CSCI 4237

Software Design for Handheld Devices

THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Lecture 7 - Networking (cont.)

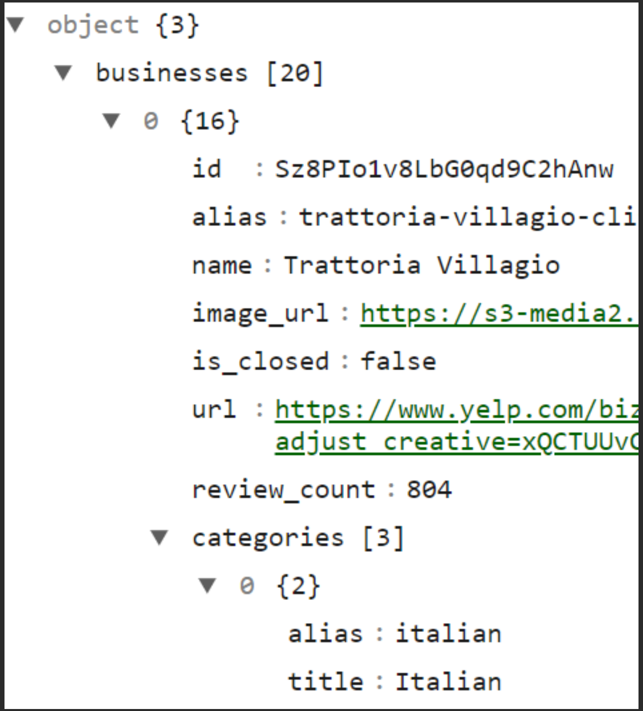
Mike Cobb

Upcoming

- Quiz 2 next week in class
- Project 1 due on 3/8Mar

Parsing Response

The Yelp Search API response is long & complex, but we only need a few fields from each restaurant



```
▼ object {3}
  ▼ businesses [20]
    ▼ 0 {16}
          id   : Sz8PIo1v8LbG0qd9C2hAnw
          alias : trattoria-villagio-cli
          name : Trattoria Villagio
          image_url : https://s3-media2.
          is_closed : false
          url  : https://www.yelp.com/biz
                 adjust_creative=xQCTUUvC
          review_count : 804
      ▼ categories [3]
        ▼ 0 {2}
              alias : italian
              title : Italian
```

We are looking for the restaurant

- Name
- Image_url
- url
- Category – Title
- Rating

JSON Parsing

Android has built-in JSON parsing support under the org.json.* package. You pick off different pieces of data from the JSON by calling getters.

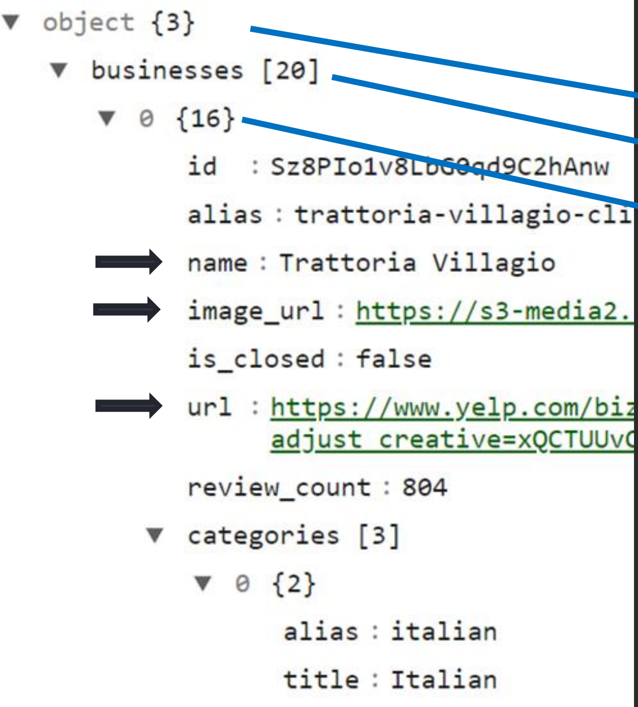
```
// Represents the JSON from the root level
val json = JSONObject(responseString)
```

JSON Parsing

This JSON parser starts at the *root* of the JSON and you write the code to work top-down thru the layers.

```
// Represents the JSON from the root level
val json = JSONObject(responseString)
```

Parsing Search Yelp Data

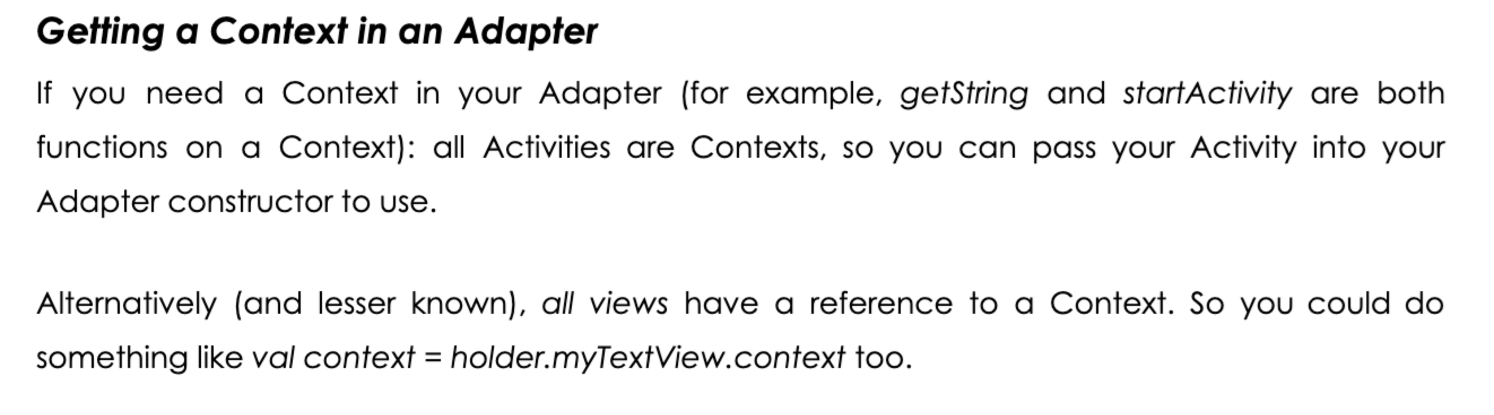


```
▼ object {3}
  ▼ businesses [20]
    ▼ 0 {16}
        id : Sz8PIo1v8LbG0qd9C2hAnw
        alias : trattoria-villagio-cli
        name : Trattoria Villagio
        image_url : https://s3-media2.
        is_closed : false
        url : https://www.yelp.com/biz
              adjust_creative=xQCTUUvC
        review_count : 804
      ▼ categories [3]
        ▼ 0 {2}
            alias : italian
            title : Italian
```

```
// Parse our JSON string
// Represents the JSON from the root level
val json = JSONObject(responseBody)
val businesses=json.getJSONArray(“businesses”)

for (i in 0 until businesses.length()){
     val currentBusiness=businesses.getJSONObject(i)

     val name=currentBusiness.getString("name")

     val rating=currentBusiness.getDouble("rating")

     val url=currentBusiness.getString("url")

     val icon=currentBusiness.getString("image_url“)

val categories=currentBusiness.getJSONArray("categories")
val currentCategory=categories.getJSONObject(0)
val title=currentCategory.getString("title")
```

Project 1 - Getting a Context in an Adapter

- There's a Tips section at the end of the project write-up – this one will probably come in handy soon...:



Getting a Context in an Adapter

If you need a Context in your Adapter (for example, *getString* and *startActivity* are both functions on a Context): all Activities are Contexts, so you can pass your Activity into your Adapter constructor to use.

