Mini2 Final Lessons Learned

Team Member:
Li Pei : lip@andrew, Weiting Zhai : wzhai@andrew, Tongyun Lu : Tongyunl@andrew

Android Lifecycle, Activity and Service

---

1. What is Android Activity.

An Activity is an application component that provides a screen with which users can interact in order to do something, such as input some text, take a photo or send some voice. Each activity is given a window in which to draw its user interfaces. The window typically fills the screen, but may be smaller than the screen and float on top of their windows. In out application, activity is used for UI that support user to input information to system, and get feedback from app.

2. What is Android Service.

A Service is an application component that can perform long-running operations in the background and does not provide a user interface. Another application component can start a service and it will continue to run in the background even if the user switches to another application.

Android service do not live on the lifecycle of callback because it did not interact with user. For this can of project, it can be start by calling the Context.startService() and stopped by Context.stopService(). It is impressive that only one Context.stopService() need to be called to stop the service, no matter how many call of start service. The service also could stop itself.

In our application, we use two local or remote service, including a background music playing and a email service, which is supported background. However, if you triggered some action to start/stop the music or writing mail with pressing button or corresponding gesture, you could get response by the Service running background and do what you require.

3. State of Activity of Android APP

An activity in Android can exist in four states below

1.Active and Running state. This is a state when an activity is in the front and has focus in it. It is completely visible and active to the user.

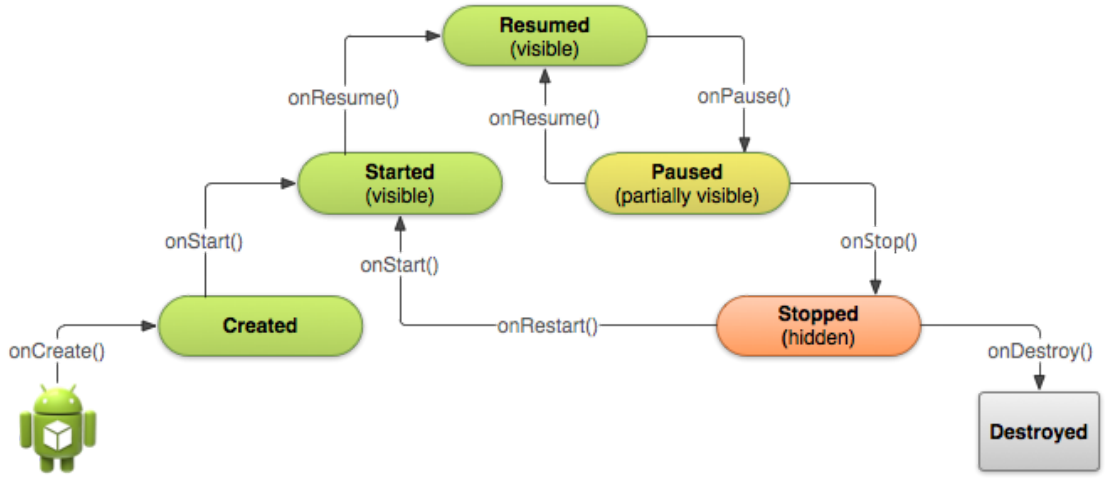
2.Pause state. In paused state, the activity is partially visible to the user, but not active and lost focus.

3.Stopped state. This is when the Activity is no longer visible in the screen. Another activity is on top of it and completely obscures its view. In this state also the activity is alive and preserves its state, but more likely to be killed by the system to free resources whenever necessary.

4. Destroyed/Dead state. An Activity is said to be dead or destroyed when it no longer exists in the memory.

In our activity, we handled the use different states and applied the life cycle safely in application.

4. Android APP lifecycle is like picture below



The above Lifecycle of Android Activity diagram can be explained as follows:

When we launch an Activity in Android, it first calls onCreate() method, then onStart() method is called before the Activity is being visible to User. Then onResume() method is called to set the activity visible. onPause() method will be called when the system is about to resume another Activity on top of this one and when another part of system is called. When the onStop() method is called, it could get started by calling onRestart(), onStart() and onResume() method, but when it start onDestroy() method, the lifecycle will be break and Activity is destroyed.

5. About Android onResume() method

With onStart() method, Activity become visible but not active, With the onResume() method, the Activity become Active for the user to interact with. The Activity will be at the top of the Activity stack at this point. Now the Activity is in running /active state and is able to receive user inputs.

In our application, the onResume() is override for most cases to custom the behavior when the screen is reset.

