The University Of Sheffield.

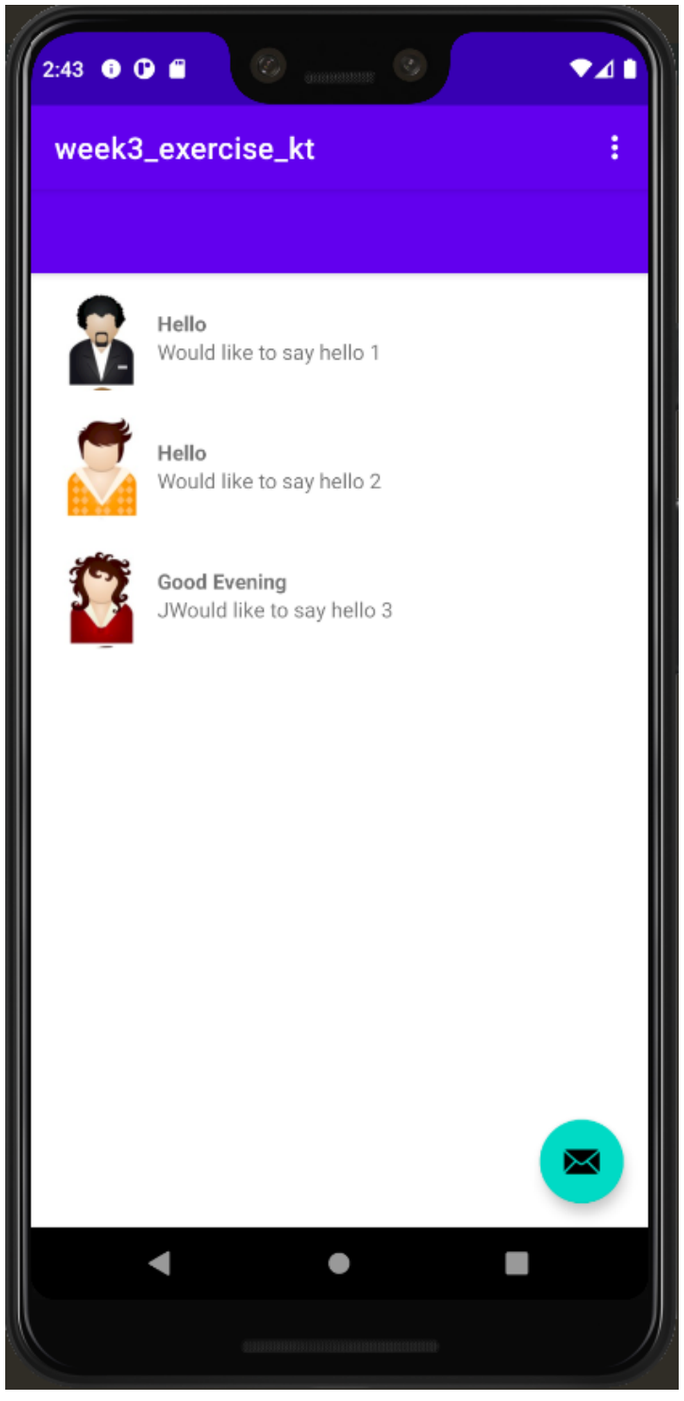
COM4510/6510
Software Development for Mobile Devices

Lab 4: Getting images from the gallery

Temitope (Temi) Adeosun
The University of Sheffield
t.adeosun@sheffield.ac.uk

Week 4

- Last week we saw how to get a list of messages contained in a RecyclerView and how to show the list in an activity.



Week 4

- This week we will see how
  - to store images in a GridLayout inside a RecyclerView rather than in a list
  - to get images from the Android gallery and to show them in the grid
  - to allowed clicking on an element and to display the element.

