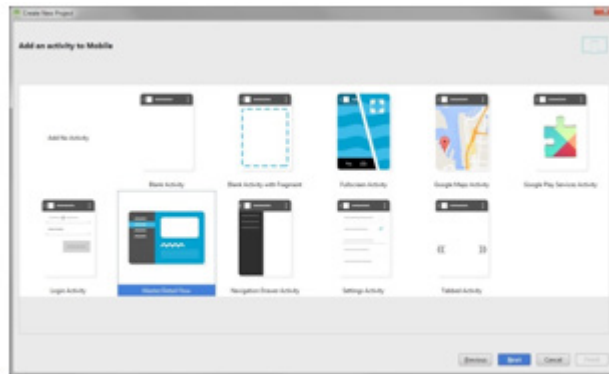


ACTIVITY 2.1.4

App Navigation

INTRODUCTION

In Android™ Studio, there are many different design patterns for getting around within an app: Master-detail view, Navigation Drawer, and Popover Menu, just to name a few. You've probably seen these and others in apps that you've used. The **File > New Project...** menu option in Android Studio provides some default navigation types.



Add an Activity to Mobile

The College App was created using the Navigation Drawer template. In this activity, you will learn how to modify a navigation drawer to suit your app's needs.

Materials

- Computer with Android™ Studio
- Android™ tablet and USB cable, or a device emulator

RESOURCES



The Pledge

Resources available online



College App Problem Statement

Resources available online

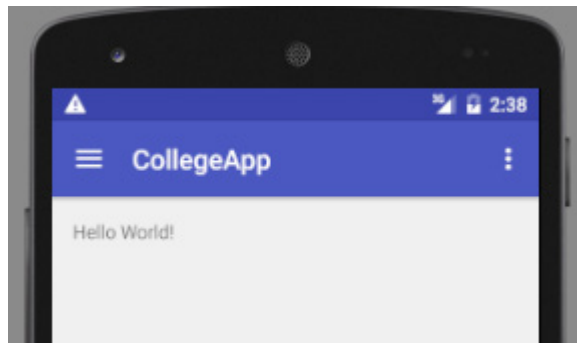
Procedure

Part I: The Nav Drawer Template

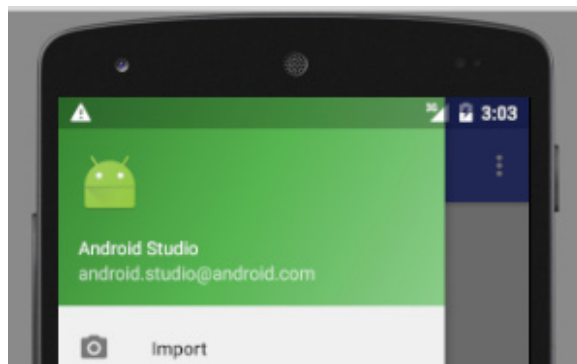
- 1 Form pairs as directed by your instructor.
- 2 Greet your partner and set team norms for pair programming.
- 3 Create a new project in Android Studio. Give this project the following values (when no value is specified, leave the defaults):
 - a. Application name: **CollegeApp**
 - b. Company Domain: **examples.pltw.org**
 - c. Store your project in your *AndroidProjects* folder.
 - d. Minimum SDK: **API 22**
 - e. Add an Activity to Mobile: **Navigation Drawer Activity**
 - f. Activity Name: **ApplicantActivity**

In this activity you will learn about only the portions of this code that are relevant to the Navigation Drawer.

- 4 Run your app. You should see a screen like the one below.

