[Android fundamentals 01.1: Android Studio and Hello World](https://developer.android.com/codelabs/android-training-hello-world?index=..%2F..%2Fandroid-training)

[10. Summary](https://developer.android.com/codelabs/android-training-hello-world?index=..%2F..%2Fandroid-training#9)

* To install Android Studio, go to [Android Studio](https://developer.android.com/sdk/index.html) and follow the instructions to download and install it.
* When creating a new app, ensure that **API 15: Android 4.0.3 IceCreamSandwich** is set as the Minimum SDK.
* To see the app's Android hierarchy in the Project pane, click the **Project** tab in the vertical tab column, and then choose **Android** in the popup menu at the top.
* Edit the build.gradle(Module:app) file when you need to add new libraries to your project or change library versions.
* All code and resources for the app are located within the app and res folders. The java folder includes activities, tests, and other components in Java source code. The res folder holds resources, such as layouts, strings, and images.
* Edit the AndroidManifest.xml file to add features components and permissions to your Android app. All components for an app, such as multiple activities, must be declared in this XML file.
* Use the [Android Virtual Device (AVD) manager](http://developer.android.com/tools/devices/managing-avds.html) to create a virtual device (also known as an emulator) to run your app.
* Add [Log](https://developer.android.com/reference/android/util/Log.html) statements to your app, which display messages in the Logcat pane as a basic tool for debugging.
* To run your app on a physical Android device using Android Studio, turn on USB Debugging on the device. Open **Settings > About phone** and tap **Build number** seven times. Return to the previous screen (**Settings**), and tap **Developer options**. Choose **USB Debugging**.

**Question 1**

What is the name of the layout file for the main activity?

* MainActivity.java
* AndroidManifest.xml
* activity\_main.xml
* build.gradle

**Question 2**

What is the name of the string resource that specifies the application's name?

* app\_name
* xmlns:app
* android:name
* applicationId

**Question 3**

Which tool do you use to create a new emulator?

* Android Device Monitor
* AVD Manager
* SDK Manager
* Theme Editor

**Question 4**

Assume that your app includes this logging statement:

Log.i("MainActivity", "MainActivity layout is complete");

You see the statement "MainActivity layout is complete" in the **Logcat** pane if the Log level menu is set to which of the following? (Hint: multiple answers are OK.)

* Verbose
* Debug
* Info
* Warn
* Error
* Assert

[Android fundamentals 01.2 Part B: The layout editor](https://developer.android.com/codelabs/android-training-layout-editor-part-b?index=..%2F..%2Fandroid-training)

[7. Summary](https://developer.android.com/codelabs/android-training-layout-editor-part-b?index=..%2F..%2Fandroid-training#6)

Using the layout editor to preview and create variants:

* To preview the app layout with a horizontal orientation in the layout editor, click the **Orientation in Editor** button  in the top toolbar and choose **Switch to Landscape**. Choose **Switch to Portrait** to return to vertical orientation.
* To create variant of the layout that is different for a horizontal orientation, click the **Orientation in Editor** button and choose **Create Landscape Variation**. A new editor window opens with the **land/activity\_main.xml** tab showing the layout for the landscape (horizontal) orientation.
* To preview the layout for different devices without having to run the app on the device or emulator, click the **Device in Editor** button  in the top toolbar, and choose a device.
* To create variant of the layout that is different for a tablet (larger screen), click the **Orientation in Editor** button and choose **Create layout x-large Variation**. A new editor window opens with the **xlarge/activity\_main.xml** tab showing the layout for a tablet-sized device.

Using ConstraintLayout:

* To clear all constraints in a layout with the ConstraintLayout root, click the **Clear All Constraints** button in the toolbar.
* You can align one UI element that contains text, such as a TextView or Button, with another UI element that contains text. A *baseline constraint* lets you constrain the elements so that the text baselines match.
* To create a baseline constraint, hover your pointer over the UI element until the baseline constraint button  appears underneath the element.
* The pack button  in the toolbar provides options for packing or expanding selected UI elements. You can use it to equally arrange the Button elements horizontally across the layout.

Using LinearLayout:

* [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout.html) is a [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html" \t "_blank) that arranges its collection of views in a horizontal or vertical row.
* A LinearLayout is required to have the layout\_width, layout\_height, and orientation attributes.
* match\_parent for layout\_width or layout\_height: Expands the View to fill its parent by width or height. When the LinearLayout is the root View, it expands to the size of the screen (the parent View).
* Wrap\_content for layout\_width or layout\_height: Shrinks the dimensions so the View is just big enough to enclose its content. If there is no content, the View becomes invisible.
* Fixed number of dp ( [density-independent pixels](https://developer.android.com/training/multiscreen/screendensities.html)) for layout\_width or layout\_height: Specify a fixed size, adjusted for the screen density of the device. For example, 16dp means 16 density-independent pixels.
* The orientation for a LinearLayout can be horizontal to arrange elements from left to right, or vertical to arrange elements from top to bottom.
* Specifying gravity and weight attributes gives you additional control over arranging views and content in a LinearLayout.
* The android:gravity attribute specifies the alignment of the content of a View within the View itself.
* The android:layout\_weight attribute indicates how much of the extra space in the LinearLayout will be allocated to the View. If only one View has this attribute, it gets all the extra screen space. For multiple View elements, the space is prorated. For example, if two Button elements each have a weight of 1 and a TextView 2, totaling 4, the Button elements get ¼ of the space each, and the TextView half.

Using RelativeLayout:

* A [RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html" \t "_blank) is a ViewGroup in which each view is positioned and aligned relative to other views within the group.
* Use android:layout\_alignParentTop to align the View to the top of the parent.
* Use android:layout\_alignParentLeft to align the View to the left side of the parent.
* Use android:layout\_alignParentStart to make the start edge of the View match the start edge of the parent. This attribute is useful if you want your app to work on devices that use different language or locale preferences. The *start* is the left edge of the screen if the preference is left-to-right, or it is the right edge of the screen if the preference is right-to-left.

**Question 1**

Which two layout constraint attributes on the **Zero**Button position it vertically equal distance between the other two Button elements? (Pick 2 answers.)

* app:layout\_constraintBottom\_toTopOf="@+id/button\_count"
* android:layout\_marginBottom="8dp"
* android:layout\_marginStart="16dp"
* app:layout\_constraintTop\_toBottomOf="@+id/button\_toast"
* android:layout\_marginTop="8dp"

**Question 2**

Which layout constraint attribute on the **Zero**Button positions it horizontally in alignment with the other two Button elements?

* app:layout\_constraintLeft\_toLeftOf="parent"
* app:layout\_constraintBottom\_toTopOf="@+id/button\_count"
* android:layout\_marginBottom="8dp"
* app:layout\_constraintTop\_toBottomOf="@+id/button\_toast"

**Question 3**

What is the correct signature for a method used with the android:onClick XML attribute?