Final solution look

Last week's solution

- Check the solution provided for last week
- Make sure you understand it
- Today we will take that code and introduce the following changes:
  - to allow more element insertion in the RecyclerView, we will use a list rather than an array
    - it is far better as it does not constrain us to a max number of elements
  - we will display only images (no text)
  - we will present the images in a grid rather than a list

Update: Changes to Layout

- Open the ***list_item_image.xml*** layout file from ***res > layout***
  - In the code view, delete the second <LinearLayout> tag and all the child elements
  - Update the <ImageView> tag:
    - Change layout_width to “wrap_content”
    - Change layout_height to 150dp
    - Remove layout weight property
- The layout’s XML should look like this when you are done:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
 android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_margin="8dp"
   android:orientation="horizontal">

   <ImageView
      android:id="@+id/image_item"
      android:layout_width="wrap_content"
      android:layout_height="150dp" />
</LinearLayout>
```

Changes to the code

- Rename the class MyElement as ImageElement
  - use Refactor > Rename in AndroidStudio
    - NEVER change by direct edit!!!
- Modify MyAdapter:
  - Create a companion object for the MyAdapter class:

```
companion object {    }
```

- Re-define the ***item*** field as a list (it was an array), remove the private access modifier, and move it into the companion object

```
lateinit var items: List<ImageElement>
```

Remember to update all references to this object to use this type.

Changes to the code (cont.)

- Modify ImageElement

```
class ImageElement {
   var image = -1
   var file: MediaFile? = null
   constructor(image: Int)  {  this.image = image    }
}
```

  - Note the removal of ***title*** and ***previews*** from the code

- Update: Update MyAdapter. In onBindViewHolder and constructor, remove lines referencing to the ***title*** and ***preview*** properties
  - In onBindViewHolder, change if conditional to:
    `if (items[position].image != -1)`
- Update: Add the EasyImage dependency to the app gradle file and allow gradle sync:

```
implementation 'com.github.jkwiecien:EasyImage:3.2.0'
```

Changes to the code (cont.)

- Update: If you get warning:
  "Failed to resolve: com.github.jkwiecien:EasyImage:3.2.0"
  - Update: Open **setthings.gradle** file and add to repositories sections:
    maven { url = uri('https://jitpack.io') }
- Update: In ImageElement,

import MediaFile:

```
import pl.aprilapps.easyphotopicker.MediaFile
```

Changes to the code (cont.)

- in MainActivity.onCreate Change the Recycler from a list to a grid

```
mRecyclerView = findViewById(R.id.my_list)

val numberOfColumns = 4

mRecyclerView.layoutManager = GridLayoutManager(this, numberOfColumns)
```

- Change myDataset to a MutableList

```
private val myDataset: MutableList<ImageElement> = ArrayList<ImageElement>()
```

- Define an init method preparing images for the grid – invoke it from onCreate

```
private fun initData() {
        repeat(5){
            myDataset.add(ImageElement(R.drawable.joe1))
            myDataset.add(ImageElement(R.drawable.joe2))
            myDataset.add(ImageElement(R.drawable.joe3))
        }
}
```

Code change result

- Run the application

The University Of Sheffield.

Getting images from the Gallery

- So far we have added images from res/drawable
- This is not good
  - We should get the pictures from the gallery or the camera
- We are going to modify the program to get images from the gallery

The University Of Sheffield.

For this exercise, download the code provided

You will have to look at the code in Lab4Exercise 2 and try to understand it

So open it in AndroidStudio

EasyImage

- We will use a library called [EasyImage]
- To add the library to our project I have modified
  - build.gradle (Module: app)
  - and inserted the line in red

```
dependencies {
    implementation 'androidx.core:core-ktx:1.6.0'
    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.1'
    implementation 'androidx.navigation:navigation-fragment-ktx:2.3.5'
    implementation 'androidx.navigation:navigation-ui-ktx:2.3.5'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

    implementation 'com.github.jkwiecien:EasyImage:3.2.0'
}
```

The University Of Sheffield.

- EasyImage allows us to get images from the Gallery/Document folder and to take pictures with the camera
- In MainActivity.onCreate
  - I have added a call to this method to initialize EasyImage (using the object builder syntax):

```
private fun initEasyImage() {
    easyImage = EasyImage.Builder(this)
            .setChooserTitle("Pick media")
            .setFolderName("EasyImage sample")
            .setChooserType(ChooserType.CAMERA_AND_GALLERY)

            // .setChooserType(ChooserType.CAMERA_AND_DOCUMENTS)
            .allowMultiple(true)
            .setCopyImagesToPublicGalleryFolder(true)
            .build()
}
```

To call the gallery app

- A floating button is defined in the layout file activity_camera.xml

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
   android:id="@+id/fab_gallery"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_gravity="bottom|left"
   android:layout_margin="@dimen/fab_margin"
   app:backgroundTint="@android:color/holo_orange_light"
   app:fabSize="normal"
   android:src="@drawable/ic_gallery_black_24dp" />
```

- In MainActivity.onCreate we define the behaviour of the button

```
val fabGallery: FloatingActionButton = findViewById(R.id.fab_gallery)
fabGallery.setOnClickListener(View.OnClickListener {
   easyImage.openChooser(activity)
})
```

The University Of Sheffield.