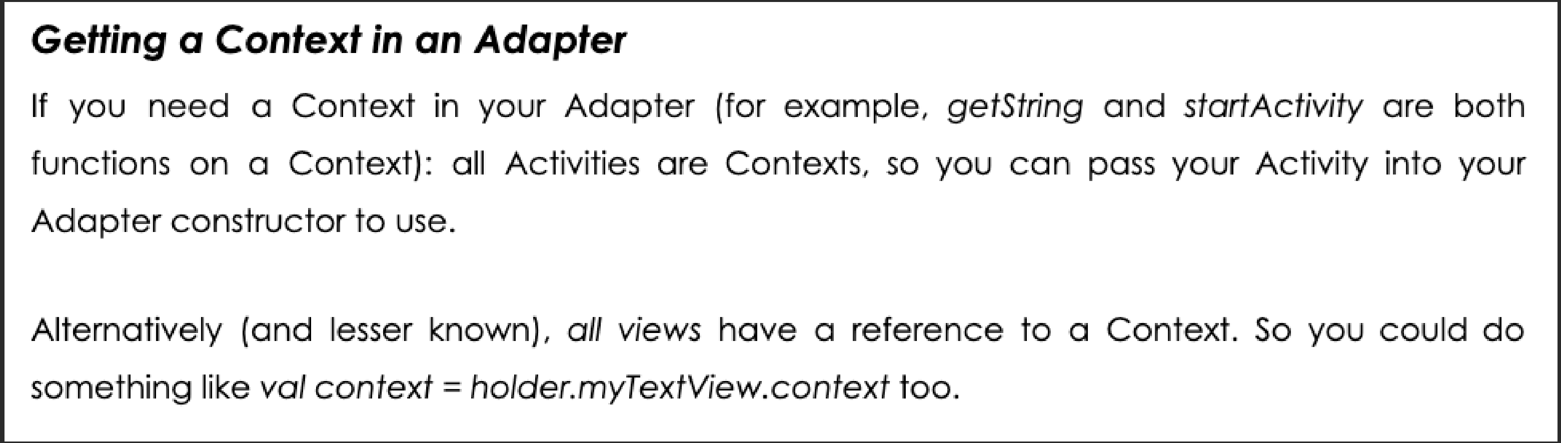
Alternatively (and lesser known), *all views* have a reference to a Context. So you could do something like *val context = holder.myTextView.context* too.

Today

- JSON Parsing
- Maps Maintenance
- Image Loading
- Serializable / Parcelable
- OAuth

Using Intents from Non-Activities

Similar to *getString*: calling *startActivity* **also** requires a Context.



Getting a Context in an Adapter

If you need a Context in your Adapter (for example, *getString* and *startActivity* are both functions on a Context): all Activities are Contexts, so you can pass your Activity into your Adapter constructor to use.

Alternatively (and lesser known), *all views* have a reference to a Context. So you could do something like *val context = holder.myTextView.context* too.

From the Tips section at the bottom of the writeup

Opening up the URL?

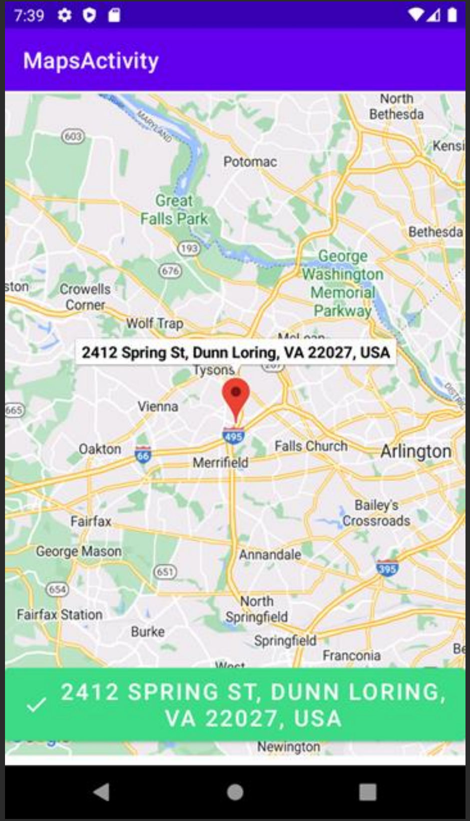
- Where can we put an onClickListener for the recyclerView of restaurants?

- Recall we can use intents to call other applications on the phone.

```
val intent=Intent(Intent.ACTION_VIEW)
intent.data= Uri.parse(url)
startActivity(intent)
```

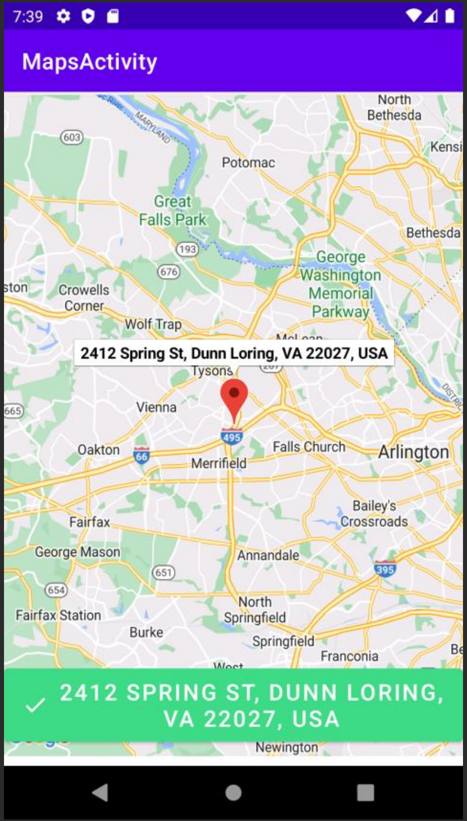
Finishing the Maps Activity

Customizing the MapsActivity



- Current Location button (in a couple weeks)
- Confirmation button materialButton

Customizing the MapsActivity



7:39

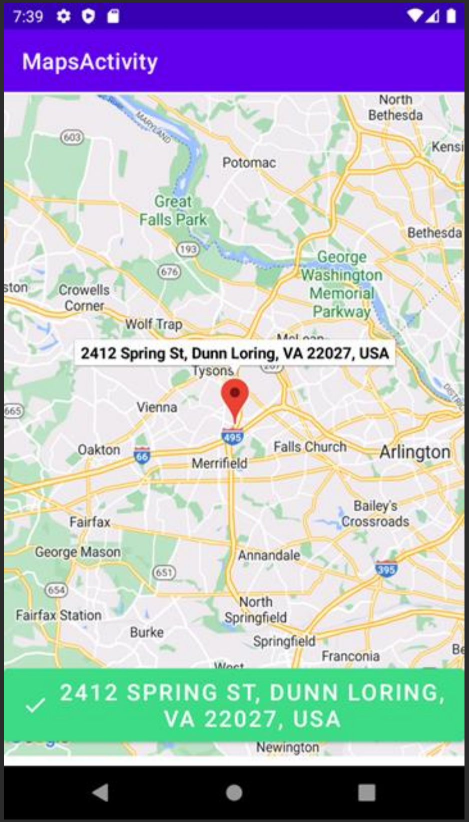
MapsActivity

2412 Spring St, Dunn Loring, VA 22027, USA

✓ 2412 SPRING ST, DUNN LORING, VA 22027, USA

The buttons will need to *overlap* the map, so we'll need to put the map fragment into a ConstraintLayout.

Customizing the MapsActivity

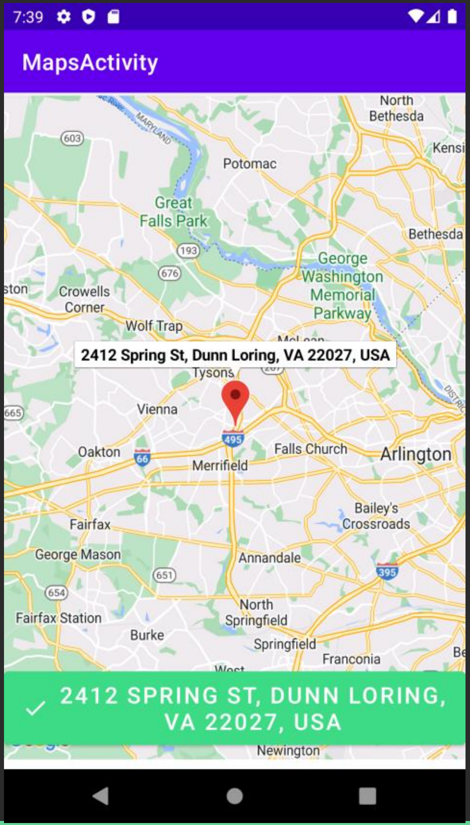


2412 Spring St, Dunn Loring, VA 22027, USA

✓ 2412 SPRING ST, DUNN LORING, VA 22027, USA

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:map="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <fragment
       android:id="@+id/map"
       android:name="com.google.android.gms.maps.SupportMapFragment"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       tools:context=".MapsActivity" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Customizing the MapsActivity