* public void callMethod()
* public void callMethod(View view)
* private void callMethod(View view)
* public boolean callMethod(View view)

**Question 4**

The click handler for the **Count**Button starts with the following method signature:

public void countUp(View view)

Which of the following techniques is more efficient to use within this handler to change the Button element's background color? Choose one:

* Use findViewById to find the **Count**Button. Assign the result to a View variable, and then use [setBackgroundColor()](https://developer.android.com/reference/android/view/View.html" \l "setBackgroundColor(int)" \t "_blank).
* Use the view parameter that is passed to the click handler with [setBackgroundColor()](https://developer.android.com/reference/android/view/View.html" \l "setBackgroundColor(int)" \t "_blank): view.setBackgroundColor()

[Android fundamentals 02.1: Activities and intents](https://developer.android.com/codelabs/android-training-create-an-activity?index=..%2F..%2Fandroid-training)

[8. Summary](https://developer.android.com/codelabs/android-training-text-and-scrolling-views?index=..%2F..%2Fandroid-training#7)

* Use a [ScrollView](https://developer.android.com/reference/android/widget/ScrollView.html" \t "_blank) to scroll a single child View (such as a TextView). A ScrollView can hold only one child View or ViewGroup.
* Use a ViewGroup such as [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout.html" \t "_blank) as a child View within a ScrollView to scroll more than one View element. Enclose the elements within the LinearLayout.
* Display free-form text in a TextView with HTML formatting tags for bold and italics.
* Use \n as an end-of-line character in free-form text to keep a paragraph from running into the next paragraph.
* Use the android:autoLink="web" attribute to make web links in the text clickable.

**Question 1**

What changes are made when you add a second Activity to your app by choosing **File > New > Activity** and an Activity template? Choose one:

* The second Activity is added as a Java class. You still need to add the XML layout file.
* The second Activity XML layout file is created and a Java class added. You still need to define the class signature.
* The second Activity is added as a Java class, the XML layout file is created, and the AndroidManifest.xml file is changed to declare a second Activity.
* The second Activity XML layout file is created, and the AndroidManifest.xml file is changed to declare a second Activity.

**Question 2**

What happens if you remove the android:parentActivityName and the <meta-data> elements from the second Activity declaration in the AndroidManifest.xml file? Choose one:

* The second Activity no longer appears when you try to start it with an explicit Intent.
* The second Activity XML layout file is deleted.
* The Back button no longer works in the second Activity to send the user back to the main Activity.
* The Up button in the app bar no longer appears in the second Activity to send the user back to the parent Activity.

**Question 3**

Which constructor method do you use to create a new explicit Intent? Choose one:

* new Intent()
* new Intent(Context context, Class<?> class)
* new Intent(String action, Uri uri)
* new Intent(String action)

**Question 4**

In the HelloToast app homework, how do you add the current value of the count to the Intent? Choose one:

* As the Intent data
* As the Intent TEXT\_REQUEST
* As an Intent action
* As an Intent extra

**Question 5**

In the HelloToast app homework, how do you display the current count in the second "Hello" Activity? Choose one:

* Get the Intent that the Activity was launched with.
* Get the current count value out of the Intent.
* Update the TextView for the count.
* All of the above.

[Android fundamentals 02.2: Activity lifecycle and state](https://developer.android.com/codelabs/android-training-activity-lifecycle-and-state?index=..%2F..%2Fandroid-training)

[9. Summary](https://developer.android.com/codelabs/android-training-create-an-activity?index=..%2F..%2Fandroid-training#8)

Overview:

* An Activity is an app component that provides a single screen focused on a single user task.
* Each Activity has its own user interface layout file.
* You can assign your Activity implementations a parent/child relationship to enable Up navigation within your app.
* A View can be made visible or invisible with the android:visibility attribute.

To implement an Activity:

* Choose **File > New > Activity** to start from a template and do the following steps automatically.
* If not starting from a template, create an Activity Java class, implement a basic UI for the Activity in an associated XML layout file, and declare the new Activity in AndroidManifest.xml.

Intent:

* An Intent lets you request an action from another component in your app, for example, to start one Activity from another. An Intent can be explicit or implicit.
* With an explicit Intent you indicate the specific target component to receive the data.
* With an implicit Intent you specify the functionality you want but not the target component.
* An Intent can include data on which to perform an action (as a URI) or additional information as Intent *extras*.
* Intent *extras* are key/value pairs in a Bundle that are sent along with the Intent.

**Question 1**

If you run the homework app before implementing onSaveInstanceState(), what happens if you rotate the device? Choose one:

* The EditText no longer contains the text you entered, but the counter is preserved.
* The counter is reset to 0, and the EditText no longer contains the text you entered.
* The counter is reset to 0, but the contents of the EditText is preserved.
* The counter and the contents of the EditText are preserved.

**Question 2**

What Activity lifecycle methods are called when a device-configuration change (such as rotation) occurs? Choose one:

* Android immediately shuts down your Activity by calling onStop(). Your code must restart the Activity.
* Android shuts down your Activity by calling onPause(), onStop(), and onDestroy(). Your code must restart the Activity.
* Android shuts down your Activity by calling onPause(), onStop(), and onDestroy(), and then starts it over again, calling onCreate(), onStart(), and onResume().
* Android immediately calls onResume().

**Question 3**

When in the Activity lifecycle is onSaveInstanceState() called? Choose one:

* onSaveInstanceState() is called before the onStop() method.
* onSaveInstanceState() is called before the onResume() method.
* onSaveInstanceState() is called before the onCreate() method.
* onSaveInstanceState() is called before the onDestroy() method.

**Question 4**

Which Activity lifecycle methods are best to use for saving data before the Activity is finished or destroyed? Choose one:

* onPause() or onStop()
* onResume() or onCreate()
* onDestroy()
* onStart() or onRestart()

[Android fundamentals 02.3: Implicit intents](https://developer.android.com/codelabs/android-training-activity-with-implicit-intent?index=..%2F..%2Fandroid-training)

[9. Summary](https://developer.android.com/codelabs/android-training-activity-with-implicit-intent?index=..%2F..%2Fandroid-training#8)

* An implicit Intent allows you to activate an Activity if you know the action, but not the specific app or Activity that will handle that action.
* An Activity that can receive an implicit Intent must define Intent filters in the AndroidManifest.xml file that match one or more Intent actions and categories.
* The Android system matches the content of an implicit Intent and the Intent filters of any available Activity to determine which Activity to activate. If there is more than one available Activity, the system provides a chooser so the user can pick one.
* The ShareCompat.IntentBuilder class makes it easy to build an implicit Intent for sharing data to social media or email.

**Question 1**

Which constructor method do you use to create an implicit Intent to launch a camera app?

