ALGONQUIN COLLEGE

CST2335 GRAPHICAL INTERFACE PROGRAMMING

Week 8

Fragments

Topics

- Different screen sizes (layout folders)
- Creating a Fragment
- Loading the fragment
- Fragments on a phone

Introduction

- So far we have learned that to view images in different languages, we place the images in different **Drawable** folders (e.g. drawable-fr, drawable-es) followed by the two-letter language code.
- The same principle applies to layout files as well. Instead of using the device's language, you will use the device pixel size.

Introduction

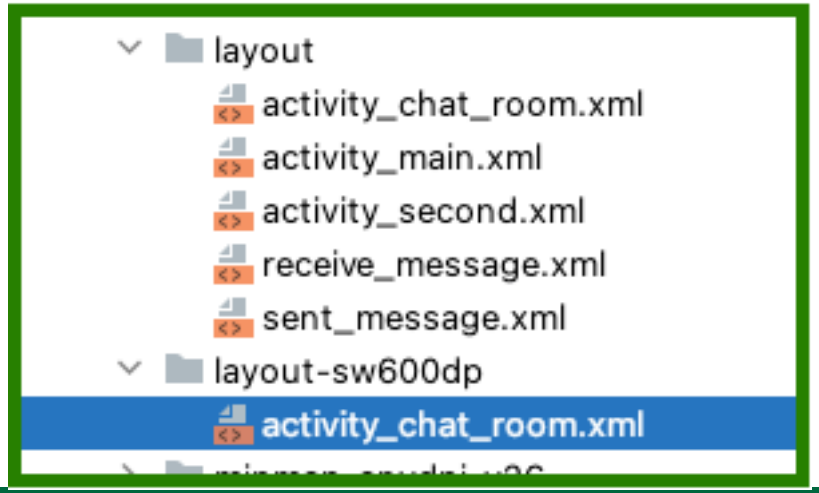
- Android uses the idea of a “**Smallest Width”**, which is the smaller number of the width and height
- For a display that is 800x600 pixels, the smallest width is 600
- For a display that is 1920x1080, the smallest width is 1080

Different Layout Folders

- There are some common folder names for different tablet sizes:
  - **layout-sw600dp** - this is for 7” tablets
  - **layout-sw720dp** - this is for 10” tablets
- There are also two names for any size display:
  - **layout-port** - for any device in portrait orientation
  - **layout-land** - for any device in landscape mode

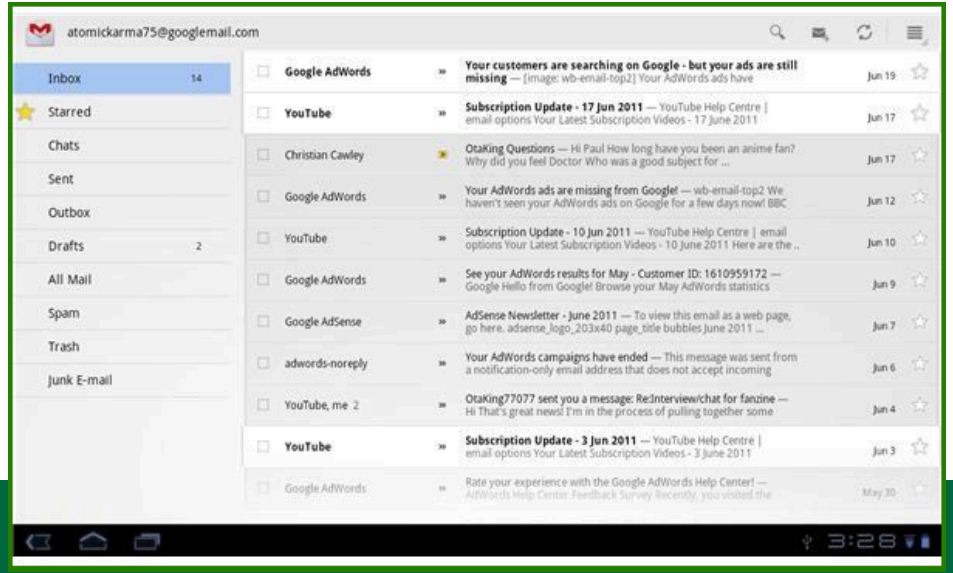
Creating layout for a tablet

- Create a new layout folder called **layout-sw600dp**, to support any tablets larger than 7”.
- Copy your **chat_layout.xml** file and paste into the new folder