6. About Android onPause() method

In the Active state, onPause() method will be called when the system is about to resume another Activity on top of this one or when the user is about to navigate to some other other parts of the system. It is the last guaranteed call to a method before the Activity can get killed by the system. That is, there's a possibility that your activity may be killed by the system at the paused state without executing any further method calls. Therefore it is important to save the user interface configuration and critical data at this method.

On a physical device, the user's action could trigger the Activity. By default, an Activity can remain in the paused state if user pressed the home button. Another activity or notification which is on top of it doe not completely obscures the visibility of underlying Activity. The device goes to sleep.

On a physical device, the user's action could trigger the Activity. By default, an Activity can remain in the paused state if user pressed the home button. Another activity or notification which is on top of it doe not completely obscures the visibility of underlying Activity. The device goes to sleep.

There are two possibility for an Activity under paused state:

1.The User resumes the Activity by closing the new Activity or notification and the paused Activity gets Active/Running by calling onResume() method.

2.It gets killed by the system under extremely low memory conditions. In this case there will be no further method calls before the destruction of the Activity and it needs to be rerun from the beginning by calling onCreate() and restoring the previous configuration from bundle object.

7. About stop states of app

1. System kills it to free resources. An activity under stopped state is more likely to be killed by system than one in the paused state. It needs to start the cycle again with onCreate().

2. It get restarted by calling onRestart(), onStart() and onResume() methods in the order if the user navigates back to the Activity again. In this case, the UI is intact and no need to be restored.

3. onDestroy() method is called and the Activity is destroyed. This is the final method we can call before the Activity is destroyed. This occurs either because the Activity is finishing the operation or the system is temporarily destroying it to save place.

In our application, the stop states is safely handled for every page to prevent wrong back page or double back time situation.

8. Three Android lifecycle loops for every

1. Entire Lifetime: This is the lifetime between the first call to the onCreate() and the final call to onDestroy() method. We create all global resources such as screen layout, global variables etc in onCreate() and release all resources with onDestroy() call.

2. Visible Lifetime. It is lifetime of an Activity between onStart() and onStop() method calls. In this the Activity is visible to the user and he may or may not be able to interact with it. During the visible lifetime, an Activity maintains its state intact.

3. Foreground Lifetime. Foreground lifetime starts with onResume() and ends with onPause() method calls. During this, the Activity is completely visible to the user and is on top of all other Activities so that user can interact with it.

Object Oriented Design

---

9. Encapsulation was applied in every class of project. Encapsulation is an approach to design the model with modularity and abstraction. We set the data field or method to private and if you want to change some of them, you may use get-set method. This is used to hide the complexity of this class component, and prevent changing these data or method from outside directly.

10. Use inner class in our project, especially for UI activity. The inner class is used mostly as a button listener. It could handle the button action customized, including variable in the inner class. And it is totally invisible outside the class, if you set it private. It helped modularize the project part. Handle the specific function in a class, but limit the scope of the class.

11. Learn to design class according to the data we need. In our project, we set two group of data objects to handle, one group is sports activity, and the other is nutrition. Members in the group all have same fields, but different factors, so they all extends an abstract parent class, Activity and Nutrition, which set all the shared getter and setter and fields, but not define the specific value of fields. This is how we design data object.

12. Good Coding convention for Android development is important. In this project, we give package name which define the module of project. We give “String” need to be hardcoded a unified resource location, and give it an id for its purpose. We give class name with activity or object purpose which is clear to understand. Comment is detailed but brief to understand the purpose of code. And finally keep the style of code constant.

13. Inheritance is used in object relationship. For inheritance relationship, you get an “has-a” relationship where one class could be an abstraction of other class. For example, Sports Activity may “has-a” football, basketball or running class.

14. Interface is implemented in the project. In our project, a Person class include lots of properties. Some of them are public, but some of them are supposed to be private. In our design, we build an interface called Friend. For friends interface, you could get all getters for public information of a Person, but no getter and setter of private information. Person class should implement the Friend class, and your friend will be instantiated with the reference class defined as Friend, which guarantee you could not access their password field, and can not change their information, but only read it.

15. Abstract class is used in our project to define Sports Activity and Nutrition. We have lots of activity and nutrition object, they all share the same field, as calorie factor, name, however, the actual value is not same. So we set the field and getter setter in abstract class, but set it specific in their own class. So each specific activity and nutrition class that implement abstract class, they do not need to recode the shared function and field, they should only type in the specific method and field.

