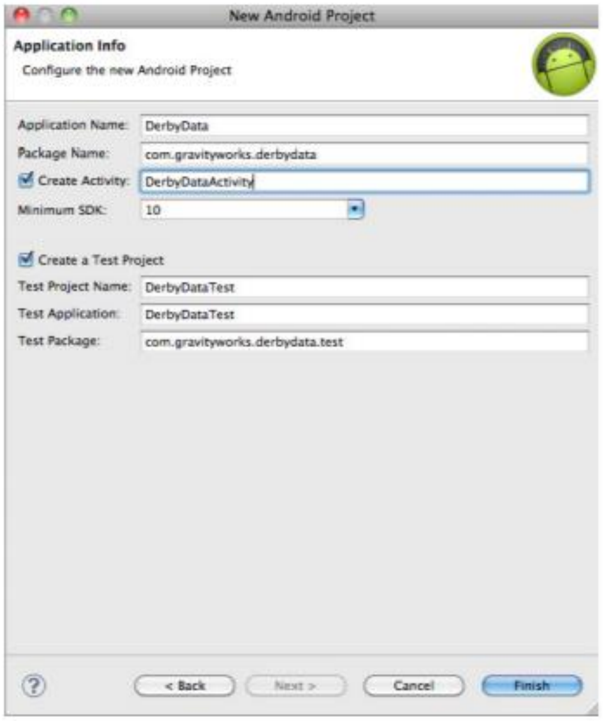
Location: /Users/scottgowell/Documents/workspace/DerbyData

Working sets

[ ] Add project to working sets

Working sets:

FIGURE 6-8: Naming your project



New Android Project

Application Info

Configure the new Android Project

Application Name: DerbyData

Package Name: com.gravityworks.derbydata

[✓] Create Activity: DerbyDataActivity

Minimum SDK: 10

[✓] Create a Test Project

Test Project Name: DerbyDataTest

Test Application: DerbyDataTest

Test Package: com.gravityworks.derbydata.test

FIGURE 6-9: Configuring application information

This step is also very important when building your application. The minimum SDK version you set specifies the lowest possible version of the SDK in which your application will run, and it is the primary version in which your application will run. Android 1.5 is the lowest version of the SDK still supported, and Android 4.0.3 (at the time of this writing) is the highest.

Q4 a) What are the components of android applications? Explain with example.

There are four different types of app components:

1. Activities
2. Services
3. Broadcast receivers
4. Content providers

Activities

An activity is the entry point for interacting with the user. It represents a single screen with a user interface. For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. Although the activities work together to form a cohesive user experience in the email app, each one is independent of the others. As such, a different app can start any one of these activities if the email app allows it. For example, a camera app can start the activity in the email app that composes new mail to allow the user to share a picture. An activity facilitates the following key interactions between system and app:

Keeping track of what the user currently cares about (what is on screen) to ensure that the system keeps running the process that is hosting the activity.

Knowing that previously used processes contain things the user may return to (stopped activities), and thus more highly prioritize keeping those processes around.

Helping the app handle having its process killed so the user can return to activities with their previous state restored. Providing a way for apps to implement user flows between each other, and for the system to coordinate these flows. (The most classic example here being share.)

You implement an activity as a subclass of the Activity class. For more information about the Activity class, see the Activities developer guide.

Services

A service is a general-purpose entry point for keeping an app running in the background for all kinds of reasons. It is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. For example, a service might play music in the background while the user is in a different app, or it might fetch data over the network without blocking user interaction with an activity. Another component, such as an activity, can start the service and let it run or bind to it in order to interact with it. There are actually two very distinct semantics services tell the system about how to manage an app: Started services tell the system to keep them running until their work is completed. This could be to sync some data in the background or play music even after the user leaves the app. Syncing data in the background or playing music also represent two different types of started services that modify how the system handles them:

Music playback is something the user is directly aware of, so the app tells the system this by saying it wants to be foreground with a notification to tell the user about it; in this case the system knows that it should try really hard to keep that service's process running, because the user will be unhappy if it goes away.

A regular background service is not something the user is directly aware as running, so the system has more freedom in managing its process. It may allow it to be killed (and then restarting the service sometime later) if it needs RAM for things that are of more immediate concern to the user.

Bound services run because some other app (or the system) has said that it wants to make use of the service. This is basically the service providing an API to another process. The system thus knows there is a dependency between these processes, so if process A is bound to a service in process B, it knows that it needs to keep process B (and its service) running for A. Further, if process A is something the user cares about, then it also knows to treat process B as something the user also cares about. Because of their flexibility (for better or worse), services have turned out to be a really useful building block for all kinds of higher-level system concepts. Live wallpapers, notification listeners, screen savers, input methods, accessibility services, and many other core system features are all built as services that applications implement and the system binds to when they should be running.

A service is implemented as a subclass of Service. For more information about the Service class, see the Services developer guide.

Broadcast receivers

A broadcast receiver is a component that enables the system to deliver events to the app outside of a regular user flow, allowing the app to respond to system-wide broadcast announcements. Because broadcast receivers are another well-defined entry into the app, the system can deliver broadcasts even to apps that aren't currently running. So, for example, an app can schedule an alarm to post a notification to tell the user about an upcoming event... and by delivering that alarm to a BroadcastReceiver of the app, there is no need for the app to remain running until the alarm goes off. Many broadcasts originate from the system—for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured. Apps can also initiate broadcasts—for example, to let other apps know that some data has been downloaded to the device and is available for them to use. Although broadcast receivers don't display a user interface, they may create a status bar notification to alert the user when a broadcast event occurs. More commonly, though, a broadcast receiver is just a gateway to other components and is intended to do a very minimal amount of work. For instance, it might schedule a JobService to perform some work based on the event with JobScheduler

A broadcast receiver is implemented as a subclass of BroadcastReceiver and each broadcast is delivered as an Intent object. For more information, see the BroadcastReceiver class.