The Big Picture

- For a large tablet, there is enough space to show more than just a list of items. Hence, you can have a list on the left and room beside it to show details about a selected item, like the picture below:



Leveraging FrameLayout

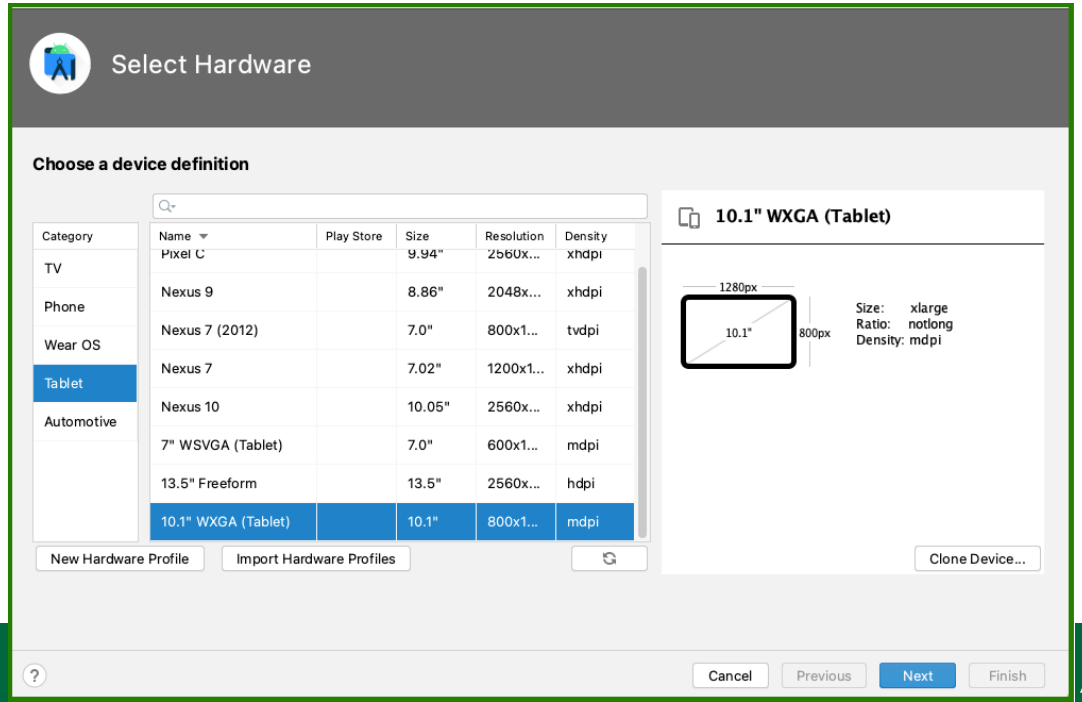
- A FrameLayout is a one that can only hold 1 item.
- We will be leveraging this layout to reserve space on our tablet screen to details about a selected chat message (similar to the diagram on the previous slide)
- For the tablet version of activity_chat_room.xml in the **layout-sw600dp** folder, we will make the RecyclerView have a width of 300dp instead of match_parent (see next slide)

Leveraging FrameLayout


```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recycleView"
    android:layout_width="300dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toTopOf="@id/textInput"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="send"
    android:id="@+id/sendButton"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"/>

<EditText
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:id="@+id/textInput"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@id/receiveButton"
    app:layout_constraintStart_toEndOf="@id/sendButton"/>

<Button android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Receive"
    android:id="@+id/receiveButton"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="@id/recycleView"/>

<FrameLayout android:id="@+id/fragmentLocation"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:background="@color/purple_200"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@id/recycleView"
    app:layout_constraintTop_toTopOf="parent"
```

Viewing the new tablet layout

- To see what the new layout looks like, we will have to create a 10” tablet emulator:



Refactoring the code

- Comment out the code you wrote for deleting a message from the RecyclerView. We want to use this instead to display messages on the side if it is a tablet

```
itemView.setOnClickListener( clk -> {
    /*
    AlertDialog.Builder builder = new Aler
    builder.setMessage("Are you sure you w
            .setTitle("Danger!")
            .setNegativeButton("Cancel", (
            .setPositiveButton("Delete", (
                position = getAbsoluteAdap
                ChatMessage removedMsg = m
                messages.remove(position);
                adt.notifyItemRemoved(posi
                Snackbar.make(messageText,
                    .setAction("UNDO", cl
                    messages.add(position,
                    adt.notifyItemInserted
                }).show();
            })
            .create().show();*/
});
```

Refactoring the code

- In ChatRoomViewModel, create a variable of type **MutableLiveData<ChatMessage>**:

```
public class ChatRoomViewModel extends ViewModel {

    public MutableLiveData<ArrayList<ChatMessage>> messages = new MutableLiveData< >();
    public MutableLiveData<ChatMessage> selectedMessage = new MutableLiveData< >();

}
```

Refactoring the code

- Now in the ChatRoomActivity class, where you used to have your AlertDialog about deleting the object, instead you should find the selected chat message and post the value to the selectedMessage variable you just created:

```
itemView.setOnClickListener( click -> {
    int position = getAbsoluteAdapterPosition();
    ChatMessage selected = messages.get(position);

    chatModel.selectedMessage.postValue(selected);
 });
```

Refactoring the code

- Then in the onCreate() function, register as a listener to the MutableLiveData object:
- Now whenever the user clicks on a row, you can retrieve the ChatMessage object at that position, and post it to the ViewModel as the new value.
- In the observe function, you can create a new Fragment object to show details for that object

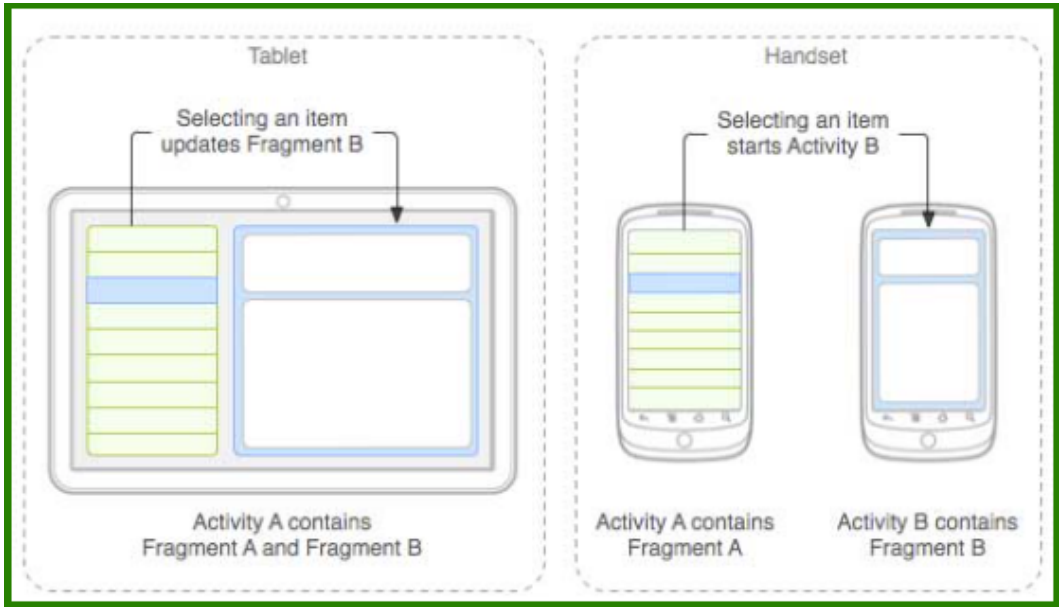
```
chatModel.selectedMessage.observe(this, (newMessageValue) -> {


 });
```

Fragments

- A Fragment is like an Activity object, except that it does not have to take up the whole screen.
- An Activity object represents the entire screen and if you want to change screens, you start another Activity with the startActivity() function.
- Fragments do the same thing although a Fragment can take up only part of the screen, like the left or right sides.