16. Exception handling
In our project, we focus on exception handling and handle every input, page transfer, service exception with customized method. For example, for input exception, we limit the input type and length first, and send toast when you still input illegal. Another example, when you press some button for server access function, if the server did not work, there will be a toast to tell you “No Internet Access”. This will also be handled background in our application.

17. Make static class for project is implemented in our project. In our Person class, there is a static object to save the information change. The static object in the class guaranteed that when you instantiate a new class, this field will stay static, and all instance of this class share the same object.

18. In our project, we use multithreading method to access local and remote database, because access to database is relatively slow and will slow down the generation and transition of pages. Furthermore, for Cursor class in android, asynchronous method is required for access to database, so we implemented Async method for accessing local SQLite database, and threading socket client to access remote database.

19. Avoid race condition for two thread access the same object is an issue for our project. Because if we create, save and update a Person object in thread accessing database, main thread’s other work or other thread update the same Person will come into a race condition, which will corrupt the information of user. So when we use thread for accessing database, we synchronized this object, and other access for this user would be waiting until this object is released from this thread, then then race condition is avoid.

20. In our project, we also face the producer-customer issue, where we get the information of person from a separate thread but edit or use it in main thread. It occurs that sometimes the edit or use process begin before the thread we get data from database. To handle this issue, we implement a empty while loop as

```
while(client thread not stop){
        empty
}
```
to wait for the data transfer thread, and it will help keep the order of getting and using data.

21. Use socket for communicate in our project is the core of client-server communication. In our project, we set up a iterative server socket to listen and accept request from client. Each request is handled by a thread in server. On client end, each request is send in a client thread through the socket, and handled by server with a new server thread.

22. We need to handle socket communication in independent thread because ServerSocket has blocking feature, the call to “accept” causes the program to wait until the method returns. This blocking call can cause a program to hang. If other operations must take place, we need some ways of placing the accept class in its own thread. In our project, every create, update and save method for user should have corresponding method for database information. So in our project, in case hanging for UI and user experience, the communication should work in a independent thread which make the work done in background.

23. Object stream in socket is the main approach for information communication for our project. With the use of object stream, DatabasePerson object, which is the main information object of our project, is able to be transferred to store in server database, and could be updated and restored from database information.

24. Write protocol to make sure the safe communication between client and server is important for our project. For this socket programming unit, protocol is not irrelevant to physical layer, but only the agreement between server and client about the sequence of dialog and how to handle each request.

To write protocol program, I think order is the first thing to consider. What is the order of input and output, when the input and output will occur is considered before programming. In this program I marked all the specific communication with comment and make sure request and response are corresponded exactly between client and server.

25. Stop thread and close socket and I/O stream after use is implemented in our project. For communication, we have to open socket and I/O stream. However, if you have finished the thread work, you would better close socket and stream to prevent EOFException and other IO exception with this unclosed, no referenced socket and stream open.

26. Get information from thread is something need to be designed for our project, because thread has no “return” for other threads. In our project, we have to get and set information of Person in thread to communicate with server, so we have to pass the object in its constructor, and offer it a reference in the thread. Use the reference for thread operation, and in the thread synchronize it in case other thread change it. After thread ends, this object is already operated and reference outside this thread could also see these operation result.

27. How to test the network access for a server based application is something we should care. You cannot return something from socket thread, but you could get referenced object in it. So in this project, I designed a referenced ArrayList for saving internet status as string in it. After thread run over, we could get all internet access status in this array, and if the server is closed, a toast will on the screen.

28. Design of database is implemented in our project. In our project, a one table database is used to store each user as a row, and we use JDBC connector to access database.

29. In our project, we only use primary key of database. The primary key is username, because username is the unique property for every user of this app.

30. Use two step to design the database, the first step in designing a database application is gathering requirements. This would include functional requirements, data requirements, performance requirements. the second step is developing a logical data model.
A logical data model is representation of the data elements used by an enterprise and relationships between those data elements. One of the most common methods for developing a logical model is entity relationship modeling.

Android Feature

31.gallery scan

In android devices, the gallery should scan images stored in the sdcard automatically. However, sometimes android' gallery will not do that. because of two reasons below.

1. ./nomedia is a file used for hiding the image file from the gallery. If you put it into your file accidently, the gallery will not scan it, which means pictures will not be shown inside the gallery. You can download the re manager and then you delete ./nomedia. If pictures still not be shown in the gallery, maybe because the gallery has already ignored the picture path. When the gallery scan the picture again, it will ignore the picture automatically.

