Luke Johnson

Lab 4

CS 555

**Synthesis questions:**

**1. What are major differences between Android Tool Bars, Action Bars and Title Bars?**

Title bars are more of a legacy feature. Their purpose was to show a title. They have been superseded by the actionbar, which provides more functionality. The toolbar class in the support library allows older versions of android to use the latest features of the action bar.

**2. What is the overflow menu?**

The overflow menu contains menu items that are maybe less commonly used, or would not normally fit as an item on the action bar.

**3. What is the | in the showAsAction attribute?**

This pipe indicates that both options apply. In this case, withText can apply along with other options in the showAsAction attribute.

**4. How does Android deal with XML attribute values that it doesn’t recognize?**

Hopefully it fails to compile.

**5. The author recommends not using system built icons. What are his reasons? Do you agree? Why?**

System built icons can be different across platforms. Including these icons in the drawables for your app increases the portability of your app. This comes at a cost of increasing the size of the app.

**6. What are two places that long presses are used?**

Long presses are used to bring up context menus, or other custom options. For example, I could set an onLongClickListener on an itemView in the CrimeHolder, in order to bring up a dialog option to delete a crime (After modifying CrimeHolder to implement View.OnLongClickListener)

@Override

public boolean onLongClick(View v) {

new AlertDialog.Builder(getActivity())

.setTitle(R.string.*delete\_crime*)

.setMessage(R.string.*confirm\_delete*)

.setPositiveButton(android.R.string.*ok*, new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

CrimeLab.*get*(getActivity()).deleteCrime(mCrime);

updateUI();

updateSubtitle();

dialog.cancel();

}

})

.setNegativeButton(android.R.string.*cancel*, new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

dialog.cancel();

}

}).create().show();

return true;

}

**7. What is the purpose of the DatabaseHelper class?**

The DatabaseHelper makes creating and maintaining the database much easier.

**8. List the steps to create and process an options menu.**

In onCreate, we have to call setHasOptionsMenu(true), to indicate that there is, in fact, an options menu.

We get to make some menu xmls. In our example, they are named the same as the fragments in which they will appear. That makes some sense, and it doesn’t cause a conflict because they are located in a different directory. These are fortunately simple xmls, and just list out items and their attributes. Make sure to set an icon. This can be in the drawable folder, or draw from the android system resources.

Finally, you have to implement onOptionsItemSelected, in order to specify what should happen when the item is selected.

**9. What are differences between ancestral (hierarchical) navigation and temporal navigation? Give an example of when each would be appropriate.**

Ancestral navigation Brings a user up the app hierarchy, to a different activity, and clears out the stack in between. Temporal navigation brings a user to the previous item on the stack.

**10. Why does the author recommend not putting the uuidString directly into the SQL where clause when adding or updating tables?**

Sanitize your inputs! This helps to prevent SQL injection issues.

**11. List a couple reasons why hierarchical navigation is superior to temporal navigation.**

Hierarchical navigation allows a user to return to a screen they want without having to hit the back button multiple times. It also could allow us to navigate out of an activity that no longer applies based on user actions. There are many ways this could be used.

**12. What does the finish() method do?**

This finishes and closes the current activity.

**13. Define the term: context menu. What are API considerations associated with these.**

The context menu is any menu that opens with a long press on a view. The contents of the menu should change based on the action the user wishes.

**14. When is a notifyDataSetChanged() method call is necessary?**

It is necessary to call notifyDataSetChanged() on return to the CrimeListFragment from CrimeFragment. Items could have been deleted, added, or modified, and that change needs to be reflected on the screen.

**15. Why was the call to invalidateOptionMenu necessary?**

The name of the menu item changes. This method tells Android to update the options menu. It destroys the options menu, and calls onCreateOptionsMenu, which updates the menu based on any new information. The other option is to override onPrepareOptionsMenu() and manipulate it in there.

**16. What is the null column hack?**

This allows you to put in a null value into a column that would normally require some value. It lets you specify a column in which a null column may be accepted.

**17. Define the terms: Graceful Fallback, Duplicate Replacement, and Duplicate on Demand. Which do you prefer? Why?**

These sound like things I didn’t do.

**18. In the multiple item delete implementation, the for loop counts down instead of up. Why is this?**

Assuming we’re talking about databases, you would want the loop to count down because you are

**19. What is the major purpose of the CursorWrapper?**

The cursorWrapper makes reading from the database much nicer. The wrapper is constructed with a cursor, and prevents repeating code.

**20. What is a major difference between the application context and an activity’s context?**

Application context applies for the whole application, while activity context applies only to the activity.

**Challenges:**

**Delete!**

I actually did this before I read the challenge section. I added a delete menu item in the crime fragment, and a deleteCrime method in CrimeLab. Additionally, I had the menu item bring up a dialog to confirm that the user wishes to delete the crime.

CrimeFragment.java

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch(item.getItemId()) {

case R.id.*menu\_item\_delete\_crime*:

new AlertDialog.Builder(getActivity())

.setTitle(R.string.*delete\_crime*)

.setMessage(R.string.*confirm\_delete*)

.setPositiveButton(android.R.string.*ok*, new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

CrimeLab.*get*(getActivity()).deleteCrime(mCrime);

getActivity().finish();

dialog.cancel();

}

})

.setNegativeButton(android.R.string.*cancel*, new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

dialog.cancel();

}

}).create().show();

return true;

default:

return super.onOptionsItemSelected(item);

}

}

@Override

public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {

super.onCreateOptionsMenu(menu, inflater);

inflater.inflate(R.menu.*fragment\_crime*, menu);

}

**CrimeLab.java**

public void deleteCrime(Crime c) {

String uuidString = c.getId().toString();

mDatabase.delete(CrimeTable.*NAME*, CrimeTable.Cols.*UUID* + " = ?",

new String[] {uuidString});

}