Content providers

A content provider manages a shared set of app data that you can store in the file system, in a SQLite database, on the web, or on any other persistent storage location that your app can access. Through the content provider, other apps can query or modify the data if the content provider allows it. For example, the Android system provides a content provider that manages the user's contact information. As such, any app with the proper permissions can query the content provider, such as ContactsContract.Data, to read and write information about a particular person

Q4 b) Discuss the components involved in android for activities and fragments

The lifecycle events of a Fragment reflect those of its parent Activity. But, when the container Activity is in its active and resumed state by adding or removing a fragment, it will affect the lifecycle independently. Figure 2.3 shows various events in lifecycle of fragments.

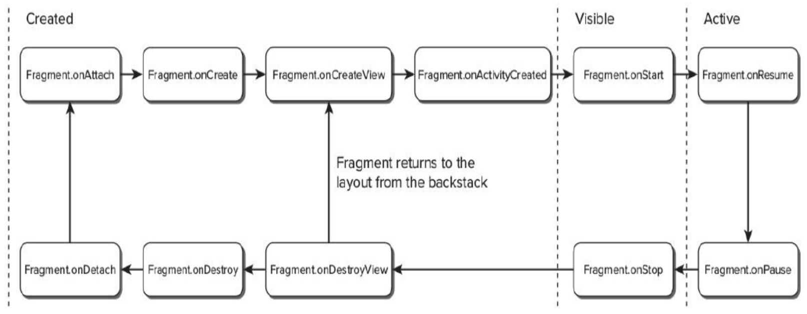


Figure 2.3 Lifecycle of Fragments

Most of the Fragment lifecycle events correspond to their equivalents in the Activity class. Those that remain are specific to Fragments and the way in which they're inserted into their parent Activity are explained here:

 Attaching and Detaching Fragments from the Parent Activity: The full lifetime of your Fragment begins when it is bound to its parent Activity and ends when it has been detached. These events are represented by the calls to onAttach and onDetach, respectively. As with any handler called after a Fragment/Activity has become paused, it's possible that onDetach will not be called if the parent Activity's process is terminated without completing its full lifecycle. The onAttach event is triggered before the Fragment's UI has been created, before the Fragment itself or its parent Activity have finished their initialization. Typically, the onAttach event is used to gain a reference to the parent Activity in preparation for further initialization tasks.

 Creating and Destroying Fragments: The created lifetime of your Fragmen occurs between the first call to onCreate and the final call to onDestroy. As it's not uncommon for an Activity's process to be terminated without the corresponding
onDestroy method being called, so a Fragment can't rely on its onDestroy handler being triggered. As with Activities, you should use the onCreate method to initialize your Fragment. It's good practice to create any class scoped objects here to ensure they're created only once in the Fragment's lifetime.

 Creating and Destroying User Interfaces: A Fragment's UI is initialized (and destroyed) within a new set of event handlers: onCreateView and onDestroyView, respectively. Use the onCreateView method to initialize your Fragment:
o Inflate the UI,
o get references (and bind data to) the Views it contains,
o and then create any required Services and Timers.
Once you have inflated your View hierarchy, it should be returned from this handler: return inflater.inflate(R.layout.my_fragment, container, false);
If your Fragment needs to interact with the UI of its parent Activity, wait until the onActivityCreated event has been triggered. This signifies that the containing Activity has completed its initialization and its UI has been fully constructed.

MODULE 3

Q5 a)Explain in detail how to design user interface using views

A view is a widget that has an appearance on screen. It is **derived from** the base class android.view.View. There are three types viz.

- Basic views
- Picker views

- List views

Each of these are discussed in the following sections.

Basic Views

Some of the basic views that can be used to design UI of Android application are:

- TextView
- EditText
- Button
- ImageButton
- CheckBox
- ToggleButton
- RadioButton
- RadioGroup

These basic views enable you to display text information, as well as perform some basic selection.

 TextView View: The TextView view is used to display text to the user. When we create a new Android project, Eclipse always creates one <TextView> element in activity_main.xml file, to display Hello World as shown below –

```
<?xml version=”1.0” encoding=”utf-8”?>
<LinearLayout xmlns:android=”http://schemas.android.com/apk/res/android”
android:orientation=”vertical”
android:layout_width=”fill_parent”
android:layout_height=”fill_parent”
>
<TextView
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”@string/hello”
/>
</LinearLayout>
```

Button — Represents a push-button widget

ImageButton — Similar to the Button view, but it also displays an image

EditText — A subclass of the TextView view, but it allows users to edit its text content

CheckBox — A special type of button that has two states: checked or unchecked

RadioGroup and **RadioButton** — The RadioButton has two states: either checked or unchecked. Once a RadioButton is checked, it cannot be unchecked. A RadioGroup is used to group together one or more RadioButton views, thereby allowing only one RadioButton to be checked within the RadioGroup.

ToggleButton — Displays checked/unchecked states using a light indicator

To understand the behavior of these views, create a new android project and place the following code in activity_main.xml file, without disturbing existing code.

```
<Button android:id=”@+id/btnSave”
android:layout_width=”fill_parent”
```

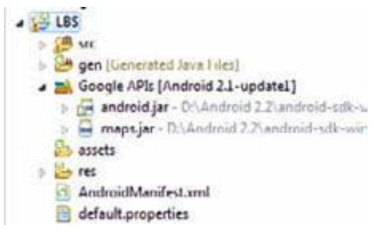
```
android:layout_height=”wrap_content”
android:text=”Save” />
<Button android:id=”@+id/btnOpen”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”
android:text=”Open” />
<ImageButton android:id=”@+id/btnImg1”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:src=”@drawable/icon” />
<EditText android:id=”@+id/txtName”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content” />
<CheckBox android:id=”@+id/chkAutosave”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Autosave” />
<CheckBox android:id=”@+id/star”
style=”?android:attr/starStyle”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content” />
<RadioGroup android:id=”@+id/rdbGp1”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:orientation=”vertical” >
<RadioButton android:id=”@+id/rdb1”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Option 1” />
<RadioButton android:id=”@+id/rdb2”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Option 2” />
</RadioGroup>

<ToggleButton android:id=”@+id/toggle1”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content” />
```

After running the application, the output will be displayed as shown in the following diagram. Click on each of these views, to observe their default behavior.

