Darshan Vekaria
Project Report - Implementation of Redis

1. **Why did I choose java?**

    - Since the project implementation heavily depends on the working of a **Key-Value pair** mechanism, Java supports data structures like **HashMap** and **TreeMap** to build foundations for such mechanisms.
    - Java provides excellent **error and exception handling** mechanisms. In java, we can catch Exception object to catch all kinds of exceptions. Java also supports a finally block that can be used to do cleanup work(if required).
    - The whole implementation is carried out on a java based framework called **Spring Boot**. Using this framework provides **scalability** to this implementation. The framework also has a great community to provide support to the developers in solving problems.
    - Thus, with java supporting a variety of features, it makes an ideal choice to implement a key-value store mechanism like Redis.

2. **What are the further improvements that can be made to make it efficient?**

    The implementation done here lacks the property of removing the key from the store after the expiry time. Some logic for Key removal after the given expiry time can be thought of by taking more time into developing this. Also, here a text file is treated as a database to store the data when the application ends and to get the data stored previously when the application starts. This proves to be an inefficient form of storing the data and also time consuming since the Maps are required to read/write into the file each time the system starts/ends. Therefore, using a proper database will also help in supporting some important database management properties Atomicity, Durability, Isolation and Consistency.

3. **What data structures I have used and why?**

    - Throughout the whole implementation, I have made the use of ArrayList, **HashMaps**, **TreeMaps**, **TreeSets** and arrays.
    - I have used the HashMap to generate the mapping between key its value and the expiry time. I have taken the key of the HashMap as a string and its value as an **array** of string of size two. The first element in this array is the **value** for the key and second element of the array is the **expiry tim**e. Thus using it lets me save the value and expiry time corresponding to each key.
    - To store the **keys** having various **score values** and many **elements** corresponding to each score value, I have used a **combination of data structures**. Firstly I have used a **TreeSet to store elements** corresponding to

each score. To save the **mapping** of **score** and the Set of **elements** corresponding to it, I have used a **TreeMap**.

- TreeMaps and TreeSets store their key/values in the **sorted order** and thus they prove to be useful when carrying out operations like ZRANGE and ZRANK which require the Keys/elements to be in sorted order to generate the output in the given time complexity.
- To make every TreeMap of score(as key) and Set of elements(i.e the Treeset as value) map to the corresponding string keys, i have used a normal HashMap with key as String and corresponding TreeMap as its value.

4. **Does this implementation support multithreaded operation? If not, why can't it be?**

   No,unfortunately this implementation is not competent enough to support multithreaded operation. Since only one thread can run the application at a time and another process/thread needs to wait if the application is already running at this time by some other thread. Once the application run has been completed by the thread running it, after that only some other process or thread is able to execute it on its own terms. So, this implementation is not supported for multithreaded operations. Since there are no proper database management operations supported by this application, multithreaded operations won't maintain consistency into this system and thus multithreaded operations are not in scope of the implementation being submitted.