EasyImage gets images from the Gallery or Camera by calling to open the system's action selector

easyImage.openChooser(activity)

This works as follows:

Activity1:
easyImage.openChooser(current_activity)

Activity2:
will performs operation and returns a result in an Intent

Activity1:

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
```

We'd see more about opening other activities using intents later.

- Results from the Gallery app will be received via

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    easyImage.handleActivityResult(requestCode, resultCode,data,this,
        object: DefaultCallback() {
            override fun onMediaFilesPicked(imageFiles: Array<MediaFile>, source: MediaSource) {
                onPhotosReturned(imageFiles)
            }

            override fun onImagePickerError(error: Throwable, source: MediaSource) {
                super.onImagePickerError(error, source)
            }

            override fun onCanceled(source: MediaSource) {
                super.onCanceled(source)
            }
        })
}
```

Add to the grid

```
/**
 * add the selected images to the grid
 */
private fun onPhotosReturned(returnedPhotos: Array<MediaFile>) {
    myDataset.addAll(getImageElements(returnedPhotos))
    // we tell the adapter that the data is changed and hence the grid needs
    // refreshing
    mAdapter.notifyDataSetChanged()
    mRecyclerView.scrollToPosition(returnedPhotos.size - 1)
}

/**
 * given a list of photos, it creates a list of ImageElements we do not know how many elements we will have
 */
private fun getImageElements(returnedPhotos: Array<MediaFile>): List<ImageElement> {
    val imageElementList: MutableList<ImageElement> = ArrayList<ImageElement>()
    for (file in returnedPhotos) {
        val element = ImageElement(file)
        imageElementList.add(element)
    }
    return imageElementList
}
```

Changing ImageElement

- EasyImage returns MediaFile object type rather than an int in res/drawable
- We have to store it in ImageElement
- Change ImageElement to look like this

```
class ImageElement {
    var image = -1
    var file: MediaFile? = null

    constructor(image: Int) {
        this.image = image
    }
    constructor(fileX: MediaFile?) {
        file = fileX
    }
}
```

- If the element has a link to res/drawable, image != -1; otherwise it will have a file associated