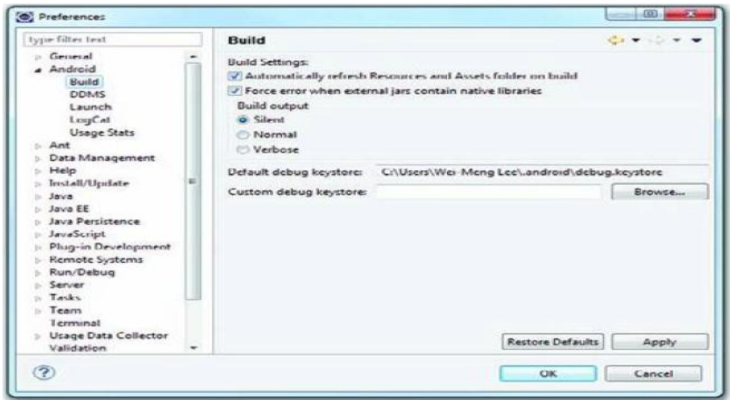
Q5 b) write in details how to display Google maps in your own android applications

Google Maps is one of the many applications bundled with the Android platform. In addition to simply using the Maps application, you can also embed it into your own applications and make it do some very cool things. This section describes how to use Google Maps in your Android applications and programmatically perform the following:
 Change the views of Google Maps.
 Obtain the latitude and longitude of locations in Google Maps.
 Perform geocoding and reverse geocoding (translating an address to latitude and longitude and vice versa).
 Add markers to Google Maps.
We will discuss how to build a project using maps.

Creating the Project: Create a new android project. In order to use Google Maps in your Android application, you need to ensure that you check the Google APIs as your build target. Google Maps is not part of the standard Android SDK, so you need to find it in the Google APIs add-on. If LBS is the name of your project, then you can see the additional JAR file (maps.jar) located under the Google APIs folder as below–


LBS
src
gen [Generated Java Files]
Google APIs [Android 2.1-update1]
android.jar - D:\Android 2.2\android-sdk-
maps.jar - D:\Android 2.2\android-sdk-win
assets
res
AndroidManifest.xml
default.properties

Obtaining the Maps API Key: Beginning with the Android SDK release v1.0, you need to apply for a free Google Maps API key before you can integrate Google Maps into your Android application. When you apply for the key, you must also agree to Google's terms of use, so be sure to read them carefully.



Build

Build Settings:

[✓] Automatically refresh Resources and Assets folder on build

[✓] Force error when external jars contain native libraries

Build output

[✓ Silent] Normal Verbose

Default debug keystore: C:\Users\Wei-Meng Lee\.android\debug.keystore

Custom debug keystore:

First, if you are testing the application on the Android Emulator or an Android device directly connected to your development machine, locate the SDK debug certificate located in the default folder (C:\Users\<username>\.android for Windows 7 users). You can verify the existence of the debug certificate by going to Eclipse and selecting Window ➪Preferences. Expand the Android item and select Build (as shown in figure above). On the right side of the window, you will be able to see the debug certificate's location. The filename of the debug keystore is debug.keystore. This is the certificate that Eclipse uses to sign your application so that it may be run on the Android Emulator or devices.

Q6 a) what is the prospective of deploying APK file in android applications?

Once you have signed your APK fi les, you need a way to get them onto your users' devices. Three methods are here: Deploying manually using the adb.exe tool

- Hosting the application on a web server
- Publishing through the Android Market
- Besides the above methods, you can install your applications on users' devices through emails, SD card, etc. As long as you can transfer the APK file onto the user's device, you can install the application. Using the adb.exe Tool: Once your Android application is signed, you can deploy it to emulators and devices using the adb.exe (Android Debug Bridge) tool (located in the platform-tools folder of the Android SDK). Using the command prompt in Windows, navigate to the "\platform-tools" folder. To install the application to an emulator/device (assuming the emulator is currently up and running or a device is currently connected), issue the following command: adb install "C:\Users\Wei-Meng Lee\Desktop\LBS.apk" (Note that, here, LBS is name of the project) Besides using the adb.exe tool to install applications, you can also use it to remove an installed application. To do so, you can use the shell option to remove an application from its installed folder: adb shell rm

/data/app/net.learn2develop.LBS.apk Another way to deploy an application is to use the DDMS tool in Eclipse. With an emulator (or device) selected, use the File Explorer in DDMS to go to the /data/app folder and use the "Push a file onto the device" button to copy the APK file onto the device. Using a Web Server: If you wish to host your application on your own, you can use a web server to do that. This is ideal if you have your own web hosting services and want to provide the application free of charge to your users or you can restrict access to certain groups of people. Following are the steps involved: Copy the signed LBS.apk file to c:\inetpub\wwwroot\. In addition, create a new

- HTML file named Install.html with the following content:

On your web server, you may need to register a new MIME type for the APK file.

- The MIME type for the .apk extension is application/vnd.android.packagearchive. From the Application settings menu, check the "Unknown sources" item. You will be prompted with a warning message. Click OK. Checking this item will allow the Emulator/device to install applications from other non-Market sources (such as from a web server). To install the LBS.apk application from the IIS web server running on your computer, launch the Browser application on the Android Emulator/device and navigate to the URL pointing to the APK file. To refer to the computer running the emulator, you should use the special IP address of 10.0.2.2. Alternatively, you can also use the IP address of the host computer. Clicking the "here" link will download the APK file onto your device. Drag the notification bar down to reveal the download status. To install the downloaded application, simply tap on it and it will show the permission(s) required by this application. Click the Install button to proceed with the installation. When the application is installed, you can launch it by clicking the Open button. Besides using a web server, you can also e-mail your application to users as an attachment; when the users receive the e-mail they can download the attachment and install the application directly onto their device.

Q6 b) Differentiate the basics of android UI design and location based services.

We have all seen the explosive growth of mobile apps in recent years. One category of apps that is very popular is location-based services, commonly known as LBS. LBS apps track your location, and may offer additional services such as locating amenities nearby, as well as offering suggestions for route planning, and so on. Of course, one of the key ingredients in a LBS app is maps, which present a visual representation of your location. Here, will learn how to make use of the Google Maps in your Android application, and how to manipulate it programmatically. In addition, you will learn how to obtain your geographical location using the LocationManager class available in the Android SDK.