* new Intent()
* new Intent(Context context, Class<?> class)
* new Intent(String action, Uri uri)
* new Intent(String action)

**Question 2**

When you create an implicit Intent object, which of the following is true?

* Don't specify the specific Activity or other component to launch.
* Add an Intent action or Intent categories (or both).
* Resolve the Intent with the system before calling startActivity() or startActivityforResult().
* All of the above.

**Question 3**

Which Intent action do you use to take a picture with a camera app?

* Intent takePicture = new Intent(Intent.ACTION\_VIEW);
* Intent takePicture = new Intent(Intent.ACTION\_MAIN);
* Intent takePicture = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);
* Intent takePicture = new Intent(Intent.ACTION\_GET\_CONTENT);

[Android fundamentals 04.1: Clickable images](https://developer.android.com/codelabs/android-training-clickable-images?index=..%2F..%2Fandroid-training)

[7. Summary](https://developer.android.com/codelabs/android-training-clickable-images?index=..%2F..%2Fandroid-training#6)

* To use an image in a project, copy the image into the project's **drawable** folder ( *project\_name***> app > src > main > res > drawable**).
* Define an ImageView to use it by dragging an ImageView to the layout and choosing the image for it.
* Add the android:onClick attribute to make an ImageView clickable like a button. Specify the name of the click handler.
* Create a click handler in the Activity to perform the action.
* Choose an icon: Expand **res** in the **Project > Android** pane, right-click (or Control-click) the **drawable** folder, and choose **New > Image Asset**. Choose **Action Bar and Tab Icons** in the drop-down menu, and click the clip art image (the Android logo next to **Clipart:**) to select a clip art image as the icon.
* Add another Activity: In the **Project > Android** pane, right-click (or Control-click) the package name folder within the **java** folder and choose **New > Activity** and a template for the Activity (such as **Empty Activity**).
* Display a [Toast](https://developer.android.com/reference/android/widget/Toast.html) message:

Toast.makeText(getApplicationContext(), message,   
                                          Toast.LENGTH\_SHORT).show();

Question 1

How do you add images to an Android Studio project? Choose one:

* Drag each image to the layout editor.
* Copy the image files into your project's drawable folder.
* Drag an ImageButton to the layout editor.
* Choose **New > Image Asset** and then choose the image file.

Question 2

How do you make an ImageView clickable like a simple Button? Choose one:

* Add the android:contentDescription attribute to the ImageView in the layout and use it to call the click handler in the Activity.
* Add the android:src attribute to the ImageView in the layout and use it to call the click handler in the Activity.
* Add the android:onClick attribute to the ImageView in the layout and use it to call the click handler in the Activity.
* Add the android:id attribute to the ImageView in the layout and use it to call the click handler in the Activity.

Question 3

Which rule applies to a click handler called from the attribute in the layout? Choose one:

* The click handler method must include the event listener View.OnClickListener, which is an interface in the View class .
* The click handler method must be public, return void, and define a View as its only parameter.
* The click handler must customize the View.OnClickListener class and override its click handler to perform some action.
* The click handler method must be private and return a View.

[Android fundamentals 04.2: Input controls](https://developer.android.com/codelabs/android-training-input-controls?index=..%2F..%2Fandroid-training)

[8. Summary](https://developer.android.com/codelabs/android-training-input-controls?index=..%2F..%2Fandroid-training#7)

The following android:inputType attribute values affect the appearance of the on-screen keyboard:

* textAutoCorrect: Suggest spelling corrections.
* textCapSentences: Start each new sentence with a capital letter.
* textPersonName: Show a single line of text with suggestions as the user types, and the **Done** key for the user to tap when they're finished.
* textMultiLine: Enable multiple lines of text entry and a Return key to add a new line.
* textPassword: Hide a password when entering it.
* textEmailAddress: Show an email keyboard rather than a standard keyboard.
* phone: Show a phone keypad rather than a standard keyboard.

You set values for the android:inputType attribute in the XML layout file for an EditText element To combine values, concatenate them using the pipe (|) character.

Radio buttons are input controls that are useful for selecting only one option from a set of options:

* Group [RadioButton](https://developer.android.com/reference/android/widget/RadioButton.html" \t "_blank) elements together inside a [RadioGroup](https://developer.android.com/reference/android/widget/RadioGroup.html" \t "_blank) so that only one [RadioButton](https://developer.android.com/reference/android/widget/RadioButton.html" \t "_blank) can be selected at a time.
* The order in which you list the RadioButton elements in the group determines the order that they appear on the screen.
* Use the android:onClick attribute for each RadioButton to specify the click handler.
* To find out if a button is selected, use the [isChecked()](https://developer.android.com/reference/android/widget/Checkable.html" \l "isChecked()" \t "_blank) method of the [Checkable](https://developer.android.com/reference/android/widget/Checkable.html) interface, which returns true if the button is selected.

A [Spinner](https://developer.android.com/reference/android/widget/Spinner.html) provides a drop-down menu:

* Add a [Spinner](https://developer.android.com/reference/android/widget/Spinner.html) to the layout.
* Use an [ArrayAdapter](https://developer.android.com/reference/android/widget/ArrayAdapter.html" \t "_blank) to assign an array of text values as the Spinner menu items.
* Implement the [AdapterView.OnItemSelectedListener](https://developer.android.com/reference/android/widget/AdapterView.OnItemSelectedListener.html" \t "_blank) interface, which requires also adding the onItemSelected() and onNothingSelected() callback methods to activate the Spinner and its listener.
* Use the [onItemSelected()](https://developer.android.com/reference/android/widget/AdapterView.OnItemSelectedListener.html" \l "onItemSelected(android.widget.AdapterView%3C?%3E,%20android.view.View,%20int,%20long)" \t "_blank) callback method to retrieve the selected item in the Spinner menu using [getItemAtPosition()](https://developer.android.com/reference/android/widget/AdapterView.html" \l "getItemAtPosition(int)" \t "_blank).

Question 1

What's the most important difference between checkboxes and a RadioGroup of radio buttons? Choose one:

* The only difference is in how they appear: checkboxes show a checkmark when selected, while circular "radio" buttons appear filled when selected.
* CheckBox elements in the layout can use the android:onClick attribute to call a handler when selected.
* The major difference is that checkboxes enable multiple selections, while a RadioGroup allows only one selection.

Question 2

Which layout group lets you align a set of CheckBox elements vertically? Choose one:

* RelativeLayout
* LinearLayout
* ScrollView

Question 3

Which of the following is the method of the [Checkable](https://developer.android.com/reference/android/widget/Checkable.html) interface to check the state of a radio button (that is, whether it has been selected or not)?