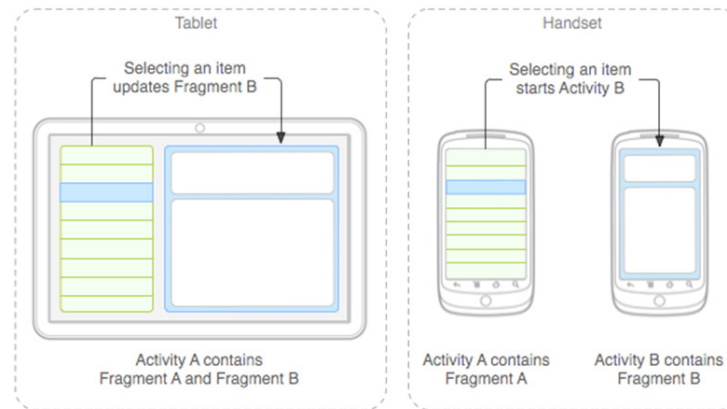


- 5 Click the hamburger icon (three horizontal lines ≡) to open the navigation drawer.



6 Review the slideshow.

Activities vs. Fragments



Computer Science A

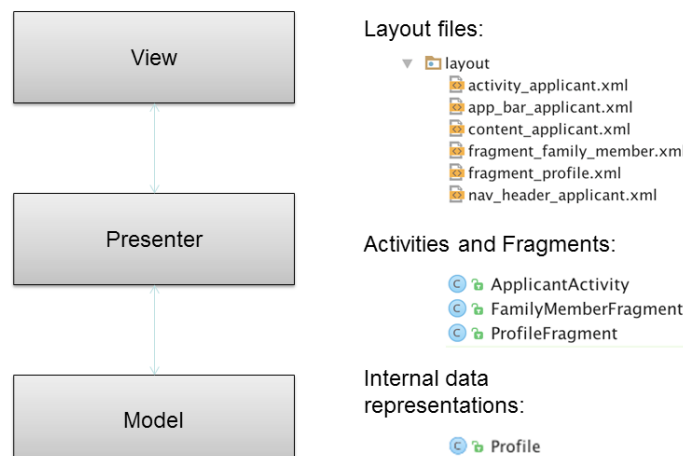
© 2016 Project Lead The Way, Inc.

According to Android's Developer website,

“Android introduced fragments in Android 3.0 (API level 11), primarily to support more dynamic and flexible UI designs on large screens, such as tablets. Because a tablet's screen is much larger than that of a handset, there's more room to combine and interchange UI components. Fragments allow such designs without the need for you to manage complex changes to the view hierarchy.”

<http://developer.android.com/guide/components/fragments.html>

The MVP Pattern



Computer Science A

© 2016 Project Lead The Way, Inc.

The Model – View – Presenter pattern is a derivative of MVC.

The big idea is encapsulation. All View concerns should be handled by the layout files.

Your Activities and Fragments should handle all of the connections between your data and the layout, and your Model classes should store all of the data you want to modify or display.

In MVP the Presenter serves as the middleman between View and Model.

<http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93presenter>

Creating an XML file

Right-click your layout directory and select
New > Layout resource file

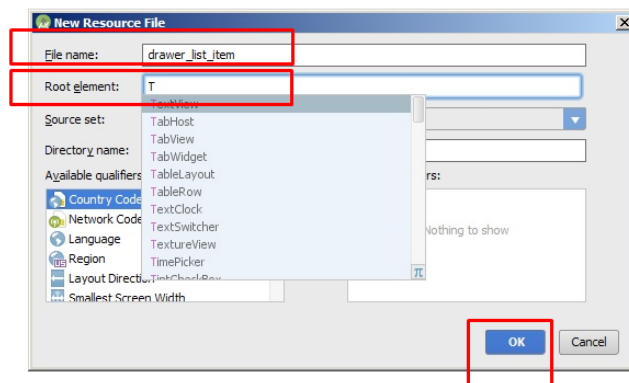


Computer Science A

© 2016 Project Lead The Way, Inc.

Creating an XML file

Name the file and give it a “Root Element”
before clicking **OK**



Computer Science A

© 2016 Project Lead The Way, Inc.

XML Root Elements

The outermost tag within an XML file



Computer Science A

© 2016 Project Lead The Way, Inc.