Getting Location Data

Nowadays, mobile devices are commonly equipped with GPS receivers.

- Because of the many satellites orbiting the earth, you can use a GPS receiver to find your location easily. However, GPS requires a clear sky to work and hence does not always work indoors or where satellites can't penetrate (such as a tunnel through a mountain).

Another effective way to locate your position is through cell tower triangulation.

- When a mobile phone is switched on, it is constantly in contact with base stations surrounding it. o By knowing the identity of cell towers, it is possible to translate this information into a physical location through the use of various databases containing the cell towers' identities and their exact geographical locations.
- The advantage of cell tower triangulation is that it works indoors, without the need to obtain information from satellites.
- It is not as precise as GPS because its accuracy depends on overlapping signal coverage, which varies quite a bit.
- Cell tower triangulation works best in densely populated areas where the cell towers are closely located. A third method of locating your position is to rely on Wi-Fi triangulation.
- Rather than connect to cell towers, the device connects to a Wi-Fi network and checks the service provider against databases to determine the location serviced by the provider

On the Android, the SDK provides the LocationManager class to help your device determine the user's physical location. lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locationListener);
This method takes four parameters:
1. Provider -The name of the provider with which you register. In this case, you are using GPS to obtain your geographical location data.
2. minTime - The minimum time interval for notifications, in milliseconds.
3. minDistance - The minimum distance interval for notifications, in meters.
4. Listener - An object whose onLocationChanged() method will be called for each location update

MODULE 4

Q7 a) write basic concepts of android messaging services for sending SMS and Email.

SMS messaging is one of the main *killer applications* on a mobile phone today — for some users as necessary as the phone itself. Any mobile phone you buy today should have at least SMS messaging capabilities, and nearly all users of any age know how to send and receive such messages. Android comes with a built-in SMS application that enables you to send and receive SMS messages. However, in some cases you might want to integrate SMS capabilities into your own android application. For example, you might want to write an application that automatically sends a SMS message at regular time intervals. For example, this would be useful if you wanted to track the location of your kids — simply give them an Android device that sends out an SMS message containing its geographical location every 30 minutes.

1 Sending SMS Messages Programmatically

To create an application that can send SMS, following are the steps to be followed:

- Create a new android application.
- Add the following statements in to the main.xml file:

```
<Button
android:id=”@+id/btnSendSMS”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Send SMS” />
```

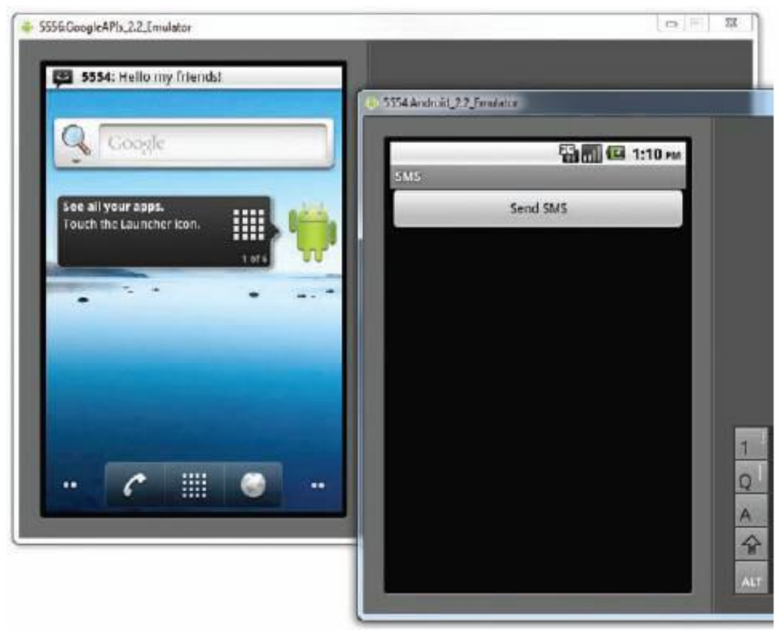
- In the AndroidManifest.xml file, add the following statements:

```
<uses-sdk android:minSdkVersion=”8” />
<uses-permission
android:name=”android.permission.SEND_SMS”>
</uses-permission>
```

- Add the following statements to the MainActivity.java file:

```
import android.app.PendingIntent;
import android.content.Intent;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
public class MainActivity extends Activity
{
Button btnSendSMS;
/** Called when the activity is first created. */

@Override
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
btnSendSMS = (Button) findViewById(R.id.btnSendSMS);
btnSendSMS.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
sendSMS(“5556”, “Hello my friends!”);
}
});
}
private void sendSMS(String phoneNumber, String message)
{
SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage(phoneNumber, null, message, null, null);
}
```

}
 Open two AVDs using AVD Manager. Run your application on one AVD to send SMS to another AVD. It looks as shown below –



Observe the sendSMS() method, which contains sendTextMessage() method. Following are the five arguments to the sendTextMessage() method:
 destinationAddress — Phone number of the recipient
 scAddress — Service center address; use null for default SMSC
 text — Content of the SMS message
 sentIntent — Pending intent to invoke when the message is sent
 deliveryIntent — Pending intent to invoke when the message has been
Delivered

2 SENDING E-MAIL
One can set email through Android program. Following are the steps involved:
 Create a new android application.
 Add the following statements in to the main.xml file:
```
<Button
android:id="@+id/btnSendEmail"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Send Email" />
```
 Add the following statements in bold to the MainActivity.java file:
```
import android.content.Intent;
import android.net.Uri;
import android.view.View;
import android.widget.Button;
```

```
public class MainActivity extends Activity
{
Button btnSendEmail;
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
btnSendEmail = (Button) findViewById(R.id.btnSendEmail);
btnSendEmail.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
String[] to = {“weimenglee@learn2develop.net”,
“weimenglee@gmail.com”};
String[] cc = {“course@learn2develop.net”};
sendEmail(to,cc,“Hello”, “Hello my friends!”);
}
});

}
//---sends an SMS message to another device---
private void sendEmail(String[] emailAddresses, String[]
carbonCopies, String subject, String message)
{
Intent emailIntent = new Intent(Intent.ACTION_SEND);
emailIntent.setData(Uri.parse(“mailto:”));
String[] to = emailAddresses;
String[] cc = carbonCopies;
emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
emailIntent.putExtra(Intent.EXTRA_CC, cc);
emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
emailIntent.putExtra(Intent.EXTRA_TEXT, message);
emailIntent.setType(“message/rfc822”);
startActivity(Intent.createChooser(emailIntent,“Email”));
}
}
```

 Test the application by running it. When you click on *Send Mail* button you can see Email application launched on device as shown below –

10:33 AM
weimenglee@gmail.com
<weimenglee@learn2develop.net>, <weimenglee@gmail.com>,
<course@learn2develop.net>,

Hello
Hello my friends!