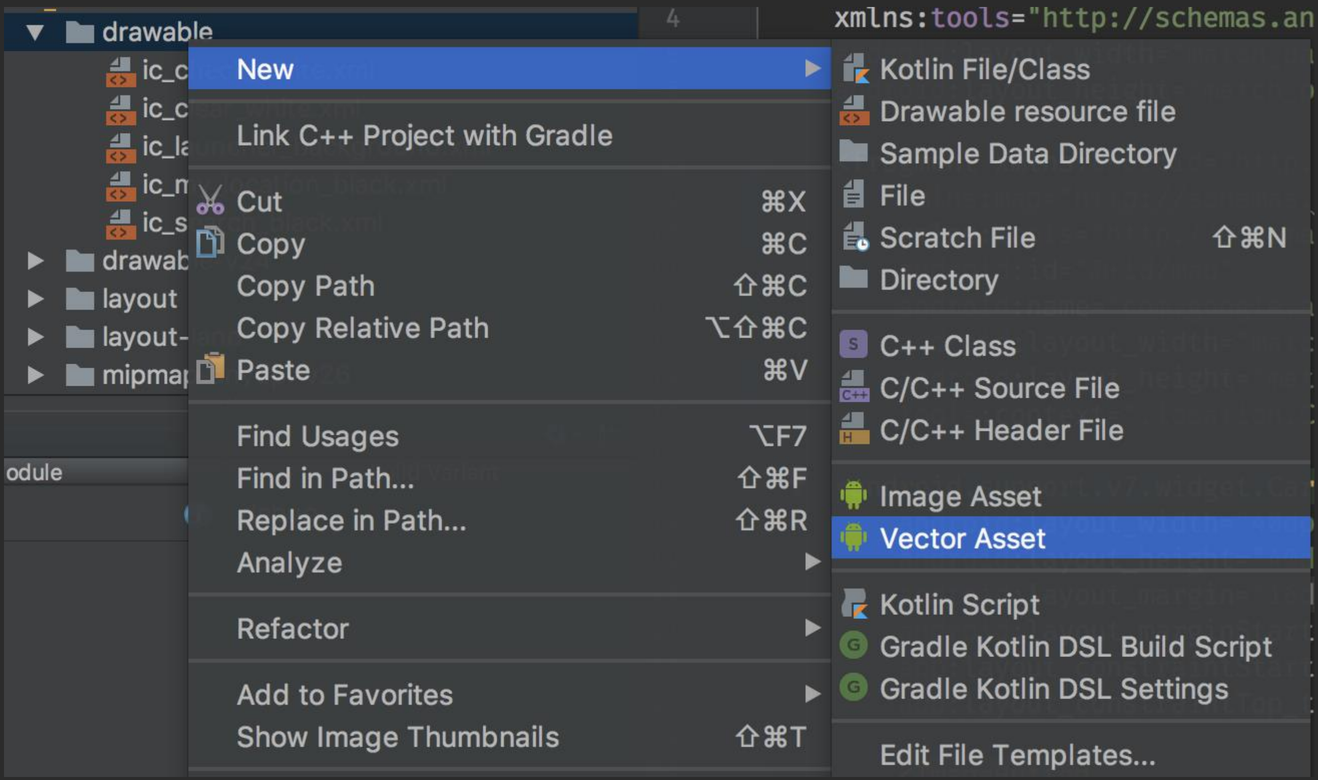


2412 Spring St, Dunn Loring, VA 22027, USA

✓ 2412 SPRING ST, DUNN LORING, VA 22027, USA

Once we have a ConstraintLayout, then we can position buttons, etc. on the layout as we are used to.

SVGs for Button Icons



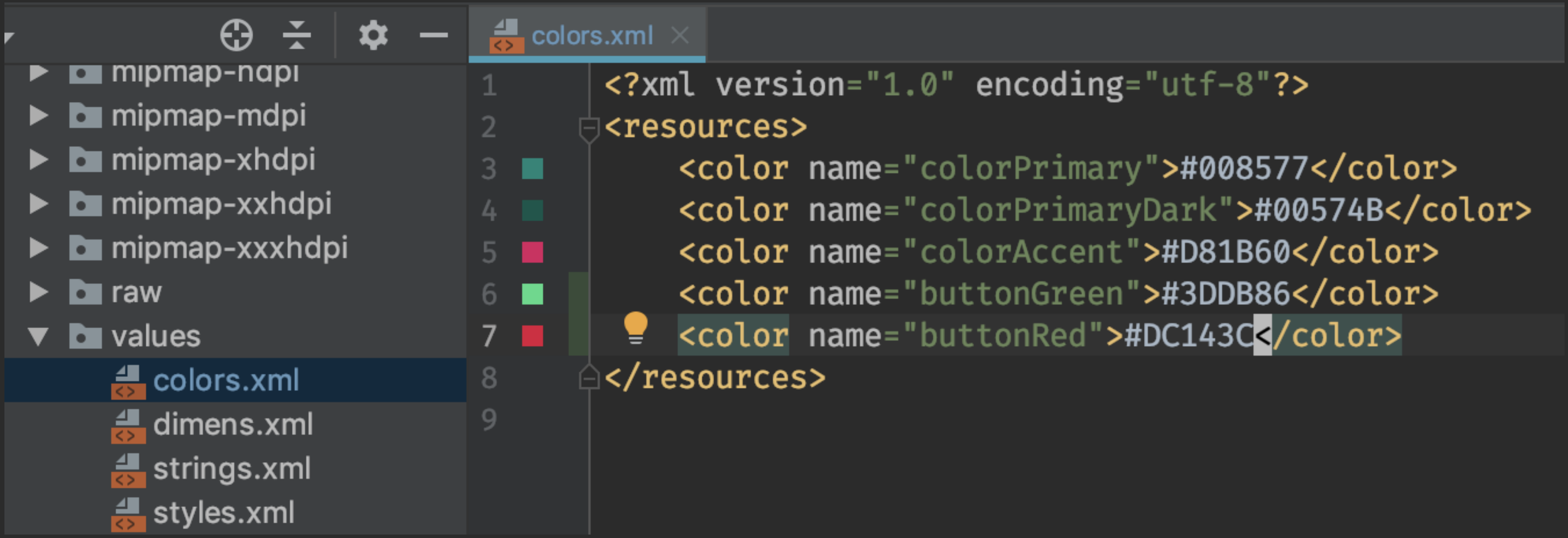
Images

If you want to add your own images, vectors / SVGs are preferred (for scaling), but you can use JPGs / PNGs instead.

- Importing Vectors
- Adding non-vector images
    - Recommended: supply pre-scaled images too

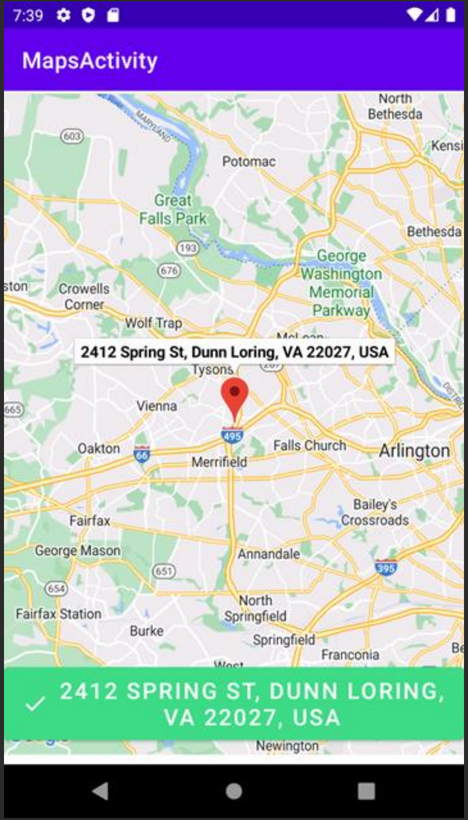
Colors

Like strings, there is also a colors.xml file that you can add custom colors into and reference using @color/...



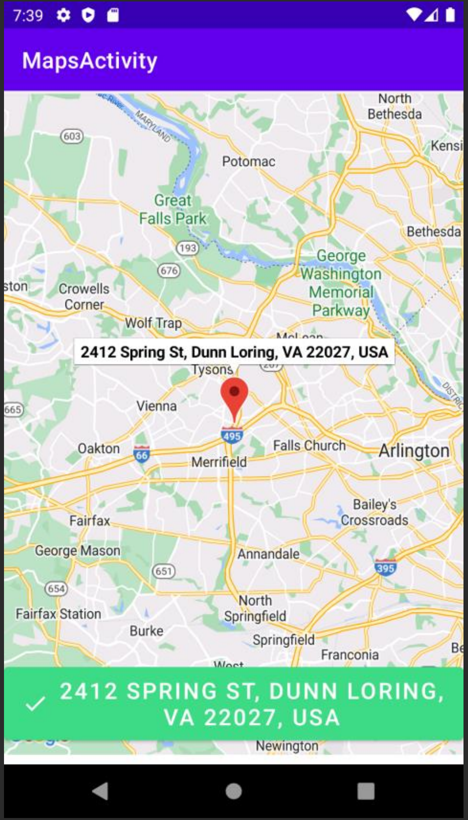
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#008577</color>
    <color name="colorPrimaryDark">#00574B</color>
    <color name="colorAccent">#D81B60</color>
    <color name="buttonGreen">#3DDB86</color>
    <color name="buttonRed">#DC143C</color>
</resources>
```

Button Icons



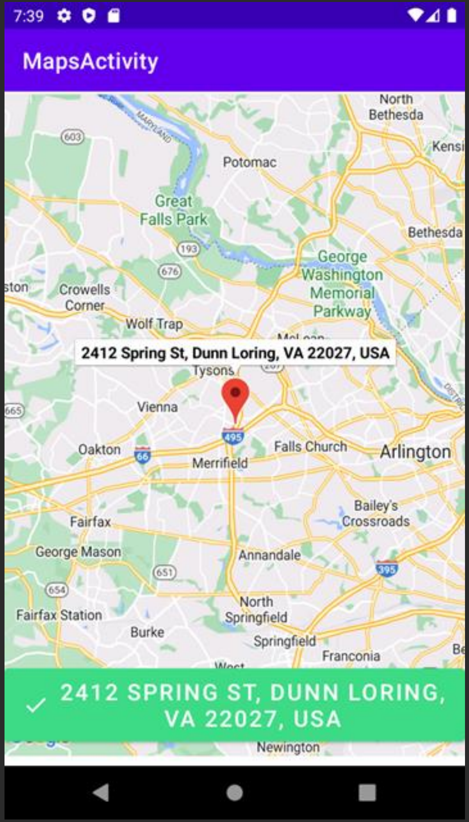
```
<ImageButton
    android:id="@+id/imageButton"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@drawable/ic_my_location"  />

