**WikiCrawler Exercise**

**Preparation**

Clone project: https://github.com/brazzy/meetup\_compFuture.git

**Exercise 1**

Implement the download() method in the class WikiCrawler which returns the contents of a given URL as a String. You have an API class which does the actual downloading, but it works asynchronously with an ugly callback interface.

Adapt it to return a nice CompletableFuture.

Your goal is to make the tests in Exercise1Tests pass.

**Exercise 2**

Now let’s do something with the downloaded content!

Using the other methods in the API class, implement the getFirstLinkInArticle() method of the WikiCrawler – it should take a word, download the raw Wiki markup for the corresponding article on Wikipedia and return the first wiki link found there.

Use the thenApplyAsync() or thenCombineAsync() methods of CompletableFuture to manage the multiple stages of the asynchronous workflow, and method references to make it clean.

In the end, the test in Exercise2Tests should pass.

**Exercise 3**

For the finale, we apply this functionality repeatedly – implement the getCycle() method of the WikiCrawler. We want to follow links until we encounter a cycle, i.e. a sequence of words that keeps repeating. The method returns that cycle.

How can this be implemented asynchronously, using a CompletableFuture? Some sort of recursion, probably. How do we recognize when we’ve encountered a cycle?

The goal: make the tests in Exercise3Tests pass.