* getId()
* isChecked()
* onRadioButtonClicked()
* onClick()

[Android fundamentals 04.3: Menus and pickers](https://developer.android.com/codelabs/android-training-menus-and-pickers?index=..%2F..%2Fandroid-training)

[10. Summary](https://developer.android.com/codelabs/android-training-menus-and-pickers?index=..%2F..%2Fandroid-training#9)

Provide an options menu and app bar:

* Start your app or Activity with the Basic Activity template to automatically set up the app bar, the options menu, and a floating action button.
* The template sets up a [CoordinatorLayout](https://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.html" \t "_blank) layout with an embedded [AppBarLayout](https://developer.android.com/reference/android/support/design/widget/AppBarLayout.html" \t "_blank) layout. AppBarLayout is like a vertical LinearLayout. It uses the [Toolbar](https://developer.android.com/reference/android/support/v7/widget/Toolbar.html) class in the support library, instead of the native ActionBar, to implement an app bar.
* The template modifies the AndroidManifest.xml file so that the .MainActivity Activity is set to use the NoActionBar theme. This theme is defined in the styles.xml file.
* The template sets MainActivity to extend AppCompatActivity and starts with the onCreate() method, which sets the content view and Toolbar. It then calls [setSupportActionBar()](https://developer.android.com/reference/android/support/v7/app/AppCompatActivity.html" \l "setSupportActionBar(android.support.v7.widget.Toolbar)" \t "_blank) and passes toolbar to it, setting the toolbar as the app bar for the Activity.
* Define menu items in the menu\_main.xml file. The android:orderInCategory attribute specifies the order in which the menu items appear in the menu, with the lowest number appearing higher in the menu.
* Use the [onOptionsItemSelected``()](https://developer.android.com/reference/android/app/Activity.html" \l "onOptionsItemSelected(android.view.MenuItem)" \t "_blank) method to determine which menu item was tapped.

Add an icon for an options menu item:

* Expand **res** in the **Project > Android** pane, and right-click (or Control-click) the **drawable** folder. Choose **New > Image Asset**.
* Choose **Action Bar and Tab Items** in the drop-down menu, and change the name of the image file.
* Click the clip art image to select a clip art image as the icon. Choose an icon.
* Choose **HOLO\_DARK** from the **Theme** drop-down menu.

Show menu items as icons in the app bar:

* Use the app:showAsAction attribute in menu\_main.xml with the following values.
* "always": Always appears in the app bar. (If there isn't enough room it may overlap with other menu icons.)
* "ifRoom": Appears in the app bar if there is room.
* "never": Never appears in the app bar; its text appears in the overflow menu.

Use an alert dialog:

* Use a dialog to request a user's choice, such as an alert that requires users to tap **OK** or **Cancel**. Use dialogs sparingly as they interrupt the user's workflow.
* Use the [AlertDialog](https://developer.android.com/reference/android/app/AlertDialog.html" \t "_blank) subclass of the Dialog class to show a standard dialog for an alert.
* Use [AlertDialog.Builder](https://developer.android.com/reference/android/app/AlertDialog.Builder.html" \t "_blank) to build a standard alert dialog, with [setTitle()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html" \l "setTitle(int)" \t "_blank) to set its title, [setMessage()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html" \l "setMessage(int)" \t "_blank) to set its message, and [setPositiveButton()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html" \l "setPositiveButton(int,%20android.content.DialogInterface.OnClickListener)" \t "_blank) and [setNegativeButton()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html" \l "setNegativeButton(int,%20android.content.DialogInterface.OnClickListener)" \t "_blank) to set its buttons.

Use a picker for user input:

* Use [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html" \t "_blank), a subclass of [Fragment](https://developer.android.com/reference/android/support/v4/app/Fragment.html), to build a picker such as the date picker or time picker.
* Create a DialogFragment, and implement [DatePickerDialog.OnDateSetListener](https://developer.android.com/reference/android/app/DatePickerDialog.OnDateSetListener.html" \t "_blank) to create a standard date picker with a listener. Include [onDateSet()](https://developer.android.com/reference/android/app/DatePickerDialog.OnDateSetListener.html" \l "onDateSet(android.widget.DatePicker,%20int,%20int,%20int)" \t "_blank) in this Fragment.
* Replace the onCreateView() method with [onCreateDialog()](https://developer.android.com/reference/android/app/DialogFragment.html" \l "onCreateDialog(android.os.Bundle)" \t "_blank) that returns Dialog. Initialize the date for the date picker from [Calendar](https://developer.android.com/reference/java/util/Calendar.html), and return the dialog and these values to the Activity.
* Create an instance of [FragmentManager](https://developer.android.com/reference/android/app/FragmentManager.html" \t "_blank) using [getSupportFragmentManager()](https://developer.android.com/reference/android/support/v4/app/FragmentActivity.html" \l "getSupportFragmentManager()" \t "_blank) to manage the Fragment and show the date picker.

Question 1

What is the name of the file in which you create options menu items? Choose one:

* menu.java
* menu\_main.xml
* activity\_main.xml
* content\_main.xml

Question 2

Which method is called when an options menu item is clicked? Choose one:

* onOptionsItemSelected(MenuItem item)
* onClick(View view)
* onContextItemSelected()
* onClickShowAlert()

Question 3

Which of the following statements sets the title for an alert dialog? Choose one:

* myAlertBuilder.setMessage("Alert");
* myAlertBuilder.setPositiveButton("Alert");
* myAlertBuilder.setTitle("Alert");
* AlertDialog.Builder myAlertBuilder = new AlertDialog.Builder("Alert");

Question 4

Where do you create a DialogFragment for a date picker? Choose one:

* In the onCreate() method in the hosting Activity.
* In the onCreateContextMenu() method in Fragment.
* In the onCreateView() method in the extension of DialogFragment.
* In the onCreateDialog() method in the extension of DialogFragment.

[Android fundamentals 04.4: User navigation](https://developer.android.com/codelabs/android-training-provide-user-navigation?index=..%2F..%2Fandroid-training)

[6. Summary](https://developer.android.com/codelabs/android-training-provide-user-navigation?index=..%2F..%2Fandroid-training#5)

App bar navigation:

* Add Up-button navigation to a child Activity by declaring the parent Activity in the AndroidManifest.xml file.
* Declare the child's parent Activity within the child's <activity ... </activity> section:

android:parentActivityName=".MainActivity">  
<meta-data android:name="android.support.PARENT\_ACTIVITY"  
        android:value=".MainActivity"/>

Tab navigation:

* Tabs are a good solution for "lateral navigation" between sibling views.
* The primary class used for tabs is [TabLayout](https://developer.android.com/reference/android/support/design/widget/TabLayout.html" \t "_blank) in the Android Design Support Library.
* [ViewPager](https://developer.android.com/reference/android/support/v4/view/ViewPager.html) is a layout manager that allows the user to flip left and right through pages of data. ViewPager is most often used in conjunction with Fragment.
* Use one of the two standard adapters for using ViewPager: [FragmentPagerAdapter](https://developer.android.com/reference/android/support/v4/app/FragmentPagerAdapter.html) or [FragmentStatePagerAdapter](https://developer.android.com/reference/android/support/v4/app/FragmentStatePagerAdapter.html).