In this example, although line 1 contains the first tag, RelativeLayout is the root element. This is because the tag or line 1 is just metadata describing the type of XML document that this is, whereas RelativeLayout encloses all other tags, finally terminating on line 64. Note the gap between lines 7 and 55.

Inheritance: An Example

Class Name	Class Fields	Class Methods
Furniture	int height, width, depth	int squareFootage void setHeight void setDepth void setWidth
Chair	boolean pillowOn	void tidyUp
Couch	boolean pillow1on, pillow2on	void tidyUp int squareFootage
Bed	boolean pillowOn, blanketOn	void make

Computer Science A

© 2016 Project Lead The Way, Inc.

Imagine:

Furniture is a superclass

Chair, Couch, and Bed are subclasses of Furniture. That means they inherit properties of Furniture. For example, an instance of Couch could call squareFootage, or the Bed class could use the variable width in its make method.

Furniture could not, however, call tidyUp.

In cases where a superclass (Furniture) and its subclass (Couch) both define a method with the same signature, the subclass can access the superclass's version of that method by preceding it with the keyword "super" and the dot operator like super.squareFootage(). By default though, an instance of Couch will use Couch's implementation of squareFootage.

Inheritance: The Vocabulary

Subclass

Superclass

Extends

Inherits

Overrides

Computer Science A

© 2016 Project Lead The Way, Inc.

Here's an overview of the automatically generated files in the Navigation Drawer that are relevant to what you're learning now. (The XML files are found in the *layout* and *menu* project folders.)

- *nav_header_applicant.xml* contains the layout information for the green header area of the Navigation Drawer, which has the Android icon in it.
- *ApplicantActivity.java* contains presenter layer code for the app.
- *activity_applicant.xml* contains the overall layout of the main activity.
- *app_bar_applicant.xml* contains layout information for the blue bar at the top of the screen and the pink floating action button circle.
- *content_applicant.xml* contains the layout information for the main section of the screen, displaying "Hello World!" with a light grey background.
- *activity_applicant_drawer.xml* contains the layout information for the menu items found in the Navigation Drawer, like Import and Share.
- *applicant.xml* contains the layout information for the menu that appears when a user taps the icon (three dots) to the far right of the blue action bar.

To which of the XML files
does the "Slideshow" title
belong?

activity_applicant_drawer.xml

- 7 In the *menu* folder, from the file *activity_applicant_drawer.xml*, remove the sub-menu titled "Communicate" from the app by deleting the XML item element with the title attribute of the same name, and all the elements contained therein. Try to run your app now. What errors do you get?

- 8 Delete the lines in your Java code in *ApplicantActivity.java* that are producing these errors.

Why do you think they were producing errors?

- 9 Run your app and test it to see what has changed. Record your observations.


- 10 Summarize in your own words the steps it takes to remove a menu item from the Navigation Drawer. Verify your explanation with your instructor.

- 11 Now that you have removed a sub-menu from the Navigation Drawer, remove the Tools, Gallery, and Slideshow items from the remaining Navigation Drawer menu.

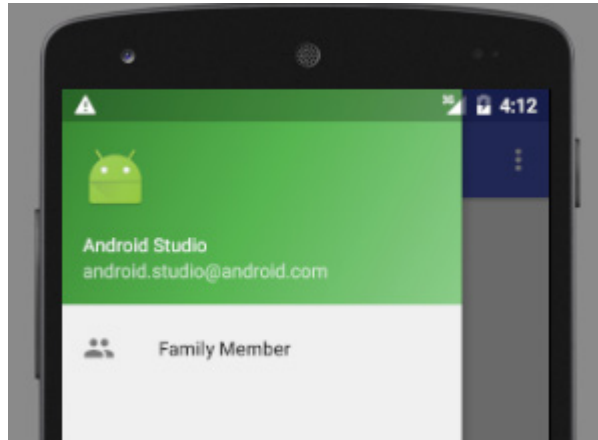
- 12 Change the remaining menu item, currently called “Import”, so that it has the following attributes:

- Id: @+id/family_member
- Icon: @drawable/ic_group_black_24dp
- Title: Family Member

- 13 Using Vector Asset Studio, you can include the *ic_group_black_24dp*, along with many other **material design icons**, in your project. Follow these steps:

- Right-click your *res* directory.
- Select **New > Vector Asset**.
- Make sure that **Material Icon** is selected.
- Click the **Choose** button.
- Select the **Social** category.
- Select the first icon that looks like: 
- Click **Next**.
- Keep the defaults and click **Finish**.