Q7 b) What is the process for binding activities to services in android application?

Real-world services are usually more sophisticated, requiring the passing of data so that they can do the job correctly for you. Using the service demonstrated earlier that downloads a set of files, suppose you now want to let the calling activity determine what files to download, instead of hard coding them in the service. Here is what you need to do.
First, in the calling activity, you create an Intent object, specifying the service name:

```
Button btnStart = (Button) findViewById(R.id.btnStartService);
btnStart.setOnClickListener(new View.OnClickListener()

{
public void onClick(View v)
{
        Intent intent = new Intent(getBaseContext(), MyService.class);
}
});
```

You then create an array of URL objects and assign it to the Intent object through its putExtra() method. Finally, you start the service using the Intent object:

```
Button btnStart = (Button) findViewById(R.id.btnStartService);
btnStart.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
Intent intent = new Intent(getBaseContext(), MyService.class);
try
{
URL[] urls = new URL[]
{
```

```
new URL("http://www.amazon.com/somefiles.pdf"),
new URL("http://www.wrox.com/somefiles.pdf"),
new URL("http://www.google.com/somefiles.pdf"),
new URL("http://www.learn2develop.net/somefiles.pdf")
};
intent.putExtra("URLs", urls);
} catch (MalformedURLException e)
{
e.printStackTrace();
}
startService(intent);
}
});
```

Note that the URL array is assigned to the Intent object as an Object array. On the service's end, you need to extract the data passed in through the Intent object in the onStartCommand() method:

```
@Override
public int onStartCommand(Intent intent, int flags, int startId)
{
// We want this service to continue running until it is explicitly
// stopped, so return sticky.
Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
Object[] objUrls = (Object[]) intent.getExtras().get("URLs");
URL[] urls = new URL[objUrls.length];
for (int i=0; i<objUrls.length-1; i++)
{
urls[i] = (URL) objUrls[i];

}
new DoBackgroundTask().execute(urls);
return START_STICKY;
}
```

The preceding first extracts the data using the getExtras() method to return a Bundle object. It then uses the get() method to extract out the URL array as an Object array.
Because in Java you cannot directly cast an array from one type to another, you have to create a loop and cast each member of the array individually. Finally, you execute the background task by passing the URL array into the execute() method.
This is one way in which your activity can pass values to the service. As you can see, if you have relatively complex data to pass to the service, you have to do some additional work to ensure that the data is passed correctly. A better way to pass data is to bind the activity directly to the service so that the activity can call any public members and methods on the service directly.

Q8 a) What is the techniques involved to create own services and building web services in android?

A service is an application in Android that runs in the background without needing to interact with the user. For example, while using an application, you may want to play some background music at the same time. In this case, the code that is playing the background music has no need to interact with the user, and hence it can be run as a service. Services are also ideal for situations in which there is no need to present a UI to the user. Following are the steps involved in creating own service.

 Create a new android application.

 Add a new class file to the project and name it MyService.java. Write the following code in it:

```
package net.learn2develop.Services;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;
public class MyService extends Service
{
@Override
public IBinder onBind(Intent arg0)
{
return null;
}
public int onStartCommand(Intent intent, int flags, int startId)
{
Toast.makeText(this,“ServiceStarted”,Toast.LENGTH_LONG).show();
return START_STICKY;
}
public void onDestroy()
{
super.onDestroy();
Toast.makeText(this,“ServiceDestroyed”,Toast.LENGTH_LONG).show();
}
}
```

The onBind() method enables you to bind an activity to a service. This in turn enables an activity to directly access members and methods inside a service. The onStartCommand() method is called when you start the service explicitly using the startService() method. This method signifies the start of the service, and you code it

to do the things you need to do for your service. In this method, you returned the constant START_STICKY so that the service will continue to run until it is explicitly stopped. The onDestroy() method is called when the service is stopped using the stopService() method. This is where you clean up the resources used by your service.

 In the AndroidManifest.xml file, add the following statement :

```
<service android:name=".MyService" />
```

 In the activity_main.xml file, add the following statements in bold:

```
<Button android:id="@+id/btnStartService"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Start Service" />
<Button android:id="@+id/btnStopService"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Stop Service" />
```

 Add the following statements in bold to the MainActivity.java file:

```
import android.content.Intent;
import android.view.View;
import android.widget.Button;
public class MainActivity extends Activity
{
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
Button btnStart = (Button) findViewById(R.id.btnStartService);
btnStart.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
startService(new Intent(getBaseContext(), MyService.class));
}
});
Button btnStop = (Button) findViewById(R.id.btnStopService);
btnStop.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
stopService(new Intent(getBaseContext(), MyService.class));
}
});
}
}
```

Q8 b) Explain the concepts of networking in terms of communicating between service and activity.

Often a service simply executes in its own thread, independently of the activity that calls it. This doesn't pose any problem if you simply want the service to perform some tasks periodically and the activity does not need to be notified of the status of the service. For example, you may have a service that periodically logs the geographical location of the device to a database. In this case, there is no need for your service to interact with any activities, because its main purpose is to save the coordinates into a database. However, suppose you want to monitor for a particular location. When the service logs an address that is near the location you are monitoring, it might need to communicate that information to the activity. In this case, you would need to devise a way for the service to interact with the activity.

