Networking With Retrofit

In this section you will learn how to use retrofit to make calls to REST APIs in android apps.
This is achieved in the following simple steps

1. Add the retrofit dependencies to your app level build.gradle file

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
```

2. Create an API client that will create an instance of retrofit that we can use in our project. Do this in a kotlin file called `ApiClient.kt` inside your main application package.

```
object ApiClient {
    var retrofit = Retrofit.Builder()
        .baseUrl("https://jsonplaceholder.typicode.com/")
        .addConverterFactory(GsonConverterFactory.create())
        .build()

    fun <T> buildApiClient(apiInterface: Class<T>): T {
        return retrofit.create(apiInterface)
    }
}
```

The buildApiClient function takes in our ApiInterface file and uses it to create our instance of Retrofit.

3. Create an ApiInterface file. We define the REST API endpoints in our ApiInterface file. This covers the url to which our request will be sent, the appropriate HTTP verb and a function definition which takes in any parameters that the endpoint expects as well as the return type of the data that we expect from the API.

```
interface ApiInterface {
    @GET("posts")
    fun getPosts(): Call<List<Post>>
}
```

The `Post` type is a data class we have designed in our app to hold data about a single post resource from the web API

```
data class Post(var userId: Int, var id: Int, var title: String, var
body: String)
```

4. Finally in our activity we create a function to fetch the data and cal it. We can display the data in the activity if we want to.

```
class MainActivity : AppCompatActivity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    getPosts()
  }

  fun getPosts(){
    val retrofit = ApiClient.buildApiClient(ApiInterface::class.java)
    val request = retrofit.getPosts()
    request.enqueue(object: Callback<List<Post>>{
      override fun onResponse(call: Call<List<Post>>, response:
Response<List<Post>>) {
        if (response.isSuccessful){
          var posts = response.body()
          Toast.makeText(baseContext, "${posts!!.size} posts", Toast.
LENGTH_LONG).show()
        }
      }

      override fun onFailure(call: Call<List<Post>>, t: Throwable) {

      }
    })
  }
}
```