- 14 Check to see that the file name `ic_group_black_24dp` has turned from red to black in your XML, indicating that Android Studio can access this resource now that it has been added to the resource directory.
- 15 Modify the code in your java file to account for the fact that the navigation item is now named `family_member` instead of `nav_camera`.
- 16 Test your app to make sure that the Navigation Drawer now looks like this:



- 17 Add a new menu item to the Navigation Drawer by following these steps:
 - a. Type the following code just before the last remaining item tag in `activity_applicant_drawer.xml`

```
1: <item
2:     android:id="@+id/profile"
3:     android:icon="@drawable/ic_person_black_24dp"
4:     android:title="Profile" />
```

- b. As in step 12, add a new vector asset to your project's drawable resources.

NOTE

You can hover over each icon to find the one that is titled `ic_person_black_24dp`.

You have added a new menu item to the View layer of the app.

`ApplicantActivity` will not register any errors with this addition, but there is currently no place in the Java code to put the instructions for what should happen when the new Profile menu item is selected by a user. To handle this event, you must add code in `ApplicantActivity.java` inside the `onNavigationItemSelectedListener` method.

- xviii. Add an `else if` statement to test whether the id of the menu item is the same as the id of your new XML element. Leave the body of the `else if` clause empty. This will be filled in later when you add content screens for Family Member and Profile.

18 Test your app.

19 Remove the floating action button. You can always add it back later if you like.

You may find it helpful to remember that XML can be commented out by placing the code in question within `<!-- -->` tags in much the same way that you could use `/* */` in Java.

20 Test your app to verify that it still runs and that the floating action button is gone.

Part II: Adding Fragments

To add Fragments to your app, allowing for modular design that will make it easy to port to tablets or other devices at a later time, you will need to create a home for the Fragment in your layout, as well as create the Fragments in `ApplicantActivity` and write code to give your app logic for how to switch between them.

21 Start by editing your `content_applicant.xml` file. Remove the existing `TextView` element in this file and replace it with a `FrameLayout` element. Give this element the following attributes:

- `layout_width: match_parent`
- `layout_height: match_parent`
- `id: @+id/content_frame`

You already have code to detect when a user selects the Family Member or Profile menu items from the Navigation Drawer.

22 In that method, create a local Fragment variable identified by `contentFragment` and set its initial value to `null`.

To complete your project within a reasonable time, you will take advantage of **inheritance**. Inheritance allows one class to gain all of the methods and fields of another, and all you have to do as a programmer is add one keyword and an identifier to your class definition.

inherit or inheritance

When a subclass created automatically gains all the methods and fields of its superclass or parent class.

23 Create just one of the Fragments that you will need, the `FamilyMemberFragment`, by creating a new class with that name that extends `android.support.v4.app.Fragment`.

This Fragment will need its own layout. For now, keep it simple.

Create a layout file named `fragment_family_member.xml` and add a vertically oriented `LinearLayout` to it that contains a single `TextView` showing the string literal "Family Member Content".

- 24 Within `FamilyMemberFragment` override `onCreateView(LayoutInflater inflater, ViewGroup view, Bundle bundle)` with return type `View`.