BINDING ACTIVITIES TO SERVICES

Real-world services are usually more sophisticated, requiring the passing of data so that they can do the job correctly for you. Using the service demonstrated earlier that downloads a set of files, suppose you now want to let the calling activity determine what fi les to download, instead of hardcoding them in the service. Here is what you need to do. First, in the calling activity, you create an Intent object, specifying the service name:

```
Button btnStart = (Button) findViewById(R.id.btnStartService);

btnStart.setOnClickListener(new View.OnClickListener()

{

public void onClick(View v)

{

Intent intent = new Intent(getBaseContext(), MyService.class);

}

});
```

You then create an array of URL objects and assign it to the Intent object through its

putExtra() method. Finally, you start the service using the Intent object:

```
Button btnStart = (Button) findViewById(R.id.btnStartService);

btnStart.setOnClickListener(new View.OnClickListener()

{

public void onClick(View v)

{

Intent intent = new Intent(getBaseContext(), MyService.class);

try
{
URL[] urls = new URL[]
{
new URL("http://www.amazon.com/somefiles.pdf"),
new URL("http://www.wrox.com/somefiles.pdf"),
new URL("http://www.google.com/somefiles.pdf"),
new URL("http://www.learn2develop.net/somefiles.pdf")
};
intent.putExtra("URLs", urls);
} catch (MalformedURLException e)
{
e.printStackTrace();
}
startService(intent);
}
});
```

Note that the URL array is assigned to the Intent object as an Object array. On the service's end, you need to extract the data passed in through the Intent object in the onStartCommand() method:

```
@Override
public int onStartCommand(Intent intent, int flags, int startId)
{
// We want this service to continue running until it is explicitly
// stopped, so return sticky.
Toast.makeText(this, “Service Started”, Toast.LENGTH_LONG).show();

Object[] objUrls = (Object[]) intent.getExtras().get(“URLs”);

URL[] urls = new URL[objUrls.length];

for (int i=0; i<objUrls.length-1; i++)

{

urls[i] = (URL) objUrls[i];

}
new DoBackgroundTask().execute(urls);
return START_STICKY;
}
```

The preceding first extracts the data using the getExtras() method to return a Bundle object. It then uses the get() method to extract out the URL array as an Object array. Because in Java you cannot directly cast an array from one type to another, you have to create a loop and cast each member of the array individually. Finally, you execute the background task by passing the URL array into the execute() method.
This is one way in which your activity can pass values to the service. As you can see, if you have relatively complex data to pass to the service, you have to do some additional work to ensure that the data is passed correctly. A better way to pass data is to bind the activity directly to the service so that the activity can call any public members and methods on the service directly.

MODULE 5

Q9 a) how to build derby app in windows phone 7? Explain.

Here, you implement the features of the Derby Names project using Microsoft Visual Studio, while also taking time to learn Windows Phone 7–specific technologies.

Creating the Project:
Open Visual Studio and create a new Windows Phone project. For this application, choose Panorama because it offers a UI in which you can share your data.
In a Panorama application the application is created with the default Panorama background. Visual Studio will create SampleData and ViewModels for your application. Ultimately, you will be able to remove these from your application when you implement your service communications. App.xaml is the entry point for your application and MainPage.xaml is the page that loads by default.

User Interface:
The default Panorama application defines its DataContext in XAML. The DataContext has first item’s binding associated by default. The Panorama control can be likened to any collection-

based UI element (UITableView in iOS or the ListView in Android), and the PanoramaItems are the respective rows in that collection element. When you feel familiar enough to start working with the data you will need to create a service reference to the Odata feed.

To reference an OData feed you need only to right-click your project, choose Add Service Reference, enter in the URL of your service, and click Go. After it has found the service it should enumerate the models. You are then allowed to update the namespace and create this reference. Once you create the service you can start working with the Panorama control to bind the data available from these entities. After you have made this service, be sure to reference this entity context when your page needs to make calls to the service:
readonly DerbyNamesEntities context = new DerbyNamesEntities(new Uri("http://localhost:1132/DerbyNames.svc/"));

Derby Names:
To bind data to your Panorama item you need to set the ItemsSource and TextBlock bindings. Each individual entry in the DerbyNames entity in OData contains properties for Name and League, which you will bind to the TextBlocks in your Panorama item.

Leagues: Each derby team belongs to a league. The entity for League is similar to the DerbyNames entity, and will make it easy to bind from.

Q9 b) Explain the anatomy for windows phone 7? (8 marks)

Here we discuss the basic design elements used in Windows Phone 7 application development, and how you can leverage the tools you have at hand to implement them.

1. **Storyboards:** Storyboards are Silverlight's control type for managing animations in code. They are defined in a given page's XAML and leveraged using code behind. Uses for these animations are limited only by the transform operations you are allowed to perform on objects.

Anytime you want to provide the user with a custom transition between your pages or element updates, you should consider creating an animation to smooth the user experience.
Because storyboards are held in XAML you can either edit them manually or use Expression Blend's WYSIWYG editor.

In Blend, in the Objects and Timelines Pane at the left, click the (+) icon to create a storyboard. Once you have a storyboard, you can add key frames on your time line for each individual element you would like to perform a transformation on. This can include moving objects and changing properties (like color or opacity). After setting up your time line, you can start the storyboard in code. The name you created for your storyboard will be accessible in code behind.

2. **Pivot vs Panorama:** Both the Pivot and Panorama controls are used to delineate categories and subsets of data. With the Pivot control you get strict isolation of these groupings, with the menu providing discoverable UI to show the other categories. 2. With the Panorama control you get transitions between the groupings with discoverable content on the window boundaries.

The Windows Phone Emulator
The Windows Phone 7 emulator is a very powerful tool. Not just a simulator, the emulator runs a completely sandboxed virtual machine in order to better mimic the actual device. It also comes with some customization and runtime tools to manipulate sensors that are being emulated on the device, including GPS and accelerometer, as well as provide a way to capture screenshots while testing and developing applications. The debugging experience inside of Visual Studio is superior to the ones in Eclipse and the third-party frameworks. The load time of the Emulator is quite fast. It acts responsively, and the step-through just works.