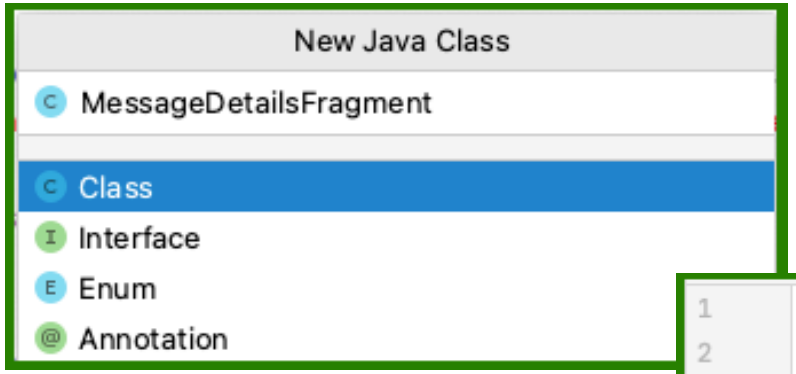
Fragments

- Here, we will use Fragments to view a Chat Message when you click on an item in the list view.



Creating a Fragment

- Create a new Java class called **MessageDetailsFragment** that extends Fragment.




```
1  package algonquin.cst2335.torunse;
2
3  import androidx.fragment.app.Fragment;
4
5  public class MessageDetailsFragment extends Fragment {
6  }
7
```

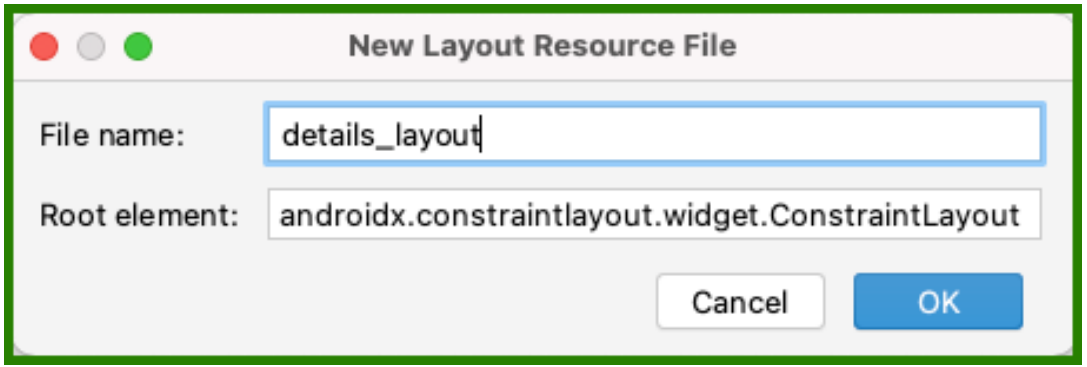
Creating a Fragment

- Type **Ctrl+O** to generate the onCreateView() function (delete everything that has to do with @Nullable)

```
public class MessageDetailsFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  Bundle savedInstanceState) {

        return super.onCreateView(inflater, container, savedInstanceState);
    }
}
```

