Rushabh Shah
CPRE 388
Section F

HW 3

1. Question 1 - Fragments
    a. What is the idea behind UI fragments?

➔ Fragments serve as a portion of UI in an activity. Fragment increases modularity in the application. In that, fragments can be added and removed while activity is running. Although a Fragment has its own life cycle, its life cycle also depends on its host activity's life cycle. Meaning when the host activity is stopped or killed, the fragments in that activity are also stopped and killed respectively. Fragment lives inside ViewGroup in the activity hierarchy.

    b. How does the use of fragments improve the modularity of UI design? Provide an illustration and use case where fragments are better for UI design.

➔ Fragments represent a portion of an activity, but not hardcoded inside an activity. Fragment has its own UI, callback functions and life cycle. Thus, same fragment can be used in other activities. This avoid the duplicate code.
One use case will be an email application on a tablet. Assume that this email application has a list view that shows a list of emails. And another view that shows details of an email.

IOWA STATE UNIVERSITY – PARKING DIVISION Parking permits are on sale now!	Click on an email to view details.
TRELLO Board summary for this week	
ALTERA – BETTER WITH INTEL	
STUDENT UNION BOARD Free Performance! Wednesday, October 26	

In the example above, we have the list of emails on the left side and detailed view on the right side. Both the list and detailed view can be implemented as a fragment. The detailed view will be reused for each email in the list. The list will be reused based on the category user has selected. (i.e. Inbox, Sent, Deleted, Drafts, etc.). These view fragments can be used in other activities.

    c. Explain how fragment life cycle is influenced by host activity's life cycle. Why do you think the relationship between the host and fragment is important?

➔ Fragments are embedded in host activity. Fragments live in a ViewGroup inside activity's hierarchy. Thus, when an activity is stopped or killed, all components residing inside the activity are killed, including the fragments. This is an important relationship because if the host activity is stopped, fragments should also be stopped. In the previous example if user navigates to a different activity, then the both fragments are also expected to be stopped.

    d. Using an example app, explain the purpose of using back stack to manage fragment transactions. Note – no need to write code, just think of an app UI that uses fragments.

➔ Few features in BlackBoard Learn application can be implemented using fragments. For example, the Announcement, Course Content, Send Email, Course Tools and My Grades sections can be implemented using fragment. Every running course will have these sections. Thus, using fragments will increase modularity in the application.

e. How does a fragment communicate with the host activity? Illustrate your answer using a block diagram.

➔ Fragments have callbacks that server as a mean of communication to host Activity. Host activity implements the callback interface to pass information. Following block diagram illustrates the how the communication happens.

2. Question 2 – Notification
   a. Explain how to preserve the User's expected app navigation experience when an app's activity is called from a notification. Note – Use only words to describe the steps, do not write code here. You may use figures to describe the steps.

   ➔ By default, android framework uses the activity hierarchy to determine navigation after user lands on an activity from notification, say for instance, user clicks on an email and is taken to the detail activity in email application. Now pressing back button would take User to the parent activity of the application and then back to home screen.

   b. How does PendingIntent help in invoking an app activity from a notification?

   ➔ Pending Intent is a wrapper for Intent Object. Say for instance, an Email notification has arrived, but user chooses to interact with it later. Pending Intent is useful here. Because explicit intents are recommended method for using Pending Intent, we can specify the exact activity that should be launched when user clicks on the notification. In the example, user will be taken to detail activity.

3. Question 3 - Threads
   a. What does a thread safety mean with respect to Android UI toolkit?

   ➔ Thread safety is used to assure that the implementation will be free from race-conditions. (i.e. When multiple threads attempt to manipulate the shared data simultaneously, implementation is guaranteed to be free of race-condition.). According to documentation, the Android UI toolkit is NOT thread safe. Meaning, when multiple threads try to simultaneously manipulate UI, results will be unpredictable.

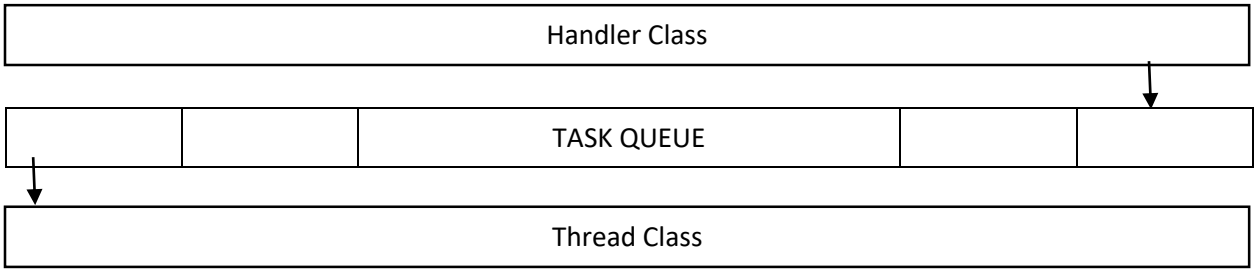
   b. How does Android API design implement thread safety for UI toolkit? Explain using the POST method of a View object.

   ➔ With two simple rules:
      - Do not block the UI thread.
      - Do not access the Android UI toolkit from outside the IU thread

   Android framework uses looper to manage runnable that will be run on UI thread. POST method of View object adds the runnable, given as a parameter, to the looper queue.

   c. Explain the concept behind Handler, Looper from thread safety perspective. Show the relation between handler, looper and the associated thread using a figure.

   ➔ Handler: Handler class manages the tasks in thread's message queue. Handler is associated with only one thread in which it was originally it was created. It delivers and processes tasks from message queue.
   Looper: Looper class is used to run a message loop for thread.
   Thread: Thread class is where the task is performed.
   This arrangement of queue ensures that simultaneous attempts are not made to update UI.



d. Thread pools are used to balance resources for better UI response. Provide arguments from Android framework and hardware perspective which support this statement.

➔ App users have a very short time span before they lose focus on the application. Thus fast UI response is necessary for good user experience. Use of thread pool allows android framework to do parallel execution. Running one job on each core, assuming hardware is multi-processor. This improves the computational time and as a result provides better UI response.