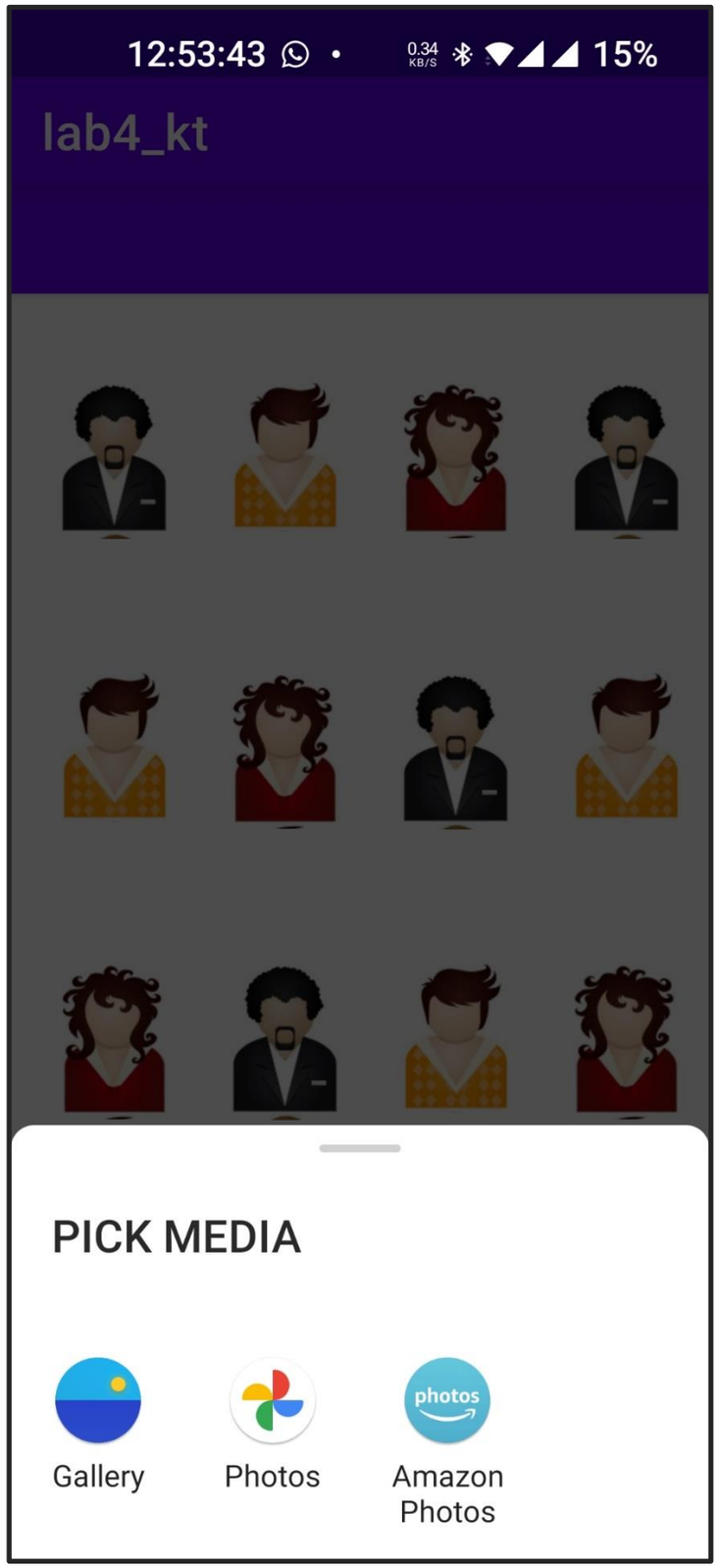
Displaying the image on the grid

Change MyAdapter to allow display of loaded image

- For images in res/drawable you assign the resource via imageView.setImageResource
- Per files you use imageView.setImageBitmap

```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    //Use the provided View Holder on the onCreateViewHolder method to populate the
    // current row on the RecyclerView
    if (items[position].image != -1) {
        holder.imageView.setImageResource(items[position].image)
    } else if (items[position].file != null) {
        val myBitmap = BitmapFactory.decodeFile(items[position].file?.file?.absolutePath)
        holder.imageView.setImageBitmap(myBitmap)
    }
}
```

Run the program



When you click on the button, Android will ask what application to launch

Choose Photos/Gallery, if multiple options available, else system will select the only available option.

Some versions of Android have issues with Gallery

The photos you choose will be added to the grid.

Show Image Activity

- An activity ShowImageActivity has been added.
  - Check it out
- shows just the clicked image from the ImageElement

ShowImageActivity

Read and understand what is happening in this code

```
class ShowImageActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_message2)
        val b: Bundle? = intent.extras
        var position = -1

        if (b != null) {
            // this is the image position in the itemList
            position = b.getInt("position")
            if (position != -1) {
                val imageView = findViewById<ImageView>(R.id.image)
                val element = MyAdapter.items[position]
                if (element.image != -1) {
                    imageView.setImageResource(element.image)
                } else if (element.file != null) {
                    val myBitmap = BitmapFactory.decodeFile(element.file!!.file.absolutePath)
                    imageView.setImageBitmap(myBitmap)

        ...
}
```

Optional

- The Layout file (list_item_image) contains only

```
<ImageView
   android:id="@+id/image_item"
   android:layout_width="wrap_content"
   android:layout_height="150dp"
   android:padding="10dp" />
```

Connect to Adapter

- Modify the onClickListener in onBindViewHolder in MyAdpter to allow response to image click

```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    ...
    holder.itemView.setOnClickListener(View.OnClickListener {
        val intent = Intent(context, ShowImageActivity::class.java)
        intent.putExtra("position", position)
        context.startActivity(intent)
    })
}
```

