PROBLEM STATEMENT

Build a file-based key-value data store that supports the basic CRD (create, read, and delete) operations. This data store is meant to be used as a local storage for one single process on one laptop. The data store must be exposed as a library to clients that can instantiate a class and work with the data store.

- The data store will support the following functional requirements.
- It can be initialized using an optional file path. If one is not provided, it will reliably create itself in a reasonable location on the laptop.
- A new key-value pair can be added to the data store using the Create operation. The key is always a string - capped at 32chars. The value is always a JSON object - capped at 16KB.
- If Create is invoked for an existing key, an appropriate error must be returned.
- A Read operation on a key can be performed by providing the key, and receiving the value in response, as a JSON object.
- A Delete operation can be performed by providing the key.
- Every key supports setting a Time-To-Live property when it is created. This property is optional. If provided, it will be evaluated as an integer defining the number of seconds the key must be retained in the data store. Once the Time-To-Live for a key has expired, the key will no longer be available for Read or Delete operations.
- Appropriate error responses must always be returned to a client if it uses the data store in unexpected ways or breaches any limits.

The data store will also support the following non-functional requirements.

- The size of the file storing data must never exceed IGB.
- More than one client process cannot be allowed to use the same file as a data store at any given time.
- A client process is allowed to access the data store using multiple threads, if it desires to.
- The data store must therefore be thread-safe.
- The client will bear as little memory costs as possible to use this data store, while deriving maximum performance with respect to response times for accessing the data store.