**Question 1**

Which template provides an Activity with an options menu and the [v7 appcompat](https://developer.android.com/tools/support-library/features.html#v7-appcompat) support library [Toolbar](https://developer.android.com/reference/android/support/v7/widget/Toolbar.html) as the app bar? Choose one:

* Empty Activity template
* Basic Activity template
* Navigation Drawer Activity template
* Bottom Navigation Activity

**Question 2**

Which dependency do you need in order to use a [TabLayout](https://developer.android.com/reference/android/support/design/widget/TabLayout.html" \t "_blank)? Choose one:

* com.android.support:design
* com.android.support.constraint:constraint-layout
* junit:junit:4.12
* com.android.support.test:runner

**Question 3**

Where do you define each child Activity and parent Activity to provide **Up** navigation? Choose one:

* To provide the **Up** button for a child screen Activity, declare the parent Activity in the child Activity section of the activity\_main.xml file.
* To provide the **Up** button for a child screen Activity, declare the parent Activity in the "main" XML layout file for the child screen Activity.
* To provide the **Up** button for a child screen Activity, declare the parent Activity in the child Activity section of the AndroidManifest.xml file.
* To provide the **Up** button for a child screen Activity, declare the parent Activity in the parent Activity section of the AndroidManifest.xml file.

[Android fundamentals 04.5: RecyclerView](https://developer.android.com/codelabs/android-training-create-recycler-view?index=..%2F..%2Fandroid-training)

[8. Summary](https://developer.android.com/codelabs/android-training-create-recycler-view?index=..%2F..%2Fandroid-training#7)

* [RecyclerView](https://developer.android.com/reference/android/support/v7/widget/RecyclerView.html) is a resource-efficient way to display a scrollable list of items.
* To create a View for each list item, the adapter inflates an XML layout resource for a list item using [LayoutInflator](http://developer.android.com/reference/android/view/LayoutInflater.html" \t "_blank).
* [LinearLayoutManager](https://developer.android.com/reference/android/support/v7/widget/LinearLayoutManager.html) is a RecyclerView layout manager that shows items in a vertical or horizontal scrolling list.
* [GridLayoutManager](https://developer.android.com/reference/android/support/v7/widget/GridLayoutManager.html) is a RecyclerView layout manager that shows items in a grid
* [StaggeredGridLayoutManager](https://developer.android.com/reference/android/support/v7/widget/StaggeredGridLayoutManager.html) is a RecyclerView layout manager that shows items in a staggered grid.
* Use [RecyclerView.Adapter](https://developer.android.com/reference/android/support/v7/widget/RecyclerView.Adapter.html" \t "_blank) to connect your data to the RecyclerView. It prepares the data in a [RecyclerView.ViewHolder](https://developer.android.com/reference/android/support/v7/widget/RecyclerView.ViewHolder.html" \t "_blank) that describes a View item and its position within the RecyclerView.
* Implement [View.onClickListener](https://developer.android.com/reference/android/view/View.OnClickListener.html" \t "_blank) to detect mouse clicks in a RecyclerView.

**Question 1**

Which of the following statements about a RecyclerView is *false*? Choose one.

* A RecyclerView is a more resource-efficient way to display scrollable lists.
* You need to provide a layout for just one item of the list.
* All list items look the same.
* You don't need a layout manager with a RecyclerView to handle the hierarchy and layout of View elements.

**Question 2**

Which of the following is the primary component you need to provide to an adapter a View item and its position within a RecyclerView? Choose one.

* RecyclerView
* RecyclerView.Adapter
* RecyclerView.ViewHolder
* AppCompatActivity

**Question 3**

Which interface do you need to implement in order to listen and respond to user clicks in a RecyclerView? Choose one.

* View.onClickListener
* RecyclerView.Adapter
* RecyclerView.ViewHolder
* View.OnKeyListener

[Android fundamentals 05.1: Drawables, styles, and themes](https://developer.android.com/codelabs/android-training-drawables-styles-and-themes?index=..%2F..%2Fandroid-training)

[9. Summary](https://developer.android.com/codelabs/android-training-drawables-styles-and-themes?index=..%2F..%2Fandroid-training#8)

* Drawable elements enhance the look of an app's UI.
* A [ShapeDrawable](https://developer.android.com/reference/android/graphics/drawable/ShapeDrawable.html" \t "_blank) is a primitive geometric shape defined in an XML file. The attributes that define a ShapeDrawable include color, shape, padding, and more.
* The Android platform supplies a large collection of styles and themes.
* Using styles can reduce the amount of code needed for your UI components.
* A style can specify common properties such as height, padding, font color, font size, and background color.
* A style should not include layout-related information.
* A style can be applied to a View, Activity, or the entire app. A style applied to an Activity or the entire app must be defined in the AndroidManifest.xml file.
* To inherit a style, a new style identifies a parent attribute in the XML.
* When you apply a style to a collection of View elements in an activity or in your entire app, that is known as a *theme*.
* To apply a theme, you use the android:theme attribute.

**Question 1**

Which type of Drawable do you use to create a Button with a background that stretches properly to accommodate the text or image inside the Button so that it looks correct for different screen sizes and orientations? Choose one:

* LevelListDrawable
* TransitionDrawable
* StateListDrawable
* NinePatchDrawable

**Question 2**

Which type of Drawable do you use to create a Button that shows one background when it is pressed and a different background when it is hovered over? Choose one:

* LevelListDrawable
* TransitionDrawable
* StateListDrawable
* NinePatchDrawable

**Question 3**

Suppose you want to create an app that has a white background, dark text, and a dark action bar. Which base style does your application style inherit from? Choose one:

* Theme.AppCompat.Light
* Theme.AppCompat.Dark.NoActionBar
* Theme.AppCompat.Light.DarkActionBar
* Theme.AppCompat.NoActionBar
* Theme.NoActionBar

[Android fundamentals 05.2: Cards and colors](https://developer.android.com/codelabs/android-training-cards-and-colors?index=..%2F..%2Fandroid-training)

[9. Summary](https://developer.android.com/codelabs/android-training-cards-and-colors?index=..%2F..%2Fandroid-training#8)

* A [CardView](https://developer.android.com/reference/android/support/v7/widget/CardView.html" \t "_blank) is a good layout to use for presenting information that has mixed media (such as images and text).
* CardView is a UI component found in the Android Support Library.
* CardView was *not* designed just for text child View elements.
* Loading images directly into an ImageView is memory intensive, because images are loaded at full resolution. To efficiently load images into your app, use an image loading library such as [Glide](https://github.com/bumptech/glide).
* The Android SDK has a class called [ItemTouchHelper](https://developer.android.com/reference/android/support/v7/widget/helper/ItemTouchHelper.html" \t "_blank) that helps your app get information about tap, swipe, and drag-and-drop events in your UI.
* A [FloatingActionButton](https://developer.android.com/reference/android/support/design/widget/FloatingActionButton.html" \t "_blank) (FAB) focuses the user on a particular action and "floats" in your UI.
* Material Design is a set of guiding principles for creating consistent, intuitive, and playful applications.
* According to Material Design, it's good practice to choose three colors for your app: a primary color, a primary dark color, and an accent color.
* Material Design promotes the use of bold imagery and colors to enhance user experience. It also promotes consistent elements across platforms, for example by using CardView and FAB widgets.
* Use Material Design to create meaningful, intuitive motion for UI elements such as cards that can be dismissed or rearranged.