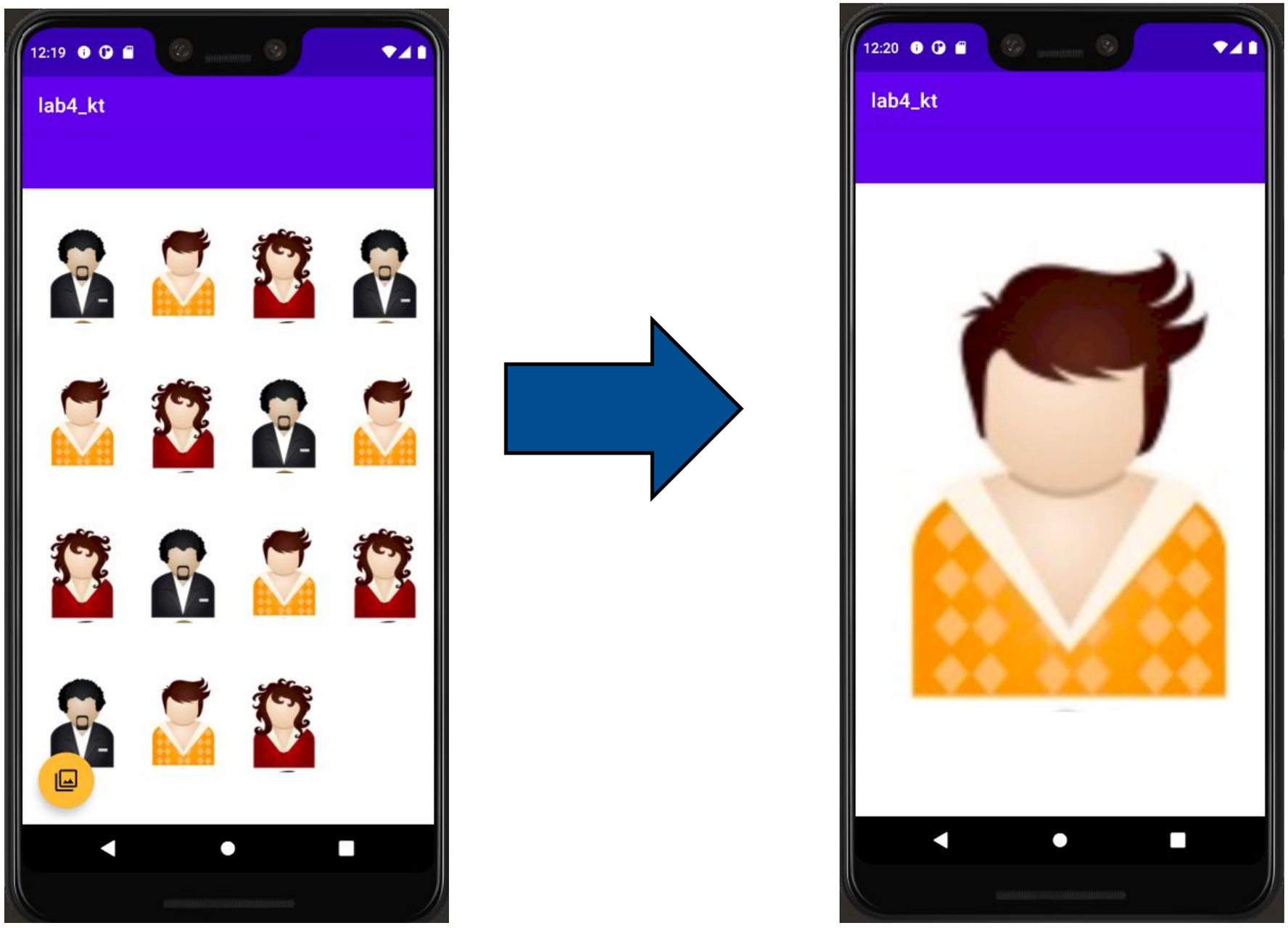
ShowImageActivity

- onCreate uses similar code to the one in MyAdapter.onBindViewHolder to display the image

```
class ShowImageActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
...
            if (position != -1) {
                val imageView = findViewById<ImageView>(R.id.image)
                val element = MyAdapter.items[position]
                if (element.image != -1) {
                    imageView.setImageResource(element.image)
                } else if (element.file != null) {
                    val myBitmap = BitmapFactory.decodeFile(element.file!!.file.absolutePath)
                    imageView.setImageBitmap(myBitmap)
                }
            }
        }
    }
}
```

ShowImageActivity

- Run code and click on an image



The University Of Sheffield.