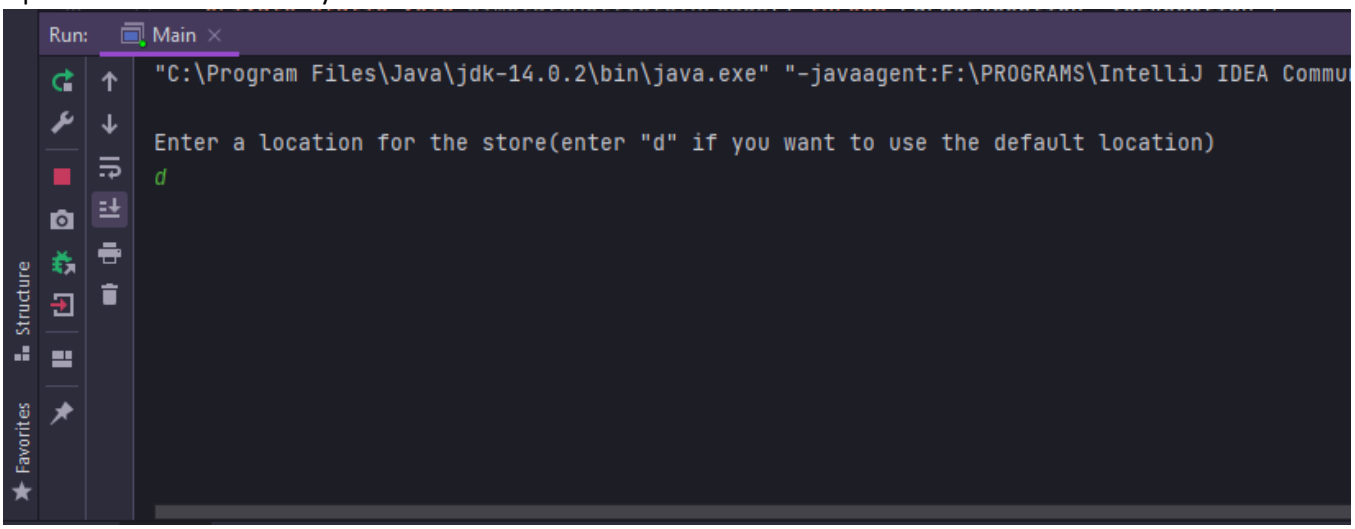
LANGUAGE

JAVA

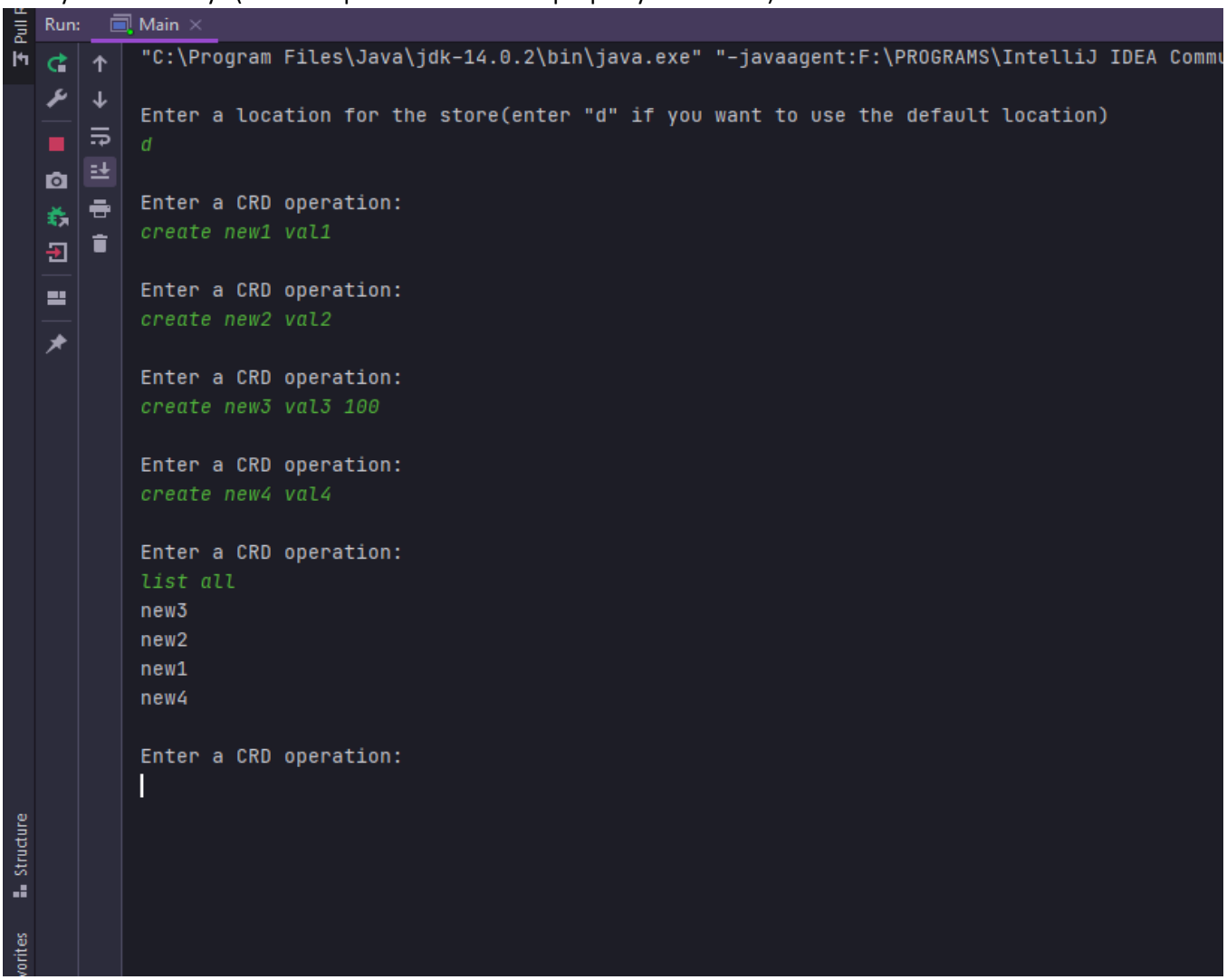
SCREENSHOTS

The program has a terminal interface as follows:

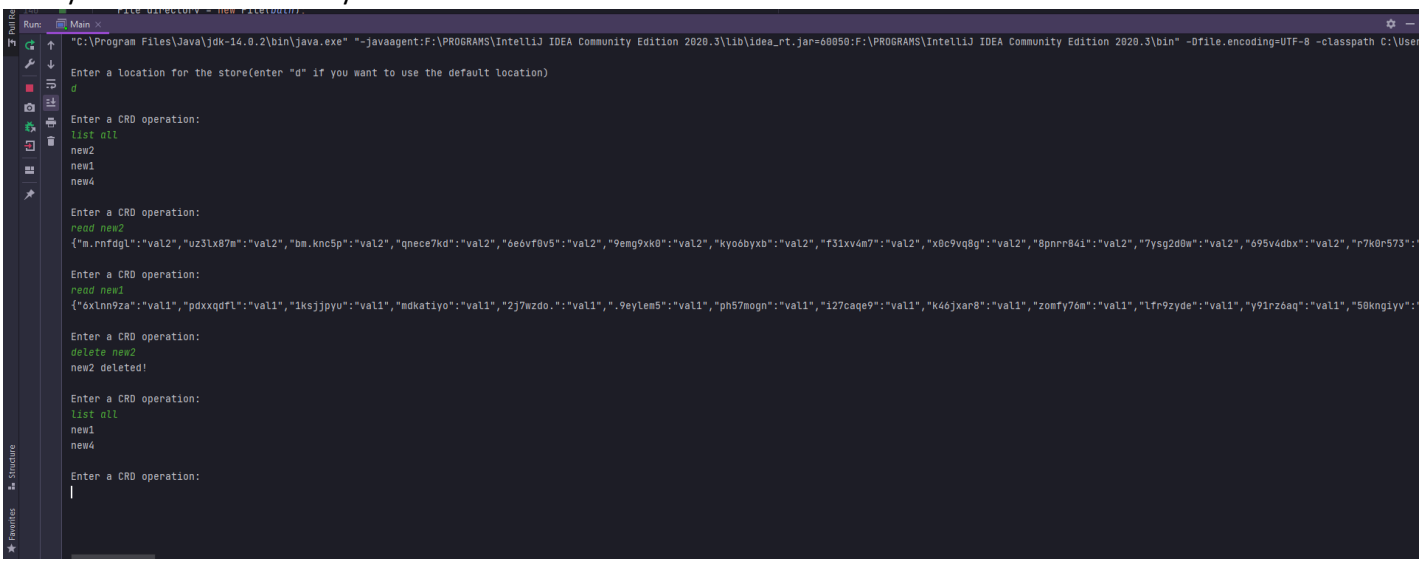
Optional file store directory



Ability to create keys (with an optional time to live property in seconds)