<Button
    android:id="@+id/confirm"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginEnd="16dp"
    android:layout_marginBottom="16dp"
    android:backgroundTint="@color/buttonRed"
    android:text="@string/choose_location"
    app:icon="@drawable/ic_close"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"  />
```

Button Icons



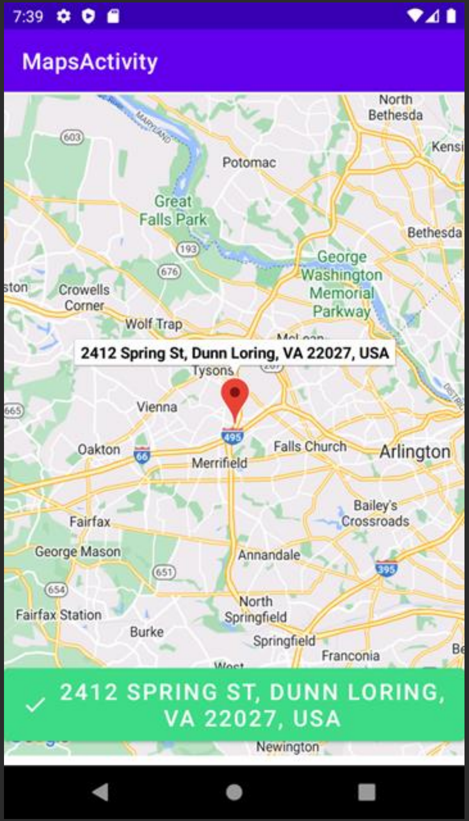
```
<ImageButton
   android:id="@+id/imageButton"
   android:layout_width="50dp"
   android:layout_height="50dp"
   android:layout_marginStart="16dp"
   android:layout_marginTop="16dp"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintTop_toTopOf="parent"
   app:srcCompat="@drawable/ic_my_location" />

<Button
   android:id="@+id/confirm"
   android:layout_width="0dp"
   android:layout_height="wrap_content"
   android:layout_marginStart="16dp"
   android:layout_marginEnd="16dp"
   android:layout_marginBottom="16dp"
   android:backgroundTint="@color/buttonRed"
   android:text="@string/choose_location"
   app:icon="@drawable/ic_close"
   app:layout_constraintBottom_toBottomOf="parent"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintStart_toStartOf="parent" />
```

Customizing the MapsActivity



Next, we just need to update the button to switch colors / text depending on the UI.

Customizing the MapsActivity



2412 Spring St, Dunn Loring, VA 22027, USA

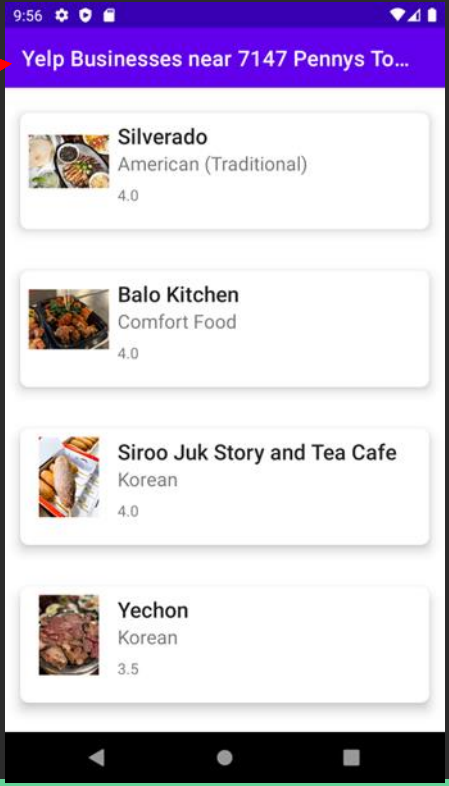
✓ 2412 SPRING ST, DUNN LORING, VA 22027, USA

```kotlin
private fun updateCurrentAddress(address: Address) {
    currentAddress = address

    confirm.text = address.getAddressLine(0)
    confirm.icon = ContextCompat.getDrawable(this, R.drawable.ic_check)
    confirm.setBackgroundColor(getColor(R.color.buttonGreen))
    confirm.isEnabled = true
}
```

Serializable / Parcelable

Sending the Address to Yelp Listing Activity

Goal: we want to send the Address result from the MapsActivity to the Yelp listings.



9:56

Yelp Businesses near 7147 Pennys To...

Silverado
American (Traditional)
4.0

Balo Kitchen
Comfort Food
4.0

Siroo Juk Story and Tea Cafe
Korean
4.0

Yechon
Korean
3.5

Sending the Address to Yelp Listing Activity

We’ve seen how we can use an Intent to pass data between Activities.

```
val intent = Intent(this, Yelp Listings Activity::class.java)
intent.putExtra("latitude", latLng.latitude)
intent.putExtra("longitude", latLng.longitude)
intent.putExtra("title", title)
startActivity(intent)
```

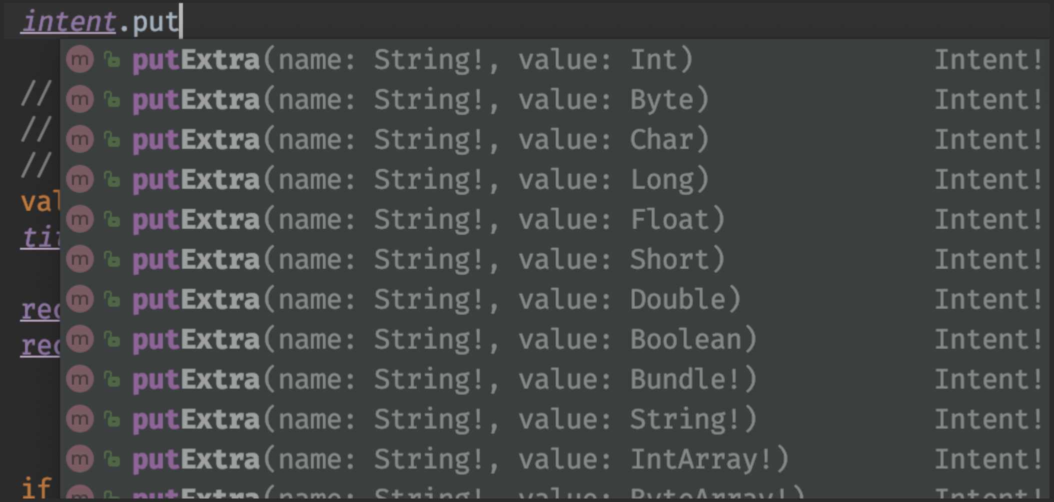
Sending the Address to Yelp Listings Activity

What if we wanted to send a full object / data class to another activity?