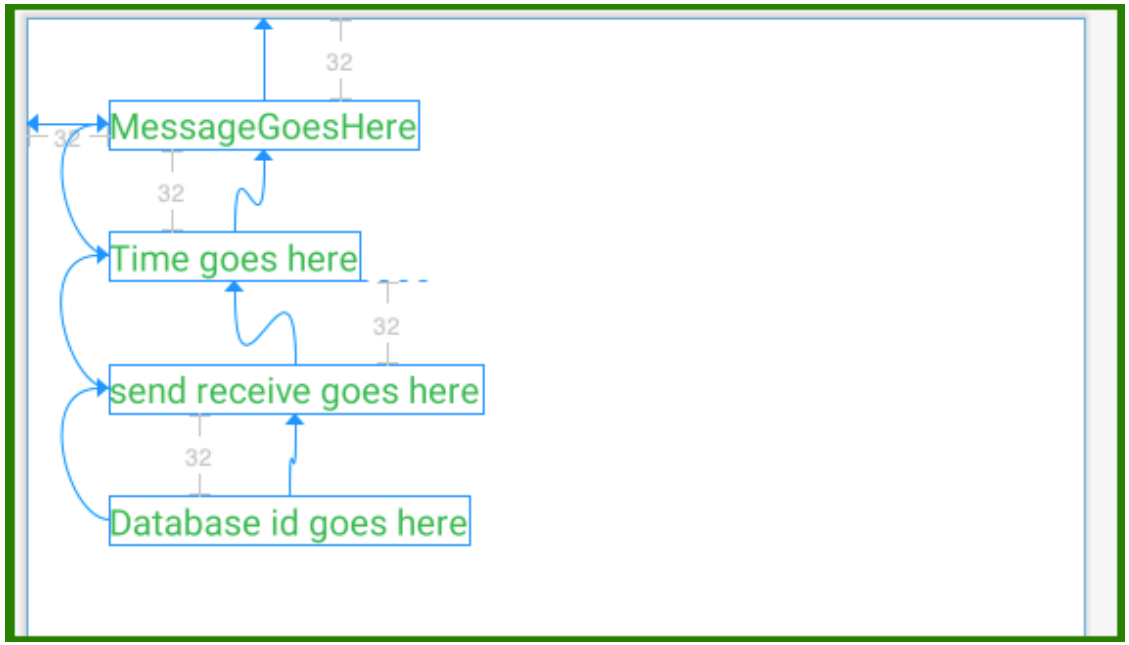
Create layout file to show message details

- Now create a Layout file for showing the details of a message: (**Database ID**, **Message**, **isSentOrReceive**, **timeSent**).
- Create a layout file called **details_layout.xml**.
- Remember, a Fragment is like an Activity, only it doesn't have to take the whole screen:



Create layout file to show message details

- There are four things to display, so we drag 4 TextViews on the display



onCreateView function of MessageDetailsFragment

- In the onCreateView function of **MessageDetailsFragment** class, create the view binding class for this layout:

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);

    DetailsLayoutBinding binding = DetailsLayoutBinding.inflate(inflater);
    return binding.getRoot();
}
```

Create a constructor for the MessageDetailsFragment

- Create a constructor for the **MessageDetailsFragment** class as follows:

```
ChatMessage selected;

public MessageDetailsFragment(ChatMessage m){
    selected = m;
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);
    DetailsLayoutBinding binding = DetailsLayoutBinding.inflate(inflater);

    binding.messageText.setText( selected.message );
    binding.timeText.setText(selected.timeSent);
    binding.databaseText.setText("Id = " + selected.id);

    return binding.getRoot();
}
```

Loading the fragment (Tablet version)

- Back in **ChatRoom** class, in the observe() function, add the code to load a Fragment.

```
chatModel.selectedMessage.observe(this, (newMessageValue) -> {
    MessageDetailsFragment chatFragment = new MessageDetailsFragment
    (newMessageValue);
    getSupportFragmentManager().beginTransaction()
    .replace(R.id.fragmentLocation, chatFragment)
    .commit();
});
```

Fragments on a phone

- We place the ConstraintLayout into a FrameLayout and move the xmlns: declarations to the FrameLayout as shown:

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent">


   <FrameLayout android:id="@+id/fragmentLocation"
                android:layout_width="match_parent"
                android:layout_height="match_parent">


   <ConstraintLayout>
     <RecyclerView> </RecyclerView>
 ...
     </ConstraintLayout> </FrameLayout>
 </LinearLayout>
```

Fragments on a phone

- Give the **FrameLayout** in the phone layout the same id as for the tablet.
- When the Fragment loads overtop the RecyclerView, you will still be able to see the RecyclerView through the Fragment.
  - You will see two things piled overtop one another.
  - You can fix this just by making the Fragment's background color to be white: **android:background="@color/white"**

Fragments on a phone

- Also, when you click on the back arrow, you will go back to the previous activity page.
  - You will not be able to undo loading a Fragment overtop the RecyclerView on a phone.
- In order to fix this, you can add a function call as part of the Builder pattern when creating a FragmentTransaction:
  - **.addToBackStack("")** - this adds the transaction to the stack of pages to undo by pressing the back arrow.