- 25 In the first line of this method, call its super class's `onCreateView` method passing in the same parameters that you just created.
- 26 On the next line, declare a `View` named `rootView` and assign it the value retrieved by calling the `inflate` method of your `LayoutInflater` parameter with the arguments `R.layout.fragment_family_member`, your `ViewGroup` parameter, and the value `false`.
- This is how your `Fragment` will know to look for `fragment_family_member.xml` to get its layout. To get other layout elements later, you will call the `findViewById` method of `rootView`.
- 27 At the end of `onCreateView`, return `rootView`. This is as minimal a `Fragment` as you can create.
- 28 In `ApplicantActivity.java`, within the conditional that detects selection of the Family Member menu item, assign a new `FamilyMemberFragment` to `contentFragment`.
- 29 Repeat steps 22-27, this time to create `ProfileFragment` and a layout file for it similar to the layout file that you created for `FamilyMemberFragment`.
- 30 After your conditionals, within `onNavigationItemSelected`, type the following:

```
1: if (contentFragment != null) {
2:     FragmentTransaction ft = getSupportFragmentManager().
        beginTransaction();
3:     ft.replace(R.id.content_frame, contentFragment);
4:     ft.commit();
5: }
```

- 31 Test your app to make sure that the `Fragments` are properly replacing one another.
- You will eventually be saving the data for an applicant's family members and for themselves, so you need to create classes for this purpose.
- 32 The first class you will create is for an applicant's family member, specifically, a parent or guardian. Create a new class called `Guardian`. Create `String` instance variables for `firstName` and `lastName`.

- 33 Add accessor and mutator methods for each of these variables by right-clicking an empty line in the `Guardian` class and selecting **Generate... > Getter and Setter**. Highlight both of your variables and select **OK**.
- 34 Similarly, you need to track the applicant's own personal data. Repeat steps 31 and 32 for a new `Profile` class that will be used to store the first and last name of the applicant.
- 35 Test that all is well with your app, but know that there will be no new functionality with the addition of the two new classes.
- 36 Navigate to the Android developer website and search for "Navigation Drawer".
You should find two links near the top of your results, one labeled "Design" and one labeled "Training".
- 37 Use the Design link to help you answer the following question: Why is a navigation drawer a good fit for the College App?
- 38 Use the Internet to find at least three non-official sources of information that you could use to help you work with navigation drawers as an app developer. Describe your findings.

Part III: Make the Pledge

When you enter digital data in any application or on any website, you must assume the data you enter "lives" forever. Apps can store personal information and save it for later use. Sometimes, malicious users can gain access to this personal data and use it to create false credit card accounts, withdraw funds from someone's bank account, etc. As a responsible member of the cyber community, you should never reveal too much personal information about yourself, such as your full name and address, Social Security number, and credit card numbers unless you are absolutely certain the website or application is trusted.

In the app you are writing, what data could represent private information?

As a responsible programmer, how could you create a trusted app that would protect your users' privacy?

- 39 Read through  **The Pledge**. Sign the document, date it and return it to your teacher.

CONCLUSION

1. Which layers of the MVP pattern are currently present in the College App?
2. Of the layers that you listed in the previous question, what parts of the College App belong to which layers?
3. How did you use inheritance in the College App?
4. Would you prefer to use the navigation drawer implementation provided by Google or create your own? Why?
5. Use the Internet or any other resource to help you find information about three other forms of navigation used within Android apps. Discuss the pros and cons of using each as a replacement for the navigation drawer in the College App.

The Pledge: Do the Right Thing

RESOURCES



The Pledge

Resources available online

Introduction

To safely use the tools and knowledge that you will acquire in your study of computer science requires good judgement and a commitment to do the right thing.

Read the following contract. By signing below, you are indicating that you have read, understood, and agreed to all of the following statements. If you choose to sign the form, return the form to your teacher.

1. I will never use my knowledge of programming and computer science to access data or computing resources that I am not authorized to access.
2. I understand that accessing computing resources without permission can be a felony-level crime under the Computer Fraud and Abuse Act and other Federal and state laws.
3. I understand that penalties even for minors have included felony conviction, prison, and large fines to pay computing professionals to attend to a company's entire computing network on the basis of small interruptions in the company's service.
4. I understand that a single instance of penetration testing without permission is sufficient for a lifetime ban from professional cybersecurity positions.

Date

Student signature

Student name, printed