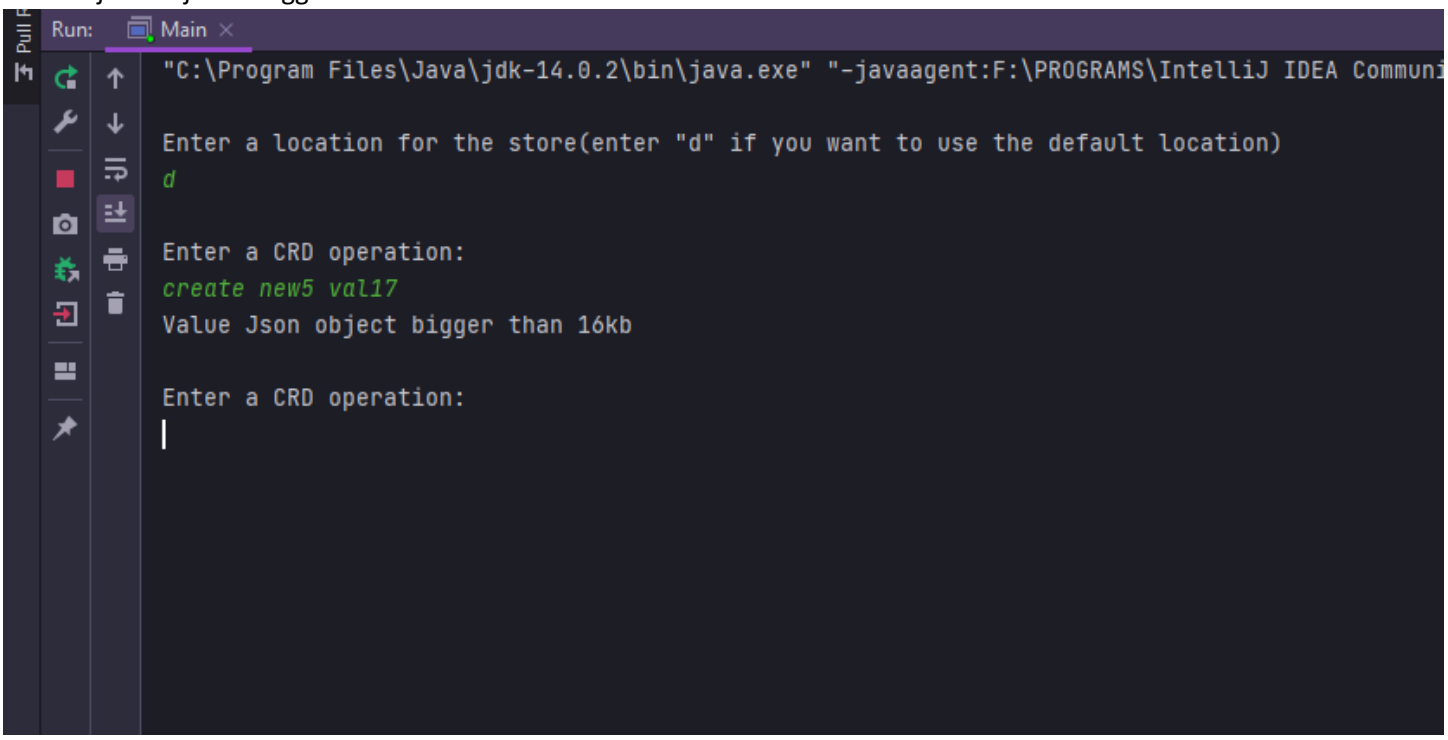


Ability to read and delete keys

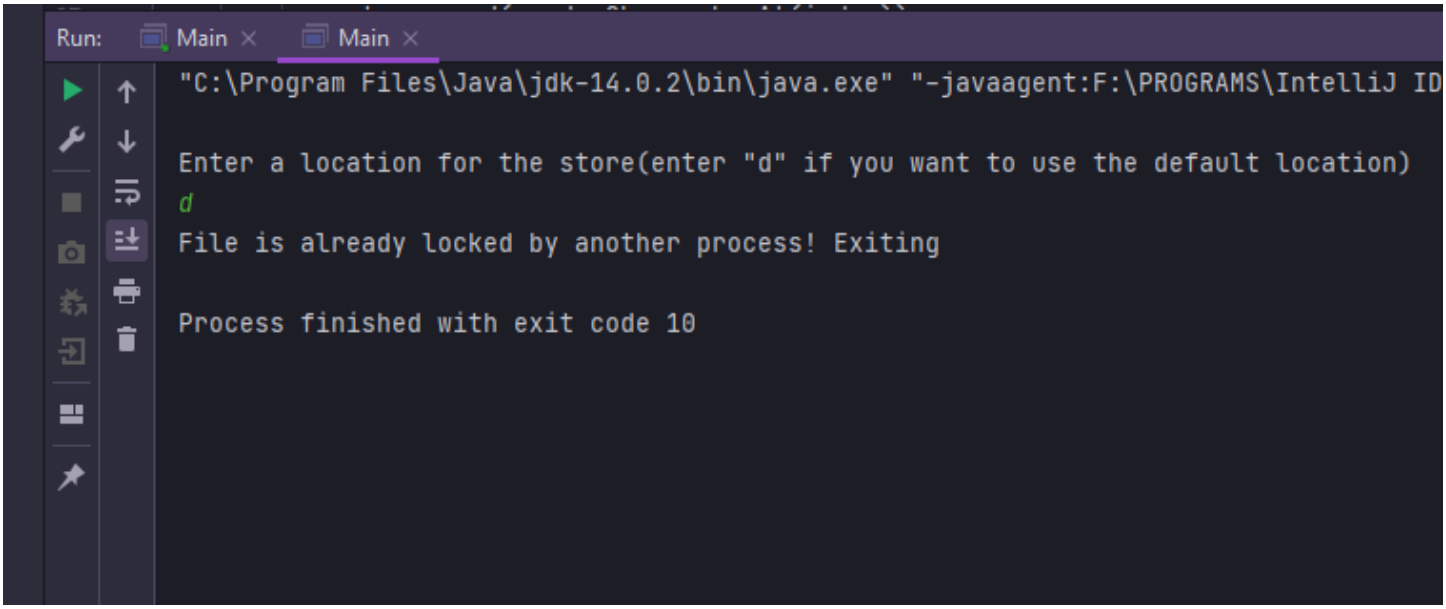


Appropriate error responses(length of string capped at 32 chars)