```
val firstAddress = results[0]

// ...


val intent = Intent(this, YelpListingsActivity::class.java)
intent.putExtra("address", firstAddress)
startActivity(intent)
```

Passing Complex Objects in an Intent

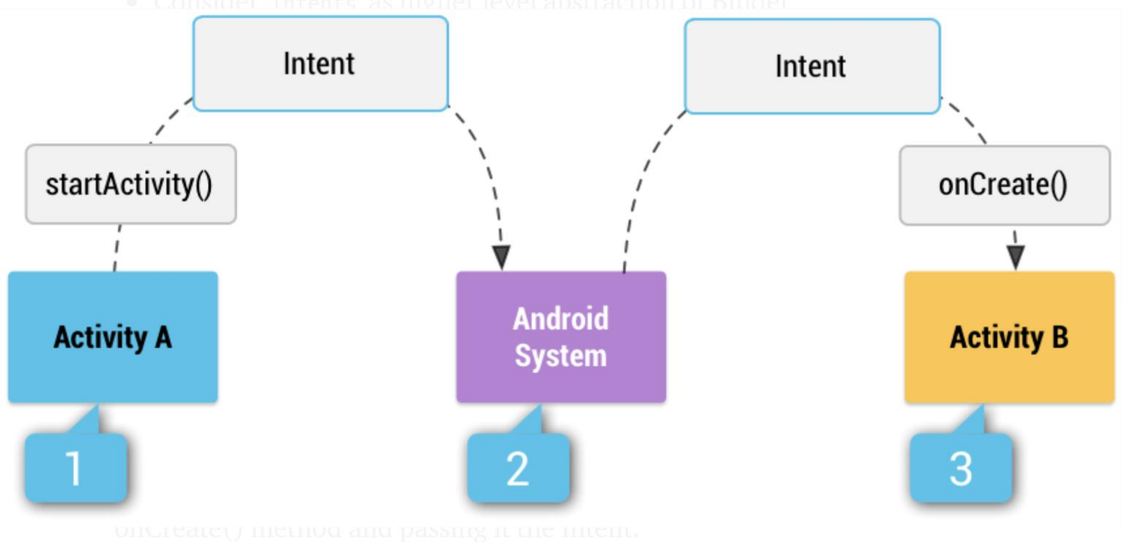
You’re restricted to only putting primitives (String, Boolean, Int, etc.) in an Intent.



```
intent.put
putExtra(name: String!, value: Int)            Intent!
putExtra(name: String!, value: Byte)           Intent!
putExtra(name: String!, value: Char)           Intent!
putExtra(name: String!, value: Long)           Intent!
putExtra(name: String!, value: Float)          Intent!
putExtra(name: String!, value: Short)          Intent!
putExtra(name: String!, value: Double)         Intent!
putExtra(name: String!, value: Boolean)        Intent!
putExtra(name: String!, value: Bundle!)        Intent!
putExtra(name: String!, value: String!)        Intent!
putExtra(name: String!, value: IntArray!)      Intent!
putExtra(name: String!, value: ByteArray!)     Intent!
```

Passing Complex Objects in an Intent

Why? Your app is essentially communicating with another process (the OS) about a new Activity it wants to launch.



Passing Complex Objects in an Intent

Why? Your app is essentially communicating with another process (the OS) about a new Activity it wants to launch.

Under-the-hood, this requires inter-process communication, and any data to-be-passed needs to be in a primitive form.

Passing Complex Objects in an Intent

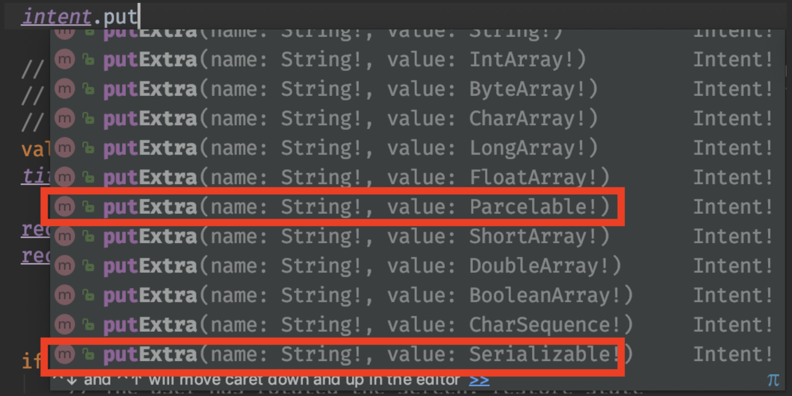
If you want to send a an actual object (e.g. Tweet, Address, etc.) your options are:

- Sending just the “pieces” you need and reconstructing the object on the other side.

Passing Complex Objects in an Intent

If you want to send a an actual object (e.g. Tweet, Address, etc.) your options are:

- Sending just the "pieces" you need and reconstructing the object on the other side.
- Using Serializable / Parcelable



```
intent.put
putExtra(name: String!, value: String!)         Intent!
putExtra(name: String!, value: IntArray!)       Intent!
putExtra(name: String!, value: ByteArray!)      Intent!
putExtra(name: String!, value: CharArray!)      Intent!
putExtra(name: String!, value: LongArray!)      Intent!
putExtra(name: String!, value: FloatArray!)     Intent!
putExtra(name: String!, value: Parcelable!)     Intent!
putExtra(name: String!, value: ShortArray!)     Intent!
putExtra(name: String!, value: DoubleArray!)    Intent!
putExtra(name: String!, value: BooleanArray!)   Intent!
putExtra(name: String!, value: CharSequence!)   Intent!
putExtra(name: String!, value: Serializable!)   Intent!
^↓ and ^↑ will move caret down and up in the editor >>     π
```

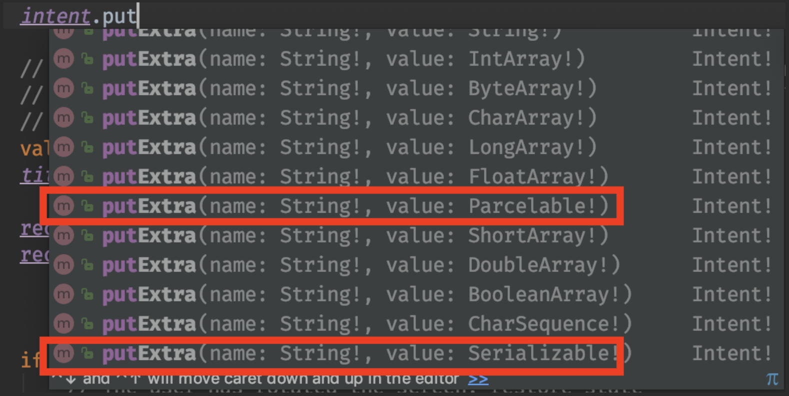
Passing Complex Objects in an Intent

If you want to send a an actual object (e.g. Tweet, Address, etc.) your options are:

- Sending just the “pieces” you need and reconstructing the object on the other side.
- Using Serializable / Parcelable
- Using components outside of Activities to store data (e.g. in-memory caches, files, DBs) - this is where we get into design / software / architecture patterns...

Passing Complex Objects in an Intent

Let’s look at the Serializable / Parcelable route, which takes advantage of Intents we’ve already learned.



```
intent.put
putExtra(name: String!, value: String!)          Intent!
putExtra(name: String!, value: IntArray!)        Intent!
putExtra(name: String!, value: ByteArray!)       Intent!
putExtra(name: String!, value: CharArray!)       Intent!
putExtra(name: String!, value: LongArray!)       Intent!
putExtra(name: String!, value: FloatArray!)      Intent!
putExtra(name: String!, value: Parcelable!)      Intent!
putExtra(name: String!, value: ShortArray!)      Intent!
putExtra(name: String!, value: DoubleArray!)     Intent!
putExtra(name: String!, value: BooleanArray!)    Intent!
putExtra(name: String!, value: CharSequence!)    Intent!
putExtra(name: String!, value: Serializable!)    Intent!
^↓ and ^↑ will move caret down and up in the editor >>      π
```

Serializable / Parcelable

Android provides two ways to make custom objects “portable”

Serializable / Parcelable

Android provides two ways to make custom objects “portable”

- Implement the Serializable interface.
- Implement the Parcelable interface.

Serializable / Parcelable

Android provides two ways to make custom objects “portable”

- Implement the Serializable interface.
  - Android (Java) will figure out, at runtime, how to convert your class into a stream of bytes.
- Implement the Parcelable interface.

Serializable / Parcelable

Android provides two ways to make custom objects “portable”

- Implement the Serializable interface.
    - Android (Java) will figure out, at runtime, how to convert your class into a stream of bytes.
    - Very easy -- just implement the interface.
- Implement the Parcelable interface.

Serializable / Parcelable

Android provides two ways to make custom objects “portable”

- Implement the Serializable interface.
  - Android (Java) will figure out, at runtime, how to convert your class into a stream of bytes.
  - Very easy -- just implement the interface.
  - Inefficient / slow -- uses Reflection.
- Implement the Parcelable interface.

Serializable / Parcelable

Android provides two ways to make custom objects “portable”

- Implement the Serializable interface.
- Implement the Parcelable interface.
    - The “preferred” way. You have to write the code to deconstruct (and reconstruct) your class to / from primitives.

Serializable / Parcelable

Android provides two ways to make custom objects “portable”

- Implement the Serializable interface.
- Implement the Parcelable interface.
  - The “preferred” way. You have to write the code to deconstruct (and reconstruct) your class to / from primitives.
  - Fast, but more work to do.

Serializable / Parcelable

Android provides two ways to make custom objects “portable”

- Implement the Serializable interface.
- Implement the Parcelable interface.
    - The “preferred” way. You have to write the code to deconstruct (and reconstruct) your class to / from primitives.
    - Fast, but more work to do.
    - Example

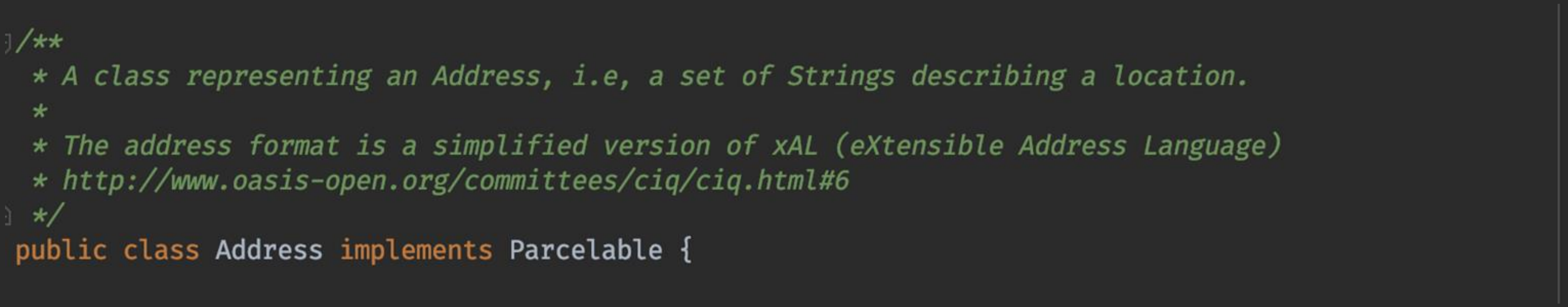
Serializable / Parcelable

For example, if we wanted to make our Yelp data class Serializable.

