Notes on the Challenge – Aden Haussmann

How I approached the task

- The first thing I noticed was that a fairly well-thought-out algorithm would be needed to correctly parse the commands and text in the .txt file, so I started by pseudo-coding how I planned to solve this part, as I knew the implementation would just be a matter of figuring out Itext's functionality.

Difficulties

- **Itext classes seem to have changed considerably from version 5 to 7**, so a lot of information on StackOverflow, for example, is outdated. This meant I had to spend time looking through the documentation.
- **I had trouble installing the Itext dependencies.** I tried doing it automatically through Maven and editing the pom.xml file myself, but neither seemed to work. The dependencies didn't seem to be downloading to the folder they needed to be in. Eventually I got them working via the pom.xml file.
- **Deciding how to split my program into Classes was tough.** My first solution had all of the logic entirely in the main method. When I began refactoring, I realised I wasn't sure what Classes to create, as there are no obvious real-world objects like a Customer or Bank Account. I settled on a TextFormatter class that contains all the logic for identifying and applying the commands to the text. I am still not certain if this was the ideal implementation, but I like it because it allows the creation of multiple TextFormatter objects, which enables easy output of many PDFs from different .txt files.
- **Unit testing a PDF is tricky.** At first, I wasn't even sure what tests I could write. I ended up testing one property: that the PDF contained exactly the correct text. Admittedly, this is more like System testing than Unit testing, but testing properties of the PDF seemed like a reasonable option, as I didn't have many functions. This property-based test could be used as a template to test more sophisticated properties of the PDF. For example, that each Text item has the correct format properties.

Possible improvements

- Writing more sophisticated property-based tests
- The algorithm I came up with requires 2 long for loops which are similar, one nested within the other. While I think my solution is robust, there may well be a more elegant one. Mine necessitates that the processText function (the function that implements the algorithm) returns void and is therefore difficult to Unit test. Perhaps this can be improved.

Feedback

- I noticed that in the example input text, there are two commands which were left out of the command explanations further down in the document: .large and .normal. It was pretty clear what these were meant to do anyway, but I thought the small discrepancy was worth noting.

Wrap-up

To show my commitment to upholding high standards in my design, I included a Javadoc, which can be found in txttopdf/target/apidocs. I hope this addition, and my program itself, will reflect the hard work I put into the task. Overall, I enjoyed the challenge.