Update:

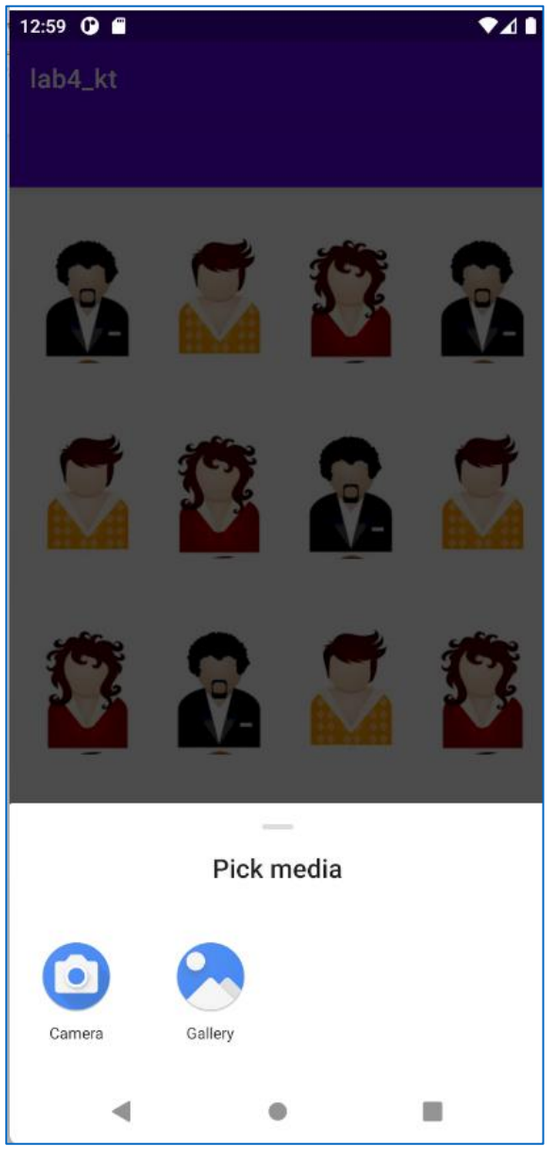
Using the camera

This is your final exercise

Enabling the camera

- Specify an intent to use the camera in your manifest using the queries tag in the AndroidManifest file

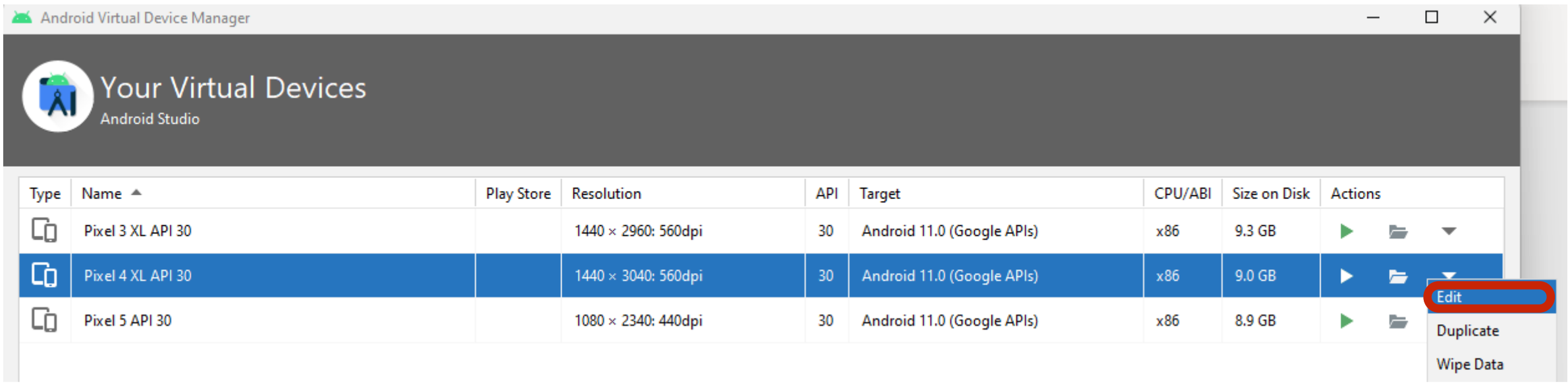
```
<queries>
   <intent>
      <action android:name="android.media.action.IMAGE_CAPTURE" />
   </intent>
</queries>
```



- Run your app and select the Fab button again. Now the camera app is listed in the action selector and you can take pictures.