```
data class YelpBusiness  (
    val restaurant_name: String,
    val category: String,
    val rating: Double,
    val icon: String,
    val url: String
    ) : Serializable
```

Serializable / Parcelable

The Address class returned by the Geocoder is actually already Parcelable, so we don't need to do anything.



```
/**
 * A class representing an Address, i.e, a set of Strings describing a location.
 *
 * The address format is a simplified version of xAL (eXtensible Address Language)
 * http://www.oasis-open.org/committees/ciq/ciq.html#6
 */
public class Address implements Parcelable {
```

Serializable / Parcelable

We can put Parcelable or Serializable objects into an Intent.

```
// Create a new Intent object
val intent = ...

// Address is Parcelable, so it can be put into the Intent
val address = ...
intent.putExtra("address", address)

// Tweet is Serializable, so it can be put into the Intent
val tweet = ...
intent.putExtra("tweet", tweet)
```

Serializable / Parcelable

To retrieve a Parcelable or Serializable entry out of an Intent:

- For Parcelable, use the generic parameter to cast
- For Serializable, cast the data with *as*

```
val address = intent.getParcelableExtra<Address>("address")
```

```
val yelp = intent.getSerializableExtra(“yelpBusiness") as
YelpBusiness
```

Serializable / Parcelable

The **ArrayList** class is Serializable, so you can use it to pass **lists of data** between Activities too.

- For Parcelable, use *putParcelableArrayListExtra*.

```
// In the sending Activity
val listOfYelps = arrayListOf(getFakeYelps())
intent.putExtra(“yelps", listOfYelps)

// In the receiving Activity
val listOfYelps = intent.getSerializableExtra(“yelps") as List<YelpBusiness>
```

Serializable / Parcelable

The **Address** class from the geocoder is already **Parcelable**, so we can use it here to pass it to the YelpsListingActivity.

```
val currentAdress: Address = ….
// ...

val intent = Intent(this@MapsActivity, YelpsListings::class.java)
intent.putExtra("address", currentAddress)
startActivity(intent)
```

Serializable / Parcelable

The Yelp Listings Activity can retrieve the Address and display the city name.

```
val intent = getIntent()
// We're now passing a full address rather than a location String
// val locationName = intent.getStringExtra("LOCATION")

// Need !! to force the data to be non-null, we’d want the app to crash otherwise
val address = intent.getParcelableExtra<Address>("address")!!
```

DEPRECATED…..
BUT

Serializable / Parcelable

The Yelp Listings Activity can retrieve the Address and display the city name.

```
// Per the docs - the "locality" resolves to the city name of the Address
// https://developer.android.com/reference/android/location/Address#getLocality()
// But it's potentially null, so we have to have a fallback String in mind
val cityName = address.locality ?: "Unknown"

val localizedString = getString(R.string.tweets_title, cityName)

setTitle(localizedString)
```

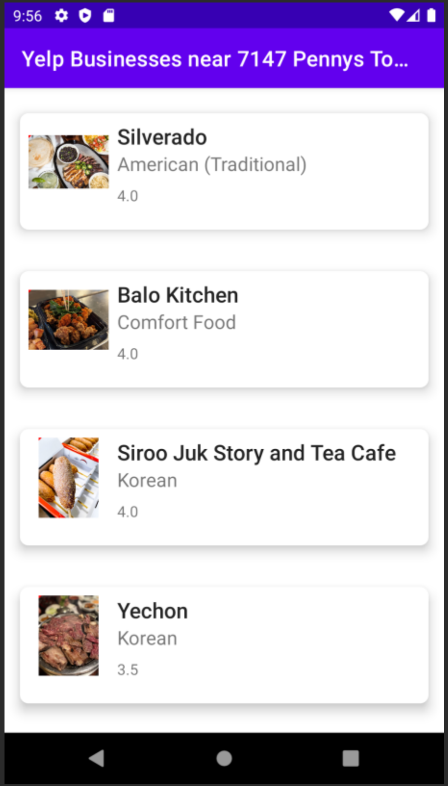
Sharing Data between Activities

Intents pass data in memory, which is favorable compared to SharedPreferences, which incurs File I/O to save the data.

Both methods are still cumbersome:

- I’ll touch briefly on this later in the semester, but you can also cache your data *outside* of Activities.
  - There was ways to “architect” your codebase to make data sharing easier across your app.

Picasso - Usage



9:56

Yelp Businesses near 7147 Pennys To...

Silverado
American (Traditional)
4.0

Balo Kitchen
Comfort Food
4.0

Siroo Juk Story and Tea Cafe
Korean
4.0

Yechon
Korean
3.5

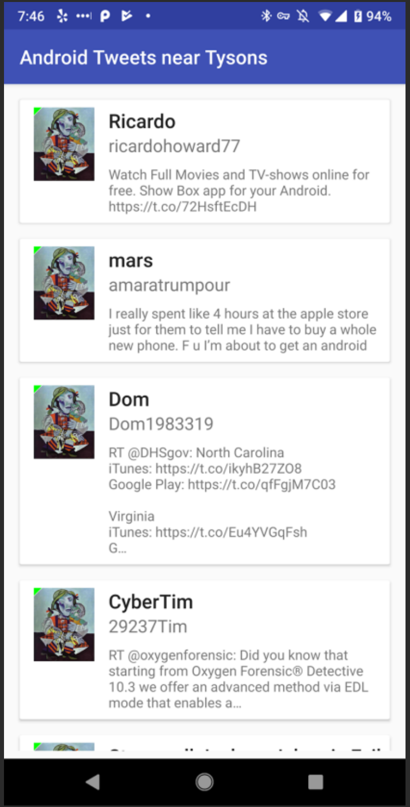
Let’s look at actually loading the users’ profile pictures into our RecyclerViews.

Separate Slide Deck on Blackboard

Picasso - Usage

```
// First, add this to app’s build.gradle to automatically add Picasso to the project
implementation 'com.squareup.picasso:picasso:2.71828'
```

Picasso - Usage



7:46 94%

Android Tweets near Tysons

Ricardo
ricardohoward77
Watch Full Movies and TV-shows online for free. Show Box app for your Android. https://t.co/72HsftEcDH

mars
amaratrumpour
I really spent like 4 hours at the apple store just for them to tell me I have to buy a whole new phone. F u I'm about to get an android

Dom
Dom1983319
RT @DHSgov: North Carolina
iTunes: https://t.co/ikyhB27ZO8
Google Play: https://t.co/qfFgjM7C03

Virginia
iTunes: https://t.co/Eu4YVGqFsh
G...

CyberTim
29237Tim
RT @oxygenforensic: Did you know that starting from Oxygen Forensic® Detective 10.3 we offer an advanced method via EDL mode that enables a...