**Question 1**

Which color attribute in your style defines the color of the status bar above the app bar? Choose one:

* colorPrimary
* colorPrimaryDark
* colorAccent
* colorAccentDark

**Question 2**

Which support library does the FloatingActionButton belong to? Choose one:

* v4 Support Library
* v7 Support Library
* Design Support Library
* Custom Button Support Library

**Question 3**

In the [Material Design color palette](https://material.google.com/style/color.html#color-color-palette), which shade of a color should you use as the primary color for your *brand* in your app? Choose one:

* Any color shade that starts with A.
* Any color shade labeled 200.
* Any color shade labeled 500.
* Any color shade labeled 900.

[Android fundamentals 05.3: Adaptive layouts](https://developer.android.com/codelabs/android-training-adaptive-layouts?index=..%2F..%2Fandroid-training)

[8. Summary](https://developer.android.com/codelabs/android-training-adaptive-layouts?index=..%2F..%2Fandroid-training#7)

* GridLayoutManager is a layout manager that handles two-dimensional scrolling lists.
* You can dynamically change the number of columns in a GridLayoutManager.
* The Android runtime uses alternative configuration files, depending on the runtime environment of the device running your app. For example, the runtime might use alternative configuration files for different device layouts, screen dimensions, locale, countries, or keyboard types.
* In your code, you create these alternative resources for the Android runtime to use. The resources are located in files that have resource qualifiers as part of their names.
* The format for a directory holding alternative resource files is *<resource\_name>*-*<qualifier>*.
* You can qualify any file in your res directory in this way.

**Question 1**

Which resource qualifier is used most frequently to select for tablets? Choose one:

* Orientation
* Screen width
* Screen height
* Smallest screen width

**Question 2**

Which folder would hold the strings.xml file for translation into French for Canada? Choose one:

* res/values-fr-rFR/
* res/values-ca-rFR/
* res/values-fr-rCA/
* res/values-en-rFR/

**Question 3**

Which folder is for XML files that contain strings, integers, and colors? Choose one:

* res/layout
* res/mipmap
* res/raw
* res/values

[Android fundamentals 07.1: AsyncTask](https://developer.android.com/codelabs/android-training-create-asynctask?index=..%2F..%2Fandroid-training)

[7. Summary](https://developer.android.com/codelabs/android-training-create-asynctask?index=..%2F..%2Fandroid-training#6)

* An [AsyncTask](http://developer.android.com/reference/android/os/AsyncTask.html" \t "_blank) is an abstract Java class that moves intensive processing onto a separate thread.
* AsyncTask must be subclassed to be used.
* Avoid doing resource-intensive work in the UI thread, because it can make your UI sluggish or erratic.
* Any code that does not involve drawing the UI or responding to user input should be moved from the UI thread to another, separate thread.
* AsyncTask has four key methods: [onPreExecute()](https://developer.android.com/reference/android/os/AsyncTask.html#onPreExecute()), [doInBackground()](https://developer.android.com/reference/android/os/AsyncTask.html#doInBackground(Params...)), [onPostExecute()](https://developer.android.com/reference/android/os/AsyncTask.html#onPostExecute(Result)) and [onProgressUpdate()](https://developer.android.com/reference/android/os/AsyncTask.html#onProgressUpdate(Progress...)).
* The doInBackground() method is the only method that is actually run on a worker thread. Do not call UI methods in the doInBackground() method.
* The other methods of AsyncTask run in the UI thread and allow you to call methods of UI components.
* Rotating an Android device destroys and recreates an Activity. This can disconnect the UI from the background thread in AsyncTask, which will continue to run.

**Question 1**

For a ProgressBar:

1. How do you determine the range of values that a ProgressBar can show?
2. How do you change how much of the progress bar is filled in?

**Question 2**

If an AsyncTask is defined as follows:

 private class DownloadFilesTask extends AsyncTask<URL, Integer, Long>

1. What is the type of the value that is passed to doInBackground() in the AsyncTask?
2. What is the type of the value that is passed to the callback that reports the progress of the task?
3. What is the type of the value that is passed to the callback that is executed when the task completes?

**Question 3**

To report progress of the work executed by an AsyncTask, what callback method do you *implement*, and what method do you *call* in your AsyncTask subclass?

* Implement publishProgress(). Call publishProgress().
* Implement publishProgress(). Call onProgressUpdate().
* Implement onProgressUpdate(). Call publishProgress().
* Implement onProgressUpdate(). Call onProgressUpdate().

[Android fundamentals 07.2: AsyncTask and AsyncTaskLoader](https://developer.android.com/codelabs/android-training-asynctask-asynctaskloader?index=..%2F..%2Fandroid-training)

[8. Summary](https://developer.android.com/codelabs/android-training-asynctask-asynctaskloader?index=..%2F..%2Fandroid-training#7)

* Tasks that connect to the network should not be executed on the UI thread. The Android runtime usually raises an exception if you attempt network connectivity or file access on the UI thread.
* Use the Books Search API to access Google Books programmatically. An API request to Google Books is in the form of a URL, and the response is a JSON string.
* Use the Google APIs Explorer to explore Google APIs interactively.
* Use getText() to retrieve text from an EditText view. To convert the text into a simple string, use toString().
* The Uri.buildUpon() method returns a URI.Builder that you can use to construct URI strings.
* To connect to the internet, you must configure network permission in the Android manifest file:

<uses-permission android:name="android.permission.INTERNET" />

The [AsyncTask](https://developer.android.com/reference/android/os/AsyncTask" \t "_blank) class lets you run tasks in the background instead of on the UI thread:

* To use an AsyncTask, you have to subclass it. The subclass overrides the doInBackground(Params...) method. Usually the subclass also overrides the onPostExecute(Result) method.
* To start an AsyncTask, use execute().
* An AsyncTask can't update the UI if the activity that the AsyncTask is controlling stops, for example because of a device-configuration change.

