Week 3 - ACTIVITIES - LIFECYCLES AND INSTANCE STATE

Terms in this set (16)

List the callbacks (lifecycle and other) in sequence that occur for a reorientation event	onPause, onSaveInstanceState, onStop, onDestroy, onCreate, onStart, onRestoreInstanceState, onResume
List the callbacks (lifecycle and other) in sequence that occur for a back even	onPause, onStop, onDestroy

What is the difference between a) and b) up to and including destroy?	With reorientation there is there is an execution of onSaveInstanceState, which doesn't occur when the user hits the back button

Week 3 - ACTIVITIES - LIFECYCLES AND INSTANCE STATE

Study

When does an inherited callback's super execute if its: a) Not overridden?	the latest "version" of the method is called directly rather than going through the method overriding it (I think this is badly worded but I'll leave it here for now)
When does an inherited callback's super execute if its: Overridden?	When the method overriding the previous implementation of it decides to call the previous implementation (in a constructor, this happens at the top but here it is most likely to happen at the bottom if the callback has not handled the event or further handling is required). OnSaveInstanceState and onRestoreInstanceState are both callback methods inherited by Activity sub classes.
With respect to saving view state and instance variable state of an Activity sub class instance, explain the consequences of not overriding and overriding these methods (3 possible scenarios)	In onSaveInstanceState and onRestoreInstanceState, not calling the super implementation will have the consequence of not saving the view state (Hierarchy) of all Views with IDs. (should always call super method in order to save and restore view hierarchy) This isn't necessarily a dangerous consequence since you could implement this on your own anyways
What are SharedPreferences files for?	Used for saving key value pairs into persistent memory, so that it can be recovered later on even if we go through a back event. - Used to store information in a Preferences file that can be shared across activities (this file is saved on device storage and is therefore persistent, it's saved as .xml Used to store relatively small key value pairs

Week 3 - ACTIVITIES - LIFECYCLES AND INSTANCE STATE

Study

Should SharedPreferences be used directly to save app settings? If not what should be used?	- SharedPreferences have been created for the purpose of saving settings and are generally good practice, however, you can also save the data into a text file or into a database depending on what the information is used for (SharedPreferences is generally better for settings whilst others are better for data, hence the name "SharedPreferences")
c) What's the difference between what is referenced by the return values of getPreferences(...) and getSharedPreferences(...)?	getSharedPreferences() : Use this if you need multiple preferences files identified by name, which you specify with the first parameter. Extra Note: getSharedPreferences can be accessed by other activities - getPreferences(...) : Use this if you need only one preferences file for your Activity. Because this will be the only preferences file for your Activity, you don't supply a name. Extra Note: getPreferences is activity dependent
er the following statement: - ıScore = ef.getInt(getString(R.string.saved_high_score),); s someVal?	It is a key value pair. The R.String.... is the key and the someVal is the value. (Not right) - someVal is the default value to return if null (nothing has been saved with that key) - If saved_high_score is empty or hasn't been assigned a value, the someVal is the default value that will be returned.

Week 3 - ACTIVITIES - LIFECYCLES AND INSTANCE STATE

Study

er the following statement: - hScore = ef.getInt(getString(R.string.saved_high_score),); does getString do exactly?	- Retrieve the String that has been saved under that name in the strings.xml resource file - Inside the strings.xml resource file there is a key instantiated (saved_high_score), this key has to be retrieved and the getString method is used to retrieve this value.
the following statement: - hScore = ef.getInt(getString(R.string.saved_high_score),); e does getString's return value play as a er of the getInt method?	getInt retrieves the int value from the preferences, and getString is nested in getString. - getString gets the String value from the strings.xml which in this case is being used as the key to which the value expected is supposedly bound (since it may return null if the value saved is null or nothing has been saved under that key, I used the word "supposedly") - objects stored in resources are saved under an ID value. This ID is an integer. The getString() method identifies the name of the string resource saved under the ID value retrieved by getInt(). - Basically the getString() method is used to identify which integer value to return from the getInt().

Week 3 - ACTIVITIES - LIFECYCLES AND INSTANCE STATE

Study

Why is there a difference between the callback of a back event vs a reorientation event	is done with the activity and its current data, so an onSaveInstanceState callback is called to allow coders to save the instance state data in a Bundle object (for non-view state data, it can be coded to perform manual save). In this method, Android can also call super to save view-state data when the method is overridden. Back: Android DOES take this as a sign that the user is done with the activity and its current data, so no opportunity to save any kind of data is provided
Describe 4 tasks that code in an Activity's onCreate callback usually does.	Activity enters the Created state (public void onCreate(Bundle saveInstanceState) - You perform basic application startup logic that should happen only once for the entire life of the activity. (super.Oncreate) - The Activity has its view created (setContentView) - This method receives the parameter saveInstanceState, which is a Bundle object containing the activities previously saved state. If the activity has never existed before, the value of the Bundle object is null super.onCreate: here we are calling the super of onCreate to complete the creation of activity like the view hierarchy. - saveInstanceState: this recovers the instance state - setContentView: this defines the user interface that will be displayed to the user upon onCreate which is defined in the XML layout file. - mTextView : initializes the member TextView so that we can manipulate it later

Week 3 - ACTIVITIES - LIFECYCLES AND INSTANCE STATE

Study

Name the method used to accomplish the task where possible	setContentView(R.layout.main)) 3. restores from the bundle 4. recovers the saved instance state. (usually via an if(savedInstanceState != null){
b) Should they be used directly to save app settings? a) If not what should be used?	Preference API's use SharedPreferences as their implementation to save app settings. - SharedPreferences have been created for the purpose of saving settings and are generally good practice, however, you can also save the data into a text file or into a database depending on what the information is used for (SharedPreferences is generally better for settings whilst others are better for data, hence the name "SharedPreferences")