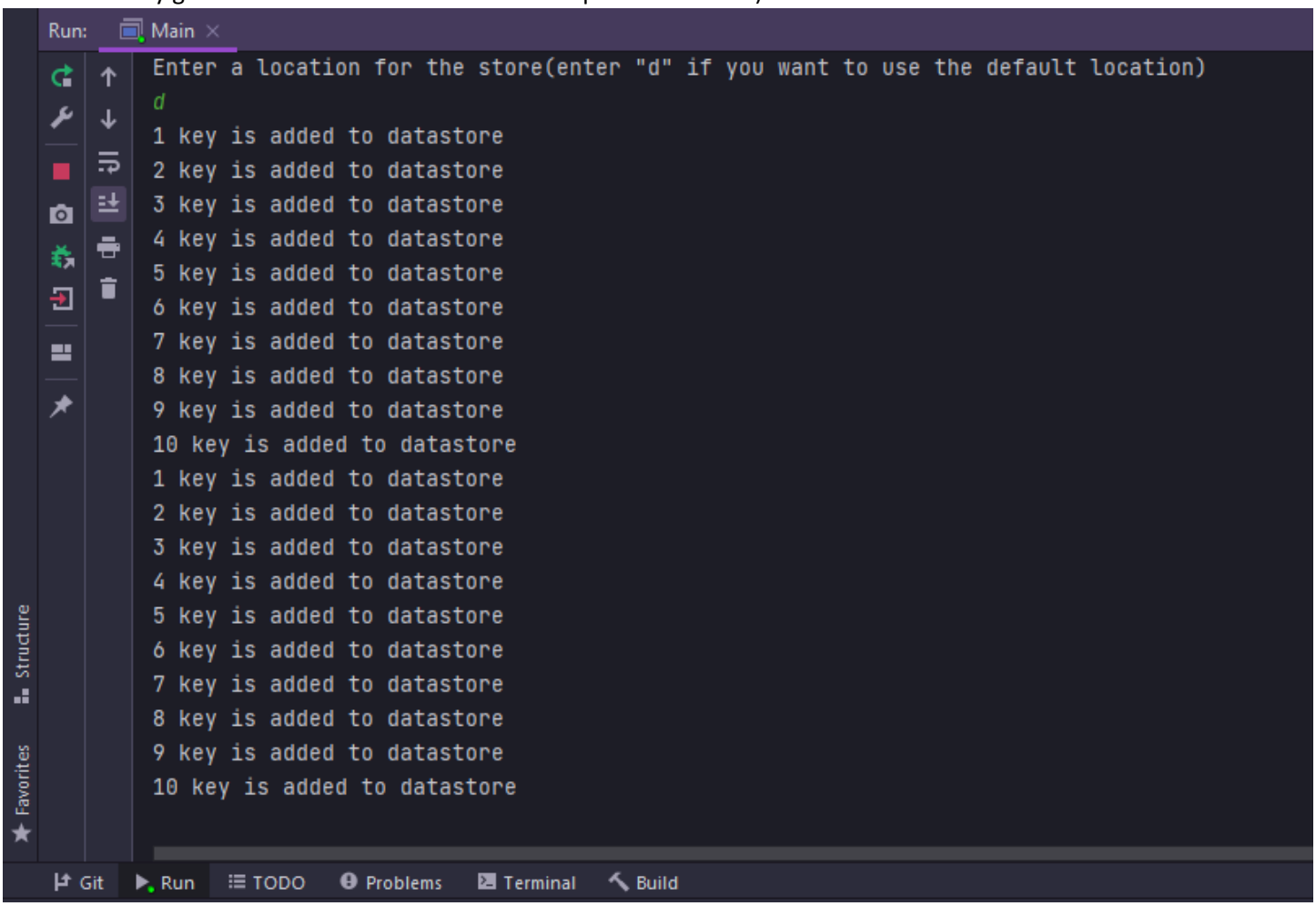
If value json object is bigger than 16kb



If other process is trying to use the file (The data store file has an exclusive lock on it)



Program can be used with multiple threads (Here two threads start at the same time which invokes a function in the datastore that makes 10 random key-value pairs. But the function is thread safe and only one thread can access the function at any given time. So the threads run in a sequential manner)



USAGE

Terminal interface with following commands:

- create key1 value1
  Where key1 and value1 are key and value pair respectively.

- create key1 value1 10
  Where 10 is the optional time to live property (Here the key will be deleted from datastore after 10 seconds)
- read key1
  Which returns the value json object (Which is printed in terminal as a string)
- delete key1
  Which deletes key1 from data store.
- list all
  Which gives the list of all keys present in data store

Program is thread safe and only one process can change or rename or delete the file at any given time (exclusive lock). To simulate json object as value given to data store, the program right now creates a random json object which is given to the function. This random json object can easily be replaced by the json object that will be stored in the data store. So, it only acts as a placeholder.