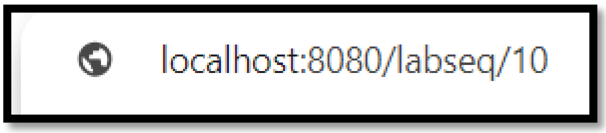
Execution instructions



localhost:8080/labseq/10

- For the user to enter the application he must access the server, in this case http://localhost:8080 (figure above)
- Must use the endpoint in the format <baseurl>/labseq/{n} where {n} represents the index of the sequence’s
- After entering the endpoint with the desired number, he must wait for the app's response which can be:
  - If everything goes well, the digit corresponds to the index entered
  - Error message if it inserts an index that is too large
  - Error message if he inserts an element other than the positive integer

Classes

MyCache

- Attributes
  - Cache<String,String> cache = Caffeine.newBuilder()
    - Is an interface provided by Caffeine that provides high-performance, near-optimal caching library for Java.It is used in the algorithm’s intermediate calculations and in the endpoint’s invocation, to store previous calculations.
  - Map<String, BigInteger> hCash = new HashMap<>();
    - Is a map used to previously store the algorithm's calculations up to 10000
    - Is used in the algorithm’s intermediate calculations and in the endpoint’s invocation, to store previous calculations.
- Methods
  - public static Map<String, BigInteger> gethCash()
    - Public method to access attribute
  - public static Cache<String, String> getCache()
    - Public method to access attribute

Utiles

- Attributes
  - private static BigInteger labSeq(String index)
    - Is the function used to calculate the algorithm
- Methods
  - public static void fillCache()
    - Is the function used to delegate the calculation and storage of the first 10000 numbers of the sequence
  - public static BigInteger getLabSeq(String index)
    - Public method to access attribute

ExampleResource

- Is the main class where http methods are invoked
- Methods
  - public CompletionStage<Response> getIndex(@PathParam("index") String index)
    - It is an asynchronous function responsible for executing the GET method and for receiving the client's request and giving feedback

Unitary tests

Function	Test Case	Input	Expected Result	Actual Result	Pass/Fail
getIndex	Invalid Input	“x”	“Invalid input”	“Invalid input”	Pass
	Invalid Input 2	“10.1”	“Invalid input”	“Invalid input”	Pass
	Invalid Input 3	“x10”	“Invalid input”	“Invalid input”	Pass
getIndex	Very Large Input	“11001”	“Very large input”	“Very large input”	Pass
getIndex	Valid Input	“0”	“0”	“0”	Pass
	Valid Input 2	“10000”	“6995...”	“6995...”	Pass