```
// So we can see loaded from network vs. cache
Picasso.get().setIndicatorsEnabled(true)

Picasso.get()
  .load("https://i.imgur.com/DvpvklR.png")
  .into(holder.icon)
```

OAuth

Open Authorization

- OAuth is a security protocol that allows applications to access a user's data on other services without exposing the user's credentials to the application.

- It relies on authorization and access tokens to grant controlled access to resources. It's crucial for enabling third-party integrations and ensuring user privacy and security online.

Yelp API

Last time we used a hardcoded value for the “Authorization” header - what was that coming from?

```
val request = Request.Builder()
    .url("https://api.yelp.com/...")
    .header("Authorization", "Bearer AA...")
    .build()
```

Sending API Keys in Requests

Generally, when using a third-party API you need to specify your API key in some way.

- News uses an “apiKey” *parameter* for this.
- WMATA uses an “api_key” *header* for this.
- etc.

Sending API Keys in Requests

Generally, when using a third-party API you need to specify your API key in some way.

- News uses an “apiKey” *parameter* for this.
- WMATA uses an “api_key” *header* for this.
- etc.

There’s not a consistent way to do this across all companies - *read the API documentation*.

Twitter API

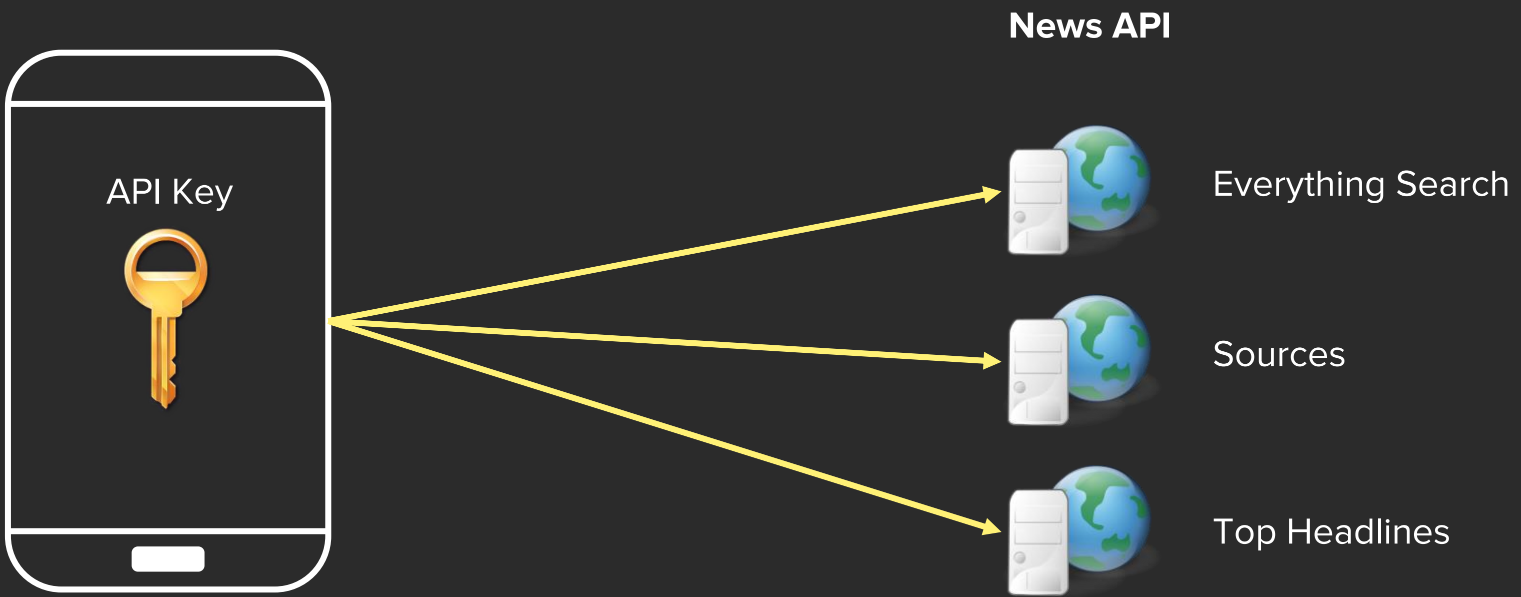
Twitter uses an “Authorization” header, but the value you pass for it is actually *not* your API key.