When an AsyncTask executes, it goes through four steps:

1. onPreExecute() runs on the UI thread before the task is executed. This step is normally used to set up the task, for instance by showing a progress bar in the UI.
2. doInBackground(Params...) runs on the background thread immediately after onPreExecute() finishes. This step performs background computations that can take a long time.
3. onProgressUpdate(Progress...) runs on the UI thread after you a call publishProgress(Progress...).
4. onPostExecute(Result) runs on the UI thread after the background computation is finished. The result of the computation is passed to onPostExecute().

[AsyncTaskLoader](https://developer.android.com/reference/android/support/v4/content/AsyncTaskLoader) is the loader equivalent of an AsyncTask.

* AsyncTaskLoader provides the loadInBackground() method, which runs on a separate thread.
* The results of loadInBackground() are delivered to the UI thread by way of the onLoadFinished() LoaderManager callback.
* To create and parse JSON strings, use the built-in Java JSON classes JSONObject and JSONArray.
* An AsyncTaskLoader uses an AsyncTask helper class to do work in the background, off the main thread.
* AsyncTaskLoader instances are managed by a LoaderManager.
* The LoaderManager lets you associate a newly created Activity with a loader using getSupportLoaderManager().initLoader().

**Question 1**

What permissions does your app need to connect to the internet?

* android.permission.CONNECTIVITY
* android.permission.INTERNET
* It doesn't need any special permissions, because all Android apps are allowed to connect to the internet.

**Question 2**

How does your app check that internet connectivity is available?

In the manifest:

* request ACCESS\_NETWORK\_STATE permission
* request ALL\_NETWORK\_STATE permission
* request NETWORK\_CONNECT permission

In the code:

* Wrap the code to connect to the internet in a try/catch block, and catch NO\_NETWORK errors.
* Use ConnectivityManager to check for an active network before connecting to the network.
* Present a dialog to the user reminding them to make sure that internet connectivity is available before they attempt to connect to the internet.

**Question 3**

Where do you implement the loader callback method that's triggered when the loader finishes executing its task?

* In the AsyncTaskLoader subclass. The AsyncTaskLoader must implement LoaderManager.LoaderCallbacks.
* In the Activity that displays the results of the task. The Activity must implement LoaderManager.LoaderCallbacks.
* In a Utility class that extends Object and implements LoaderManager.LoaderCallbacks.

**Question 4**

When the user rotates the device, how do AsyncTask and AsyncTaskLoader behave differently if they are in the process of running a task in the background?

* A running AsyncTask becomes disconnected from the activity, but keeps running. A running AsyncTaskLoader becomes disconnected from the activity and stops running, preserving system resources.
* A running AsyncTask becomes disconnected from the activity and stops running, preserving system resources. A running AsyncTaskLoader automatically restarts execution of its task from the beginning. The activity displays the results.
* A running AsyncTask becomes disconnected from the activity, but keeps running. A running AsyncTaskLoader automatically reconnects to the activity after the device rotation. The activity displays the results.

**Question 5**

How do you initialize an AsyncTaskLoader to perform steps, such as initializing variables, that must be done before the loader starts performing its background task?

* In onCreateLoader() in the activity, create an instance of the AsyncTaskLoader subclass. In the loader's constructor, perform initialization tasks.
* In onCreateLoader() in the activity, create an instance of the AsyncTaskLoader subclass. In the loader's init() method, perform initialization tasks.
* In the Activity, implement initLoader() to initialize the loader.
* Perform initialization tasks for the loader at the start of loadInBackgroud() in the Loader.

**Question 6**

What methods must an AsyncTaskLoader implement?

[Android fundamentals 07.3: Broadcast receivers](https://developer.android.com/codelabs/android-training-broadcast-receivers?index=..%2F..%2Fandroid-training)

[7. Summary](https://developer.android.com/codelabs/android-training-broadcast-receivers?index=..%2F..%2Fandroid-training#6)

* Broadcast receivers are fundamental components of an Android app.
* Broadcast receivers can receive broadcasts sent by the system or by apps.
* The Intent used in the broadcast mechanism is completely different from intents used to start activities.
* To process the incoming Intent associated with a broadcast, you subclass the BroadcastReceiver class and implement onReceive().
* You can register a broadcast receiver in the Android manifest file or programmatically.
* Local broadcasts are private to your app. To register and send local broadcasts, use LocalBroadcastManager. Local broadcasts don't involve interprocess communication, which makes them efficient. Using local broadcasts can also protect your app against some security issues, because data stays inside your app.
* To create unique Intent action names for broadcasts, a common practice is to prepend the action name with your package name.
* If your app targets API level 26 or higher, you cannot use the manifest to declare a receiver for most implicit broadcasts. (*Implicit broadcasts*, which include most system broadcasts, are broadcasts that don't target your app.) A few implicit broadcasts are [exceptions](https://developer.android.com/guide/components/broadcast-exceptions.html). However, you can use dynamic receivers to receive all broadcasts.

**Question 1**

What is a system broadcast?

* A message that your app sends and receives when an event of interest occurs in the app.
* A message that is sent from an app to a different component of the same app.
* A message that the Android system sends when a system event occurs.
* A message that the Android system receives when an event of interest occurs in your app.

**Question 2**

Which pair of methods do you use to register and unregister your broadcast receiver dynamically?

* registerBroadcast() and unRegisterBroadcast().
* registerComponentCallbacks() and unRegisterComponentCallbacks().
* registerBroadcastReceiver() and unRegisterBroadcastReceiver().
* registerReceiver() and unRegisterReceiver().

**Question 3**

Which of the following are true?

* Broadcast receivers can't see or capture the intents used to start an activity.
* Using a broadcast intent, you can't find or start an activity.
* You can use a broadcast intent to start an activity.
* You can receive the intent used to start activity in your broadcast receiver.

**Question 4**

Which class is used to mitigate the security risks of broadcast receivers when the broadcasts are not cross-application (that is, when broadcasts are sent and received by the same app)?

* SecureBroadcast
* LocalBroadcastManager
* OrderedBroadcast
* SecureBroadcastManager

[Android fundamentals 08.1: Notifications](https://developer.android.com/codelabs/android-training-notifications?index=..%2F..%2Fandroid-training)

[8. Summary](https://developer.android.com/codelabs/android-training-notifications?index=..%2F..%2Fandroid-training#7)

A *notification* is a message that you can display to the user outside of your app's normal UI:

* Notifications provide a way for your app to interact with the user even when the app is not running.
* When Android issues a notification, the notification appears first as an icon in the notification area of the device.
* To specify the UI and actions for a notification, use NotificationCompat.Builder.
* To create a notification, use NotificationCompat.Builder.build().
* To issue a notification, use NotificationManager.notify() to pass the notification object to the Android runtime system.
* To make it possible to update or cancel a notification, associate a notification ID with the notification.
* Notifications can have several components, including a small icon (setSmallIcon(), required); a title (setContentTitle()); and detailed text (setContentText()).
* Notifications can also include pending intents, expanded styles, priorities, etc. For more details, see [NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \t "_blank).

