Download the "TryKotlin" project

In IntelliJ:

1. Menu Option : `File > New > Project from Version Control > GitHub`
2. Git Repository Url: `git@github.com:LeedsCodeDojo/TryKotlin.git`

Project contains:

- `src` folders
- `build.gradle` which defines external dependencies
- Cheatsheet which provides syntax hints & tips

Problem 1 - Hello World

1. Create a main() function which just prints "Hello, World".
2. Extend the function to personalise the output so that a persons name can be supplied as the first argument.

Problem 2 - Fizz Buzz

1. Create a function which will return:
    - 'Fizz' if number is divisible by 3
    - 'Buzz' if number is divisible by 5
    - 'Fizz Buzz' if number is divisible by both 3 & 5
    - otherwise return the number
2. Invoke this function for a supplied of numbers i.e. 1..50
3. Extend your function to allow 3 & 5 to be optionally replaced by alternative integers

Hint - x % 2 will return zero if x is divisible by 2

Problem 3 - Simplification of Java Classes

3 Java classes (JAddress, JCompany, and JPerson) have been created in dojo.leeds.problem3.

Convert these classes to Kotlin data classes to see how much boilerplate code can be removed.

Remember that Kotlin allows multiple classes to be defined in a single file.

\pagebreak

Problem 4 - Working with Collections

1. Create a Pupil Data class to hold the details of each line and their associate scores in English, Maths and Science.
    - Grading works as follows: 80%+ = A, 70->79% = B, 60->69% = C, 50->59% = D, 45-50% = E, <45% = F
2. Load Details of Pupils from 'endOfYearsResults.csv'
3. Create functions to answer the following questions:
    - Count the number of pupils in the file.
    - Count the number of unique pupil names.
    - How many pupils are there in each class?
    - Which class has the highest average score per pupil?
    - Which students were awarded 3 grade 'A's?
    - Which students were awarded at least a 'C' in all subjects?
    - Which were the highest scoring Boy & Girl in each class?

Problem 5 - Got to Catch Them All (or at least 1 of them)

This problem uses Kotlin to create a basic REST client using JSON over HTTP.
We will access the publicly available database of Pokemon and use a couple of libraries to assist us.

Details of the Pokemon API can be found here: https://pokeapi.co/docsv2/#pokemon-section

1. Create a Pokemon Data Class to hold the details of a Pokemon (start with just the basic details e.g. id, name, height, width)
2. Test that JSON parser can be used to transform the supplied Pokemon JSON file (TBD - give path) into an instance of your Pokemon data class
3. Create a function which will send an HTTP Get Request to … to retrieve the details of a specific Pokemon by ID
4. Transform the JSON in the response into an instance of your Pokemon data class.
5. Extend your Pokemon data class to add additional properties.

The base project has already included dependencies to the Moshi Json Parser and the Fuel HTTP Networking library which can be used.
NB The Pokemon API requires a User-Agent header to be supplied on all requests, any value can be supplied

```
val moshi = Moshi.Builder()
```

```
            .add(KotlinJsonAdapterFactory())
            .build()

// Create an adapter which can convert Pokemon classes to/from Json
val pokemonAdapter = moshi.adapter(Pokemon::class.java)
```

```
val headers = mapOf("User-Agent" to "Mozilla/5.0")
val (request, response, result) = "https://myurl/..".httpGet().header(headers).resp
```

\pagebreak

Useful Links

- Pokemon API - https://pokeapi.co/docsv2/
- Moshi Json Parser - https://github.com/square/moshi
- Fuel Http Networking Library - https://github.com/kittinunf/Fuel

Additional Problems

If you have made it this far, try using Kotlin to solve some of the Google Code Jam problems. For example:

- Hegemony : https://code.google.com/codejam/contest/2334486/dashboard

All contests can be found here: https://code.google.com/codejam/past-contests