Q10) write the short note on following with appropriate examples (16 marks)

a) **Components of xcode**

The iPhone SDK includes a great number of tools that help create iOS for apps. These tools range from debugging and profiling to developing.
Following is the list of most common tools that we use that are included in the iOS SDK:

- **xCode:** xCode is Apple's Integrated Development Environment (IDE) for creating Objective-C applications. xCode enables you to manage, author, and debug your Objective-C projects.

- **Dashcode:** Dashcode is an IDE that enables you to develop web-based iPhone/iPad applications and Dashboard widgets.

- **iPhone Simulator:** This tool provides a method to simulate an iPhone or iPad device on your Mac, for use with testing your iOS applications.

- **Interface Builder:** The Interface Builder, or IB, is a visual editor that is used for designing the user interface for your iOS application.

- **Instruments:** Instruments is a suite of tools that helps you analyze your iOS application and monitor for performance bottlenecks as well

b) **Tools required for window phone 7**

To develop software for Windows Phone 7 you need a machine running Windows (Vista or Windows 7), Visual Studio 2010, and the Phone SDK, as well as a Windows Phone 7 device to test with.

Hardware: HTC, Nokia, and Samsung are currently manufacturing Windows Phone 7 devices.

Device resolutions are 800X480 pixels, and most are outfitted with both front- and rear-facing cameras, and screens from 4.3 to 4.7 inches. They are primarily found on GSM carriers.

Visual Studio and Windows Phone SDK:
•Code for the Windows Phone is written in the .NET Framework, either with XNA (Microsoft's run time for game development), or a custom version of Silverlight (Microsoft's Rich Internet Application framework).
• User interfaces are created with XAML (eXtensible Application Markup Language).
• The Windows Phone SDK works with Visual Studio Express, and will install it if you don't already have it installed.
– If you have another version of Visual Studio 2010 on your machine it will add the functionality to that install.
– The SDK also installs a specialized version of Expression Blend set up to work specifically for Windows Phone 7 development.
• Expression Blend is a tool developed by Microsoft for working with XAML.
– It provides similar functionality to Visual Studio, though its layout is designed to be more user friendly to designers.

Installation:
• Microsoft's App Hub (http://create.msdn.com/) is the download site for the Windows Phone SDK.
• The Windows Phone SDK installer includes:
– Visual Studio 2010 Express for Windows Phone (if you do not have another version of Visual Studio 2010 installed)
– Windows Phone Emulator
– Windows Phone SDK Assemblies
– Silverlight 4 SDK
– Phone SDK 7.1 Extensions for XNA Game Studio
– Expression Blend for Windows Phone 7
– WCF Data Services Client for Windows Phone
– Microsoft Advertising SDK
To install the Windows Phone SDK you need:
• Vista (x86 or x64) or Windows 7 (x86 or x64)
• 4 GB of free disk space
• 3 GB of RAM
• DirectX 10 or above capable graphics card with a WDDM 1.1 driver

c) Getting usefull ios things

IOS applications can be used do many more tasks as explained in following sections.

1. **Offline Storage**

Even if your application is using a web service for retrieving information, at some point you may need to save information on the device. Depending on the size and type of data, you have a few different options.
Plist : Property lists are the simplest way to store information on the device. In the Mac world, many applications use the plist format to store application settings, information about the application, and even serialized objects. It's best to keep the data contained in these files simple and small, though.
The following example finds the path to a plist stored in the supporting files directory, with a name of example. It then loads the plist into a dictionary object, and loops through each time writing the contents of each item in the plist to the debug console.

```
- (void)getValuesFromPlist
{
// build the path to your plist
NSString *path = [[NSBundle mainBundle] pathForResource:
@"example" ofType:@"plist"];
// load the plist into a dictionary
NSDictionary *pListData = [[NSDictionary alloc]
initWithContentsOfFile:path];
// loop through each of the Items in the property list and log
for (NSString *item in pListData)
NSLog(@"Value=%@", item);
}
```

Core Data: If the data that you need to persist on the device is nontrivial, meaning there is a great deal of it or its complex, Core Data is the way to go. Core Data is described by Apple as a "schemadriven object graph management and persistence framework." Core Data is not an ORM (Object Relational Mapper). Core Data is an API that abstracts the actual data store of the objects. Core
Data can be configured to store these objects as a SQLite database, a plist, custom data, or a binary file. Core Data has a steep learning curve, but is well worth learning more about if your app will have a great deal of data held within.

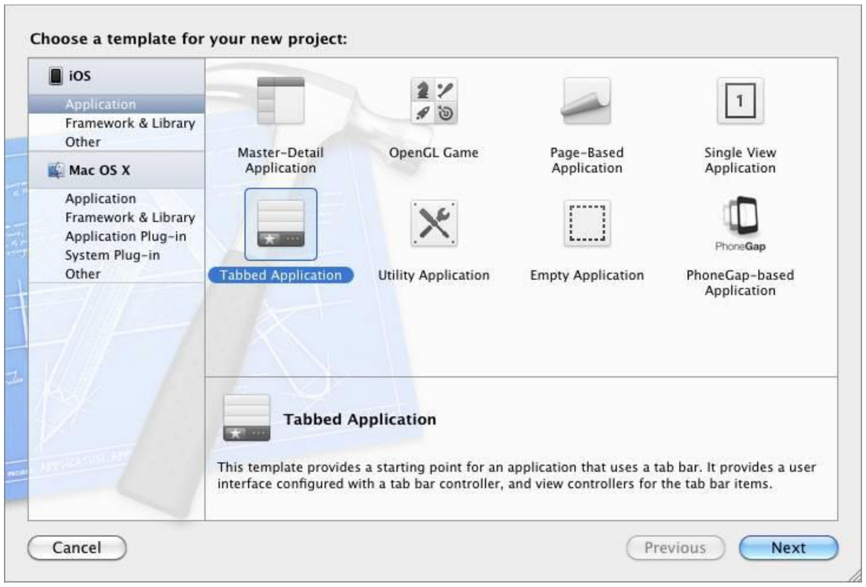
2. GPS

One of the great benefits to mobile devices is the GPS functionality. Once you are able to get over the hurdles of learning the basic functions within the iOS platform, starting to work with the GPS functions can be a great deal of fun. The GPS functions are located in the CoreLocation framework, which is not added to a new project by default. To do this, you will need to click the Build Phases tab on the project settings page.
Once on the Build Phases tab, expand the Link Binary with Libraries section, and click the + button.
You are then prompted with a list of frameworks to add. Select the CoreLocation.framework.
Use the code given below –

```
// GPS Example
locationManager = [[CLLocationManager alloc] init];
locationManager.delegate = self;
locationManager.distanceFilter = kCLDistanceFilterNone;
// get GPS DatalocationManager.desiredAccuracy =
kCLLocationAccuracyHundredMeters;
[locationManager startUpdatingLocation];
```