**Question 1**

Select all that are true for notification channels:

* You use notification channels to display notifications to the user in the device status bar.
* You use notification channels to group multiple notifications so that the user can control the notifications' behavior.
* Notification channels are available in older devices, those running Android 7.0 Nougat (API 24) and lower.
* Notification channels are not yet available in the Android Support Library package.

**Question 2**

Which API do you use to show a notification in the notification drawer and in the device's status bar?

* Notification.notify()
* NotificationManager.notify()
* NotificationCompact.notify()
* NotificationCompat.Builder.notify()

**Question 3**

Which component is *not* needed when you add a notification action?

* Icon that represents the action
* Title that describes the action
* Click listener for the action button click event.
* PendingIntent that's sent when the user taps the action button.

**Question 4**

Which API do you use to add an action button to a notification?

* NotificationCompat.addActionButton()
* NotificationCompat.Builder.addAction()
* Notification.Builder.addActionButton()
* NotificationManager.addAction()

**Question 5**

Suppose that you create an app that downloads a work of art on the user's device every day. Once the day's image is available, the app shows a notification to the user, and the user can download the image or skip the download. What PendingIntent method would you use to start a service to download the image?

* Activity.startService()
* PendingIntent.getBroadcast()
* PendingIntent.getActivity()
* PendingIntent.getService()

[Android fundamentals 09.1: Shared preferences](https://developer.android.com/codelabs/android-training-shared-preferences?index=..%2F..%2Fandroid-training)

[7. Summary](https://developer.android.com/codelabs/android-training-shared-preferences?index=..%2F..%2Fandroid-training#6)

* The [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html" \t "_blank) class allows an app to store small amounts of primitive data as key-value pairs.
* Shared preferences persist across different user sessions of the same app.
* To write to the shared preferences, get a [SharedPreferences.Editor](https://developer.android.com/reference/android/content/SharedPreferences.Editor" \t "_blank) object.
* Use the various "put" methods in a SharedPreferences.Editor object, such as [putInt()](https://developer.android.com/reference/android/content/SharedPreferences.Editor" \l "putInt(java.lang.String,%20int)" \t "_blank) or [putString()](https://developer.android.com/reference/android/content/SharedPreferences.Editor" \l "putString(java.lang.String,%20java.lang.String)" \t "_blank), to put data into the shared preferences with a key and a value.
* Use the various "get" methods in a SharedPreferences object, such as [getInt()](https://developer.android.com/reference/android/content/SharedPreferences" \l "getInt(java.lang.String,%20int)" \t "_blank) or [getString()](https://developer.android.com/reference/android/content/SharedPreferences" \l "getString(java.lang.String,%20java.lang.String)" \t "_blank), to get data out of the shared preferences with a key.
* Use the [clear()](https://developer.android.com/reference/android/content/SharedPreferences.Editor#clear()) method in a SharedPreferences.Editor object to remove all the data stored in the preferences.
* Use the [apply()](https://developer.android.com/reference/android/content/SharedPreferences.Editor#apply()) method in a SharedPreferences.Editor object to save the changes to the preferences file.

### ****Question 1****

In which lifecycle method do you save the app state to shared preferences?

onPause()

### ****Question 2****

In which lifecycle method do you restore the app state?

onCreate

### ****Question 3****

Can you think of a case where it makes sense to have both shared preferences and instance state?

[Android fundamentals 10.1 Part A: Room, LiveData, and ViewModel](https://developer.android.com/codelabs/android-training-livedata-viewmodel?index=..%2F..%2Fandroid-training)

[16. Summary](https://developer.android.com/codelabs/android-training-livedata-viewmodel?index=..%2F..%2Fandroid-training#15)

Now that you have a working app, let's recap what you've built. Here is the app structure again, from the beginning:

* You have an app that displays words in a list (MainActivity, RecyclerView, WordListAdapter).
* You can add words to the list (NewWordActivity).
* A word is an instance of the Word entity class.
* The words are cached in the RecyclerViewAdapter as a List of words (mWords). The list is automatically updated and redisplayed when the data changes.
* The automatic update happens because in the MainActivity, there is an Observer that observes the words and is notified when the words change. When there is a change, the observer's onChange() method is executed and updates mWords in the WordListAdapter.
* The data can be observed because it is LiveData. And what is observed is the LiveData<List<Word>> that is returned by the WordViewModel object.
* The WordViewModel hides everything about the backend from the user interface. It provides methods for accessing the UI data, and it returns LiveData so that MainActivity can set up the observer relationship. Views, activities, and fragments only interact with the data through the ViewModel. As such, it doesn't matter where the data comes from.
* In this case, the data comes from a Repository. The ViewModel does not need to know what that Repository interacts with. It just needs to know how to interact with the Repository, which is through the methods exposed by the Repository.
* The Repository manages one or more data sources. In the RoomWordsSample app, that backend is a Room database. Room is a wrapper around and implements an SQLite database. Room does a lot of work for you that you used to have to do yourself. For example, Room does everything that you used to use an SQLiteOpenHelper class to do.
* The DAO maps method calls to database queries, so that when the Repository calls a method such as getAllWords(), Room can execute SELECT \* from word\_table ORDER BY word ASC**\*\*\*\*.**
* The result returned from the query is observed LiveData. Therefore, every time the data in Room changes, the Observer interface's onChanged() method is executed and the UI is updated.

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Question 1**

What are the advantages of using a Room database?

* Creates and manages an Android SQLite database for you.
* Eliminates a lot of boilerplate code.
* Helps you manage multiple backends.
* Using a DAO, provides a mechanism for mapping Java methods to database queries.

**Question 2**

Which of the following are reasons for using a ViewModel?

* Cleanly separates the UI from the backend.
* Often used with LiveData for changeable data that the UI will use or display.
* Prevents your data from being lost when the app crashes.
* Acts as a communication center between the Repository and the UI.
* ViewModel instances survive device configuration changes.

**Question 3**

What is the DAO?

* Short for "data access object."
* A library for managing database queries.
* An annotated interface that maps Java methods to SQLite queries.
* A class whose methods run always in the background, not on the main thread.
* A class that the compiler checks for SQL errors, then uses to generate queries from the annotations.

**Question 4**

What are features of LiveData?

* When LiveData is used with Room, data updates automatically if all the intermediate levels also return LiveData (DAO, ViewModel, Repository).
* Uses the observer pattern and notifies its observers when its data has changed.
* Automatically updates the UI when it changes.
* Is lifecycle aware.