Practical class # 12 – Word List App

1) Download starter code from Google GitHub.

https://github.com/google-developer-training/android-basics-kotlin-words-app

2) Open the code in Android Studio and import the new project

The structure is the following:

1. LetterAdapter is used by the RecyclerView in MainActivity. Each letter is a button with an onClickListener, which is currently empty. This is where you'll handle button presses to navigate to the DetailActivity.
2. WordAdapter is used by the RecyclerView in DetailActivity to display a list of words. While you can't navigate to this screen quite yet, just know that each word also has a corresponding button with an onClickListener. This is where you'll add code that navigates to the browser, to show a definition for the word.

3) Set up the explicit intent

1. Open LetterAdapter.kt and scroll down to onBindViewHolder(). Below the line to set the button text, set the onClickListener for holder.button.

```
holder.button.setOnClickListener {

}
```

2. Then get a reference to the context.

```
val context = holder.itemView.context
```

3. Create an Intent, passing in the context and the class name of the destination activity.

```
val intent = Intent(context, DetailActivity::class.java)
```

The name of the activity you want to show is specified with DetailActivity::class.java. An actual DetailActivity object is created behind the scenes.

4. Call the putExtra method, passing in "letter" as the first argument and the button text as the second argument.

```
intent.putExtra("letter", holder.button.text.toString())
```

5. Call the startActivity() method on the context object, passing in the intent.

```
context.startActivity(intent)
```

4) Set up DetailActivity

In the onCreate method of DetailActivity, after the call to setContentView, replace the hard coded letter with code to get the letterId passed in from the intent.

```
val letterId = intent?.extras?.getString("letter").toString()
```

5) Cleaning Up

There's a handy Kotlin feature that can be used to separate your constants and make them usable without a particular instance of the class called companion objects. A companion object is similar to other objects, such as instances of a class. However, only a single instance of a companion object will exist for the duration of your program, which is why this is sometimes called the singleton pattern.

1. In DetailActivity, just above onCreate, add the following:

```
companion object {
        const val LETTER = "letter"
}
```

2. To use the new constant, update your hard coded letter call in onCreate() as follows:

```
val letterId = intent?.extras?.getString(LETTER).toString()
```

3. Switch over to LetterAdapter, and modify the call to putExtra to use the new constant.

```
intent.putExtra(DetailActivity.LETTER, holder.button.text.toString())
```

6) Set up an Implicit Intent

This will create an implicit intent to search a word on Google.

1. In DetailActivity, modify the companion object to add a new constant, SEARCH_PREFIX. This is the base URL for a Google search.

```
companion object {
        const val LETTER = "letter"
        const val SEARCH_PREFIX = "https://www.google.com/search?q="
}
```

2. Open up WordAdapter and in the onBindViewHolder() method, call setOnClickListener() on the button. Start by creating a Uri for the search query. When calling parse() to create

a Uri from a String, you need to use string formatting so that the word is appended to the SEARCH_PREFIX

```
holder.button.setOnClickListener {
        val queryUrl: Uri = Uri.parse("${DetailActivity.SEARCH_PREFIX}${item}")
}
```

3. Initialize the new Intent and start it

```
val intent = Intent(Intent.ACTION_VIEW, queryUrl)
context.startActivity(intent)
```