2.You can add a media scanner inside your android code, in order to force the gallery to scan the image file. The code is as below, please put the created file path into this scanner.

```
public void fileScan(String filePath){
Uri data = Uri.parse("file://"+filePath);
 sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE data));
}
```

32. Camera API

If you want to use the camera and save it inside the sdcard, first you have to add the permission inside the manifest.xml including camera used permission and external read and write permission. Next you have to use the intent to start the camera, and then you have to create the file path and store it into the default path named sdcard storage. When using the camera, you should make sure to handle the exception of failing to take a picture. If you want to take more than two pictures on the page, you have to use picture flags to differentiate pictures from each other.

33. Storage Options

Android provide several storage options.

1.Shared preferences: store private data in key-value pairs.

2.internal storage:store private data on the device memory.

3.external storage:store public data on the shared external storage. We use external storage to store pictures. We have to put "<uses-permission android:name=”android.permission.WRITE_EXTERNAL_STORAGE” />” inside the manifest.xml

4.SQLite Databases:store structured data in a private database. Android provides full support for SQLite databases. Any databases you create will be accessible by name to any class in the application, but not outside the application. The recommended method to create a new SQLite database is to create a subclass of SQLiteOpenhelper and override the onCreate() method, in which you can execute a SQLite command to create tables in the database.

5.Network Connection: store data on the web with your own network server

34. How to make the picture be suitable for more than two sizes of screen?

It is really hard to make pictures be suitables for many screen sizes. If we really want to do that, we have a few methods to achieve that.

1.ScrowView in the layout in order to make the screen be extensive.

2.we should use “fill parent” for width size in order to wrap the screen width.

3.We can change the picture size ratio by using the android weight. If we make two picture both weight 1, each picture will wrap the width half.

4.Besides, we can use linearlayout to make pictures or TextView in the same line.

35. Common layouts

Each subclass of the ViewGroup class provides a unique way to display the views you nest within it. Below are some of the more common layout types that are built into the Android platform.

You should pay attention to the layout hierarchy. Although you can nest one or more layouts within another layout to achieve your UI design, you should strive to keep your layout hierarchy

as shallow as possible. Your layout draws faster if it has fewer nested layouts (a wide view hierarchy is better than a deep view hierarchy).

1.Linear Layout: A layout that organize it children into a single horizontal or vertical row. It creates a scrollbar if the length of the window exceeds the length of the screen

2.Relative layout: Enables you to specify the location of child objects relative to each other(child A to the left of child B) or to the parent(aligned to the top of the parent)

36. Building Layouts with an Adapter

When the content for your layout is dynamic or not pre-determined, you can use a layout that subclasses AdapterView to populate the layout with views at runtime. A subclass of the AdapterView class uses an Adapter to bind data to its layout. The Adapter behaves as a middleman between the data source and the AdapterView layout. The Adapter retrieves the data(from a source such as an array or a database query) and converts each entry into a view that can be added into the AdapterView layout.

1.ListView : Display a scrolling single column list.

2.Grid View : Display a scrolling grid of columns and rows

37. Filling an Adapter view with data

You can populate an AdapterView such as ListView or GridView by binding the AdapterView instance to an Adapter, which retrieves data from an external source and creates a View that represents each data entry.

Android provides several subclasses of Adapter, most common adapters are:

1.ArrayAdapter: Use this adapter when your data source is an array. By default, ArrayAdapter creates a view for each array item by calling toString() on each item and placing the contents in a TextView. In our application,we create an ArrayAdapter and then use ListView to set the adapter. We use ArrayAdapter to achieve the list of friends

2.SimpleCursorAdapter: use this adapter when your data comes from a cursor. When using SimpleCursorAdapter, you must specify a layout to use for each row in the Cursor and which columns in the Cursor should be inserted into which views of the layout.

38. Using a Service with MediaPlayer

If you want your media to play in the background even when your application is not onscreen-that is, you want it to continue playing while the user is interacting with other applications, then you must start a Service and control the MediaPlayer instance from there. You should be careful about this setup, because the user and the system have expectations about how an application running a background service should interact with the rest of the system.

For instance, when using a MediaPlayer from your main thread, you should call prepareAsync() rather than prepare(), and implements a mediaPlayer.OnpreparedListener in order to be notified when the preparations is complete and you can start playing.

39. Responding to touch events

Making objects move according to a preset program like the rotating triangle is useful for getting some attention, but what if you want to have users interact with your OpenGL ES graphics? The key to making your OpenGL ES application touch interactive is expanding your implementation of GLSurfaceView to override the onTouchEvent() to listen for touch events.