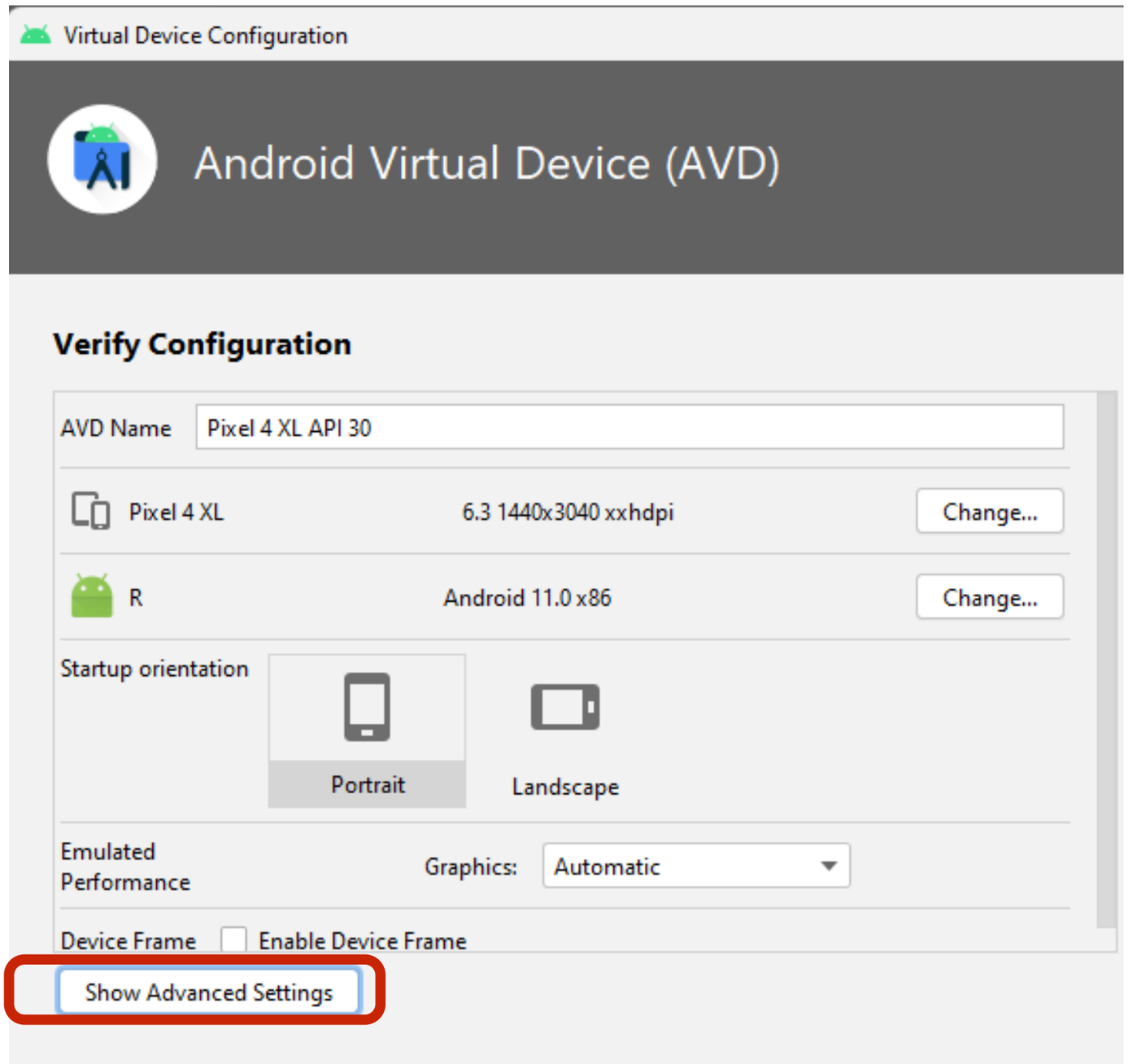
(Optional) Enabling the host camera

- Your virtual device will use the emulated camera for the front camera. To change:
  - Go to Tools > Android > AVD Manager
  - Select the emulator you are going to use e.g. Pixel 4 XL API 30 in the image below
  - under Actions click on the arrow to open the action menu and select **Edit**



(Optional) Enabling the host camera continues

- Click the **Show Advanced Settings** button



(Optional) Enabling the host camera continues

- Scroll to the Camera setting and select a webcam from the host computer (if any present).
- Then click finish.