Futher Java
Supervision 2

Supervisor: Joe Isaacs (josi2).
All work should be submitted in PDF form with page numbers 36 hours before the supervision to the email `josi2@cam.ac.uk`. If you have any questions on the course please include these at the top of the supervision work and we can talk about them in the supervision. Please include all code that you think is relevant (that you want me to look at in the PDF).

Tasks:

1. Currently your KVS is not persistent (if the server process if killed all the (key,value) pairs are lost).

   - Make your KVS persistent or durable the D of aciD[1].
   - Try removing the strict durability constraint and instead 'try'[2] to be persistent.

   Explain you persistence in detail. Make sure that you are really writing to disk when you think you are. For both persistence schemes measure the performance how much does a strict durability constraint affect performance? Explain your tests.

   What is the space requirement of your durability scheme?

2. How does persistance affect the protocol from last week?

3. Use non-blocking IO (Java NIO) to implement a server as in the previous SV but only using a single thread to handle all conncetions. Can you combine the best of non-blocking IO and threading to increase the servers performance? How do these changes affect queueing time?

4. Define and implement a static method `then` which takes a `function<A,B>` and a function `function<B,C>` and returns a `function<A,C>`

5. Add a method to your two implementations of `Future<T>` called `<U> Future<U> thenCombine(Function<T,Future<U>>)` which allows chaining together two futures F_1 and F_2 such that when F_1.`isDone` holds the result is passed to F_2. Add this to both implementation in the previous question. Try to only start running F_2 once F_1 has completed. Can you combine a naïve thread future with a thread pool thread future?

[1] `https://en.wikipedia.org/wiki/ACID_(computer_science)#Durability_failure`
[2] this means that some data maybe lost on a failure but most is recoverable.