In order to make your project responds to touch events, you must implement the onTouchEvent() method in your GLSurfaceView class. The example implementation below shows how to listen for MotionEvent.ACTION_MOVE events and translate them to an angle of rotation for a shape.

In our application, we use touch events to turn on or turn off the music

40. Toasts

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. We use the toast in every page.

1.when the service is closed or the internet is closed, toast will be sent on the screen.

2.Any input problems will be showed and some hints will also be shown.

3.When the function succeed or fail, toast will be sent on the screen.

4.When the music turn up or down, the toast will tell you how to turn off or turn on the music.

41. Dialogue

A dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for modal events that require users to take an action before they can proceed.

In our application, we use the dialogue to remind the user whether they want to leave the go back to the login page. If they decide to go back, they have to login in again by a user id and password.

42. Typeface

The Typeface class specifies the typeface and intrinsic style of a font. This is used in the paint, along with optionally Paint settings like text size, textScaleX to specify how text appears when drawn (and measured).

In our application, we use typeface to change the style and size of text in order to make the app more beautiful.

43. AsyncTask

AsyncTask enables proper and easy use of the UI thread. This class allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.
AsyncTask is designed to be a helper class around thread and handler and does not constitute a generic threading framework. AsyncTasks should ideally be used for short operations

In our application, we use AsyncTask to update or insert the data into the mysql of service in case of the conflicting issues among threads.

44 Send the user to another App

One of Android's most important features is an app's ability to send the user to another app based on an "action" it would like to perform. For example, if your app has the address of a business that you'd like to show on a map, you don't have to build an activity in your app that shows a map. Instead, you can create a request to view the address using an Intent. The Android system then starts an app that's able to show the address on a map.

In our application, We use the intent and startActivity to jump from one activity to another. We also use Bundles to transfer one object from one activity to the other.

45. Context

Interface to global information about an application environment. This is an abstract class whose implementation is provided by the Android system. It allows access to application-specific resources and classes, as well as up-calls for application-level operations such as launching activities, broadcasting and receiving intents, etc.
In our application, We use this context interface to transfer the data in the Person class. We do not need to create a new object any more. By passing the local activity class into the context of person class, we can change the data in the Person class dynamically.

46 Implicit Intent

some implicit intents require "extra" data that provide different data types, such as a string. You can add one or more pieces of extra data using the variou put extra methods.
By default, the system determines the appropriate MIME type required by an intent based on the Uri data that's included. If you don't include a Uri in the intent, you should usually use setType to specify the type of data associated with the intent.
In our application, we use the method below to send the email to other team members

```
String[] mailAddress = new String[1];
Intent emailIntent = new Intent(Intent.ACTION_SEND);
emailIntent.setType(“plain/text”);
emailIntent.putExtra(Intent.EXTRA_EMAIL, mailAddress);
startActivity(emailIntent);
```

47. ScrowView

Layout container for a view hierarchy that can be scrolled by the user, allowing it to be larger than the physical display. A ScrollView is a Framelayout, meaning you should place one child in it containing the entire contents to scroll; this child may itself be a layout manager with a complex hierarchy of objects. A child that is often used is a Linearlayout in a vertical orientation, presenting a vertical array of top-level items that the user can scroll through.
In our application, we use the ScrowView in every layout except the Team layout, we have to use the linearlayout in order to achieve the listview. We should never use a ScrollView with a ListView, because ListView takes care of its own vertical scrolling.

48 Using toast to communicate with client is a good design. It exceeds alert dialog in two aspect. The one is that you could display a toast even if there is no exception or error, for example, in our app, we can provide a server just remind user how to use our music service. The other reason is that toast did not interrupt your UI operation. If you input something wrong and a alert dialog display, you have to press some button to close the dialog, but for toast, you could see what you do wrong, but you do not have to wait to press button, but redo it again is fine. This is user friendly.

49. Toast is independent from activity and fragment. In our application, toast may occur when you jump from one activity to another, which may mislead user. So in the onStop() method, we use cancel method of Toast class to stop showing toast for next activity.

50. Every activity can listen on a touch event. Implement an OnTouchListener interface and have onTouch method, activity could track your touch position.

51. Motion event is triggered every time you touch screen. Unlike common sense, when you touch down and up is treated as two motion event, which means that your click on page trigger two consecutive action, and two listener is instantiated to handle these two touch event.

52. Motion event contains the touch position data. For our project, we designed a gesture control based on this property. We track the position of touch down and touch up, and use the distance to judge if it triggered a customized gesture, then we use it to start and stop music service.