```
val request = Request.Builder()
    .url("https://api.twitter.com/...")
    .header("Authorization", "Bearer AA...")
    .build()
```

API Access

Why? Let’s use the News API as an example - your API key gives you access to *all* of the News APIs’ endpoints.

API Access - News API



API Access Restrictions

But what if API access needs to be selectively restricted?

API Access Restrictions

But what if API access needs to be selectively restricted?

- Twitter has APIs to Search Tweets… but also to: post Tweets, send direct messages, follow / unfollow / block, etc.

API Access Restrictions

Our Android Tweets app should be allowed to call the Search Tweets API, which indexes publicly-available information...

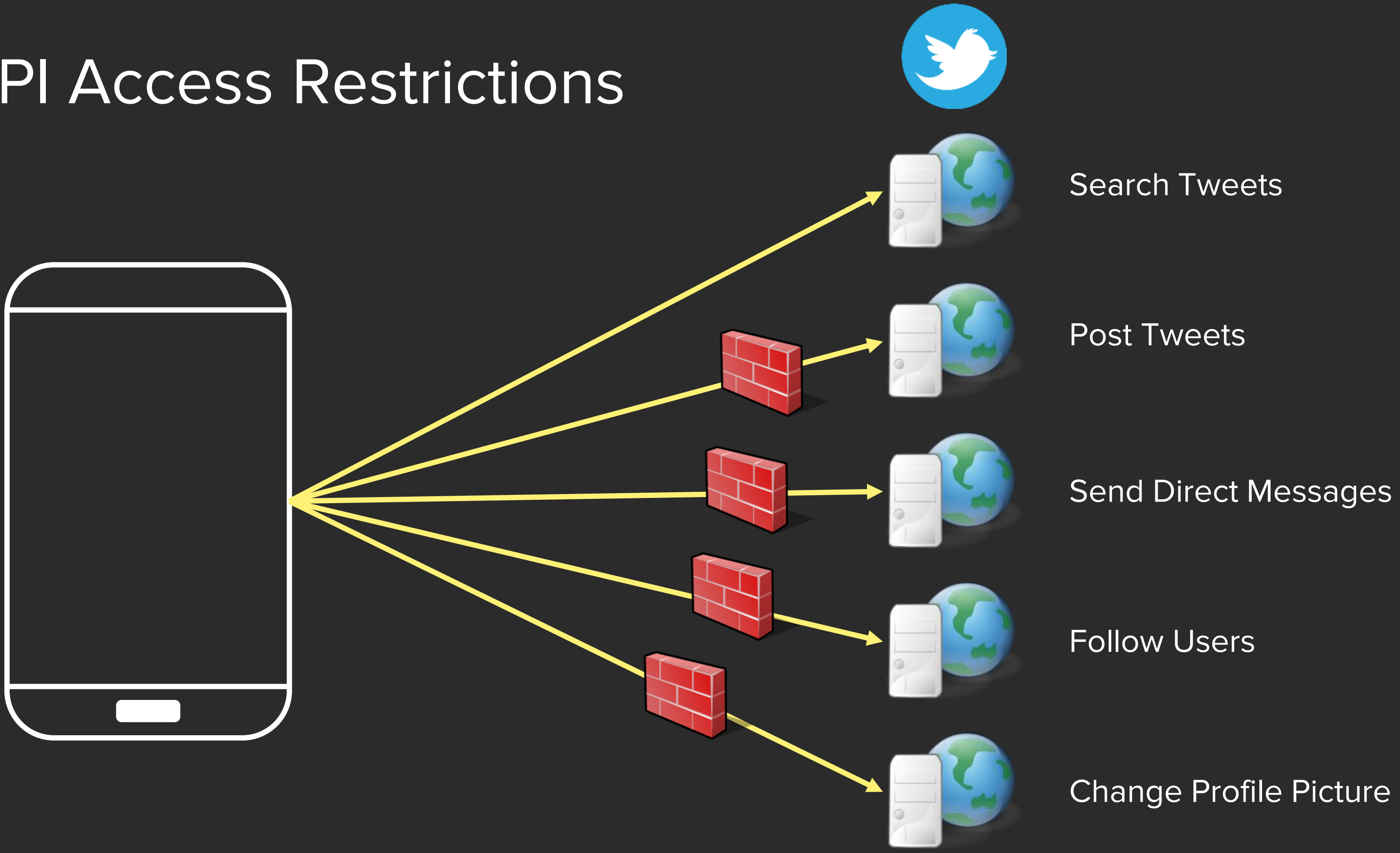
... but it should **not** be allowed to post Tweets, etc. on a user's behalf.

API Access Restrictions

Our Android Tweets app should be allowed to call the Search Tweets API, but **not** post Tweets, etc.

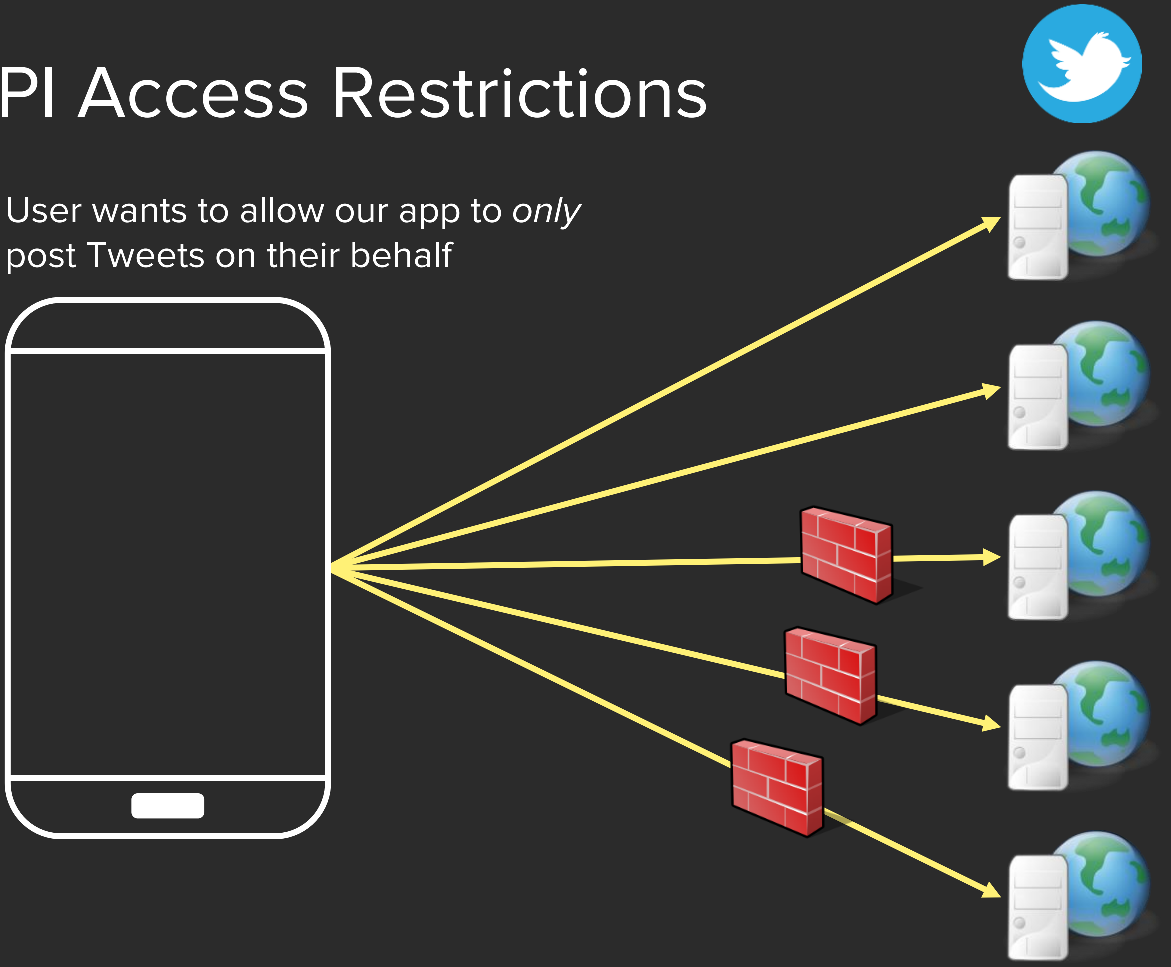
- You might want this to be dynamic, maybe a user should be able to *grant access* to their account to post Tweets.
    - i.e. the API key alone isn’t sufficient enough to dynamically apply user-based permissions.

API Access Restrictions



API Access Restrictions

User wants to allow our app to *only* post Tweets on their behalf



OAuth

- All of Twitter’s APIs are protected by a standard called **OAuth** (Open Authorization), which controls access to the APIs by *external* apps (like Android applications).

OAuth

- Basic idea: instead of using your API Key directly, you tell the server what you plan to do, and the server will grant you a token (random string) to send on every subsequent API call.

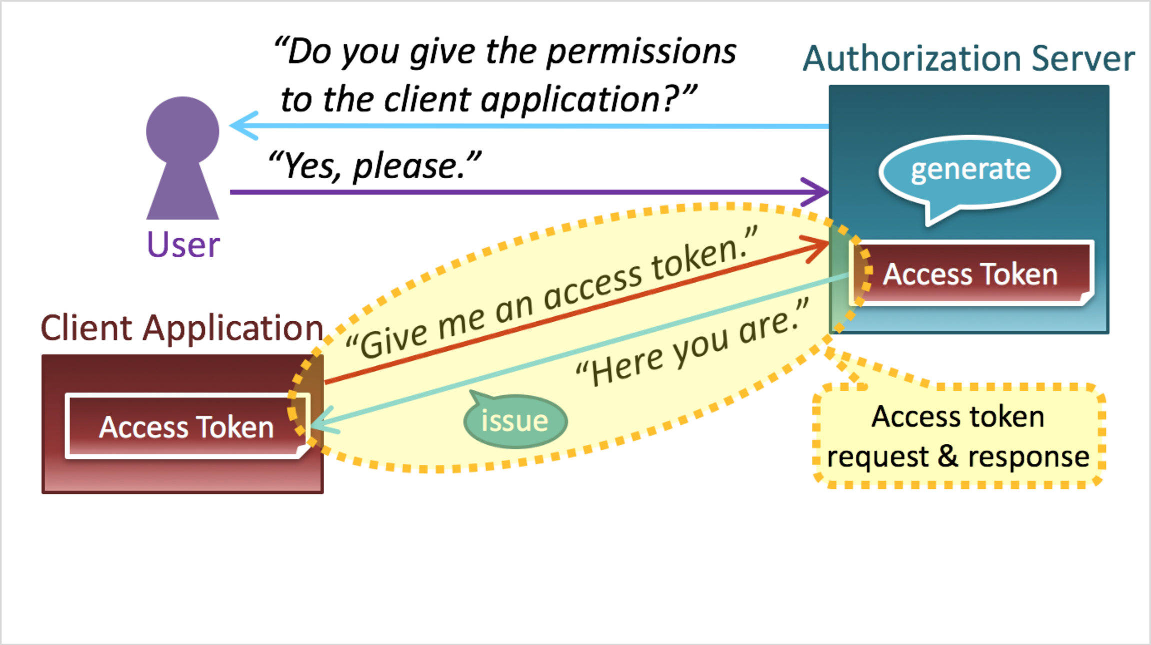
OAuth

- Basic idea: instead of using your API Key directly, you tell the server what you plan to do, and the server will grant you a token (random string) to send on every subsequent API call.
  - The server can look at the token later to figure out what access you are supposed to have.

OAuth

- Basic idea: instead of using your API Key directly, you tell the server what you plan to do, and the server will grant you a token (random string) to send on every subsequent API call.
    - The server can look at the token later to figure out what accesses you are supposed to have.
    - If you try and access an API that isn’t allowed by your token, the server will block the request.

User OAuth


"Do you give the permissions to the client application?"
Authorization Server
generate
"Yes, please."
User
Access Token
"Give me an access token."
Client Application
"Here you are."
Access Token
issue
Access token request & response

OAuth

- Bottom line: the goal is to get a OAuth token issued by the server, to pass on subsequent API calls.
    - The server uses the token to determine what data you’re allowed to access / manipulate.

OAuth

- Bottom line: the goal is to get a OAuth token issued by the server, to pass on subsequent API calls.
    - The server uses the token to determine what data you’re allowed to access / manipulate.
    - If the data in question is user-specific data, then the user needs to login first and grant access.

Questions?

Quiz 2

- Quiz 2 next week in class
- **Team competition next week (Read Ahead provided)**

Project 1 - Final Submission

- We’ll talk about the final submission details next week.

- Main pieces of functionality left for the final deliverable:
    - Data Persistence requirements
    - Image loading requirements
    - News Results screen
      (e.g. Search “Cryptocurrency” ➔ Choose Sources ➔ View Results)
    - Top Headlines screen
      (the Paging requirements may be a little tricky)

Helpful Links

- A Beginner’s Tutorial for Understanding RESTful API
- OkHttp
    - OkHttp Android Example