d)Derby app in IOS

The idea of the Derby App is to build the same app over all of the mobile platforms. The iOS version is very similar to the program built in Android. Following are the steps to be followed:
1. **User Interface**
2. **Team Roster**
3. **Details**
4. **Leagues and Team Names**
We will discuss each of the steps now.
User Interface: The derby app contains a list of data, so you use table views throughout the application to show the information to the user. Start by creating a new tabbed application within
xCode as shown in Figure 5.3:

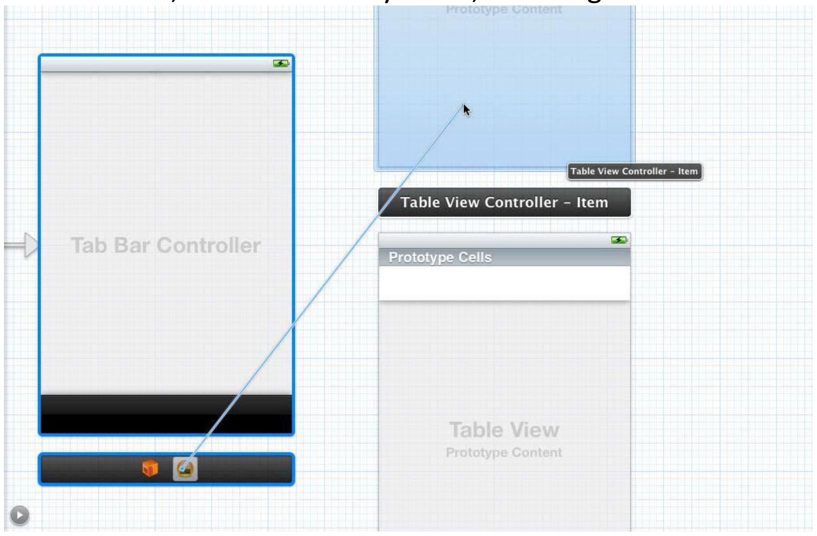


Tabbed Application

This template provides a starting point for an application that uses a tab bar. It provides a user interface configured with a tab bar controller, and view controllers for the tab bar items.

Your project needs to contain three table views and one Navigation controller. The table views are used to the list the data about the derby teams and rosters, and the navigation controller enables you to create a "back stack" that will allow users to navigate back to the team names table view, when finished viewing the team's roster data. Start by dragging three Table View Controllers from the Object Library onto the storyboard, and then remove the views that were added by default.

Once the previous views are removed and the table views are added, you need to connect them to the navigation controller. To do this, Control-click the Tab Bar Controller icon on the Tab Bar View, within the Storyboard, and drag it to the new view as shown in Figure 5.4.



Team Roaster: With the storyboard in place, you can now write the code to populate the table views. Start with the Team Roster tab, which will go out to a web service, and obtain a list of all the Lansing Derby Vixens. First create a new class named VixensViewController:

```
@interface VixensViewController : UITableViewController
@property(nonatomic, retain) NSArray *listData;
@end
```

In the viewDidLoad method within the implementation file, add your logic to go out the web service and get the roster for the Lansing Derby Vixens. Notice the filter criteria contained within the URL.
iOS 5 is the first version of the iOS SDK that contains built-in support for parsing JSON strings. This is great because you do not have to depend on a third-party tool.

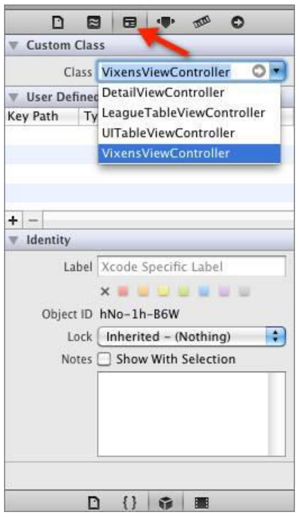
Details: The next class you need to create is DetailViewController. This is the code that drives the details that are displayed when a team name is selected.

```
#import <UIKit/UIKit.h>

@interface DetailViewController : UITableViewController
@property (nonatomic, retain) NSString *data;
@property(nonatomic, retain) NSArray *listData;
@end
```

Leagues and Team Names: Listing all of the team names is very similar to the code you just created for displaying the roster and the details. Start by creating a new class named LeagueTableViewController:

```
#import <UIKit/UIKit.h>
@interface LeagueTableViewController : UITableViewController
@property(nonatomic, retain) NSArray *listData;
@end
```

After all of the code has been added to the new View Controllers you created, you need to go back to the Storyboard and attach them as shown in Figure 5.5. You do this using the Identity Inspector found near the upper right of the screen when you are viewing the storyboard. Map each view to the correct class that was created. Final application looks as given in Figure 5.6.



Custom Class

Class VixensViewController

DetailViewController
LeagueTableViewController
UITableViewController
VixensViewController

User Defined

Key Path Ty

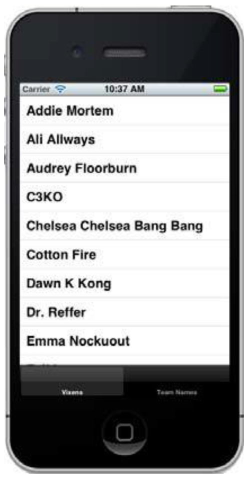
Identity

Label Xcode Specific Label

Object ID hNo-1h-B6W

Lock Inherited - (Nothing)

Notes Show With Selection



Carrier 10:37 AM

Addie Mortem
Ali Allways
Audrey Floorburn
C3KO
Chelsea Chelsea Bang Bang
Cotton Fire
Dawn K Kong
Dr. Reffer
Emma Nockuout