

Jevgenij Ivanov, 20748055

MH602 Computer Science & Software Engineering, CS210

The general idea of the project initially was to make a program that randomly generates “In Soviet Russia” jokes (The joke usually is the opposite of what happens in normal day-to-day life, e.g. “In Soviet Russia, Bitcoin mine you.”), and then compares 2 jokes together to get a match of hexadecimal characters. To achieve this I decided to upload two text files into my project, one for a list of nouns and one for a list of verbs, then hardcoding the first part as “In Soviet Russia,” and padding it with a random noun followed by a random verb, finally ending it with “you.”. To make my program read a text file, a `BufferedReader` class was used and the code for that looked like this:

```
BufferedReader noun = new BufferedReader(new FileReader("Nouns.txt"));  
BufferedReader verb = new BufferedReader(new FileReader("Verbs.txt"));
```

To make sure that the nouns and verbs are truly random, I had to create an array list and fill it in with all the lines from the text file, then using `.get(nounList.size())` to pull a noun/verb from a random index between 0 and the size of the list. This resulted in some very funny lines, and it was quite amusing to keep running the program to see what lines my code would generate.

After running the program numerous times, I noticed that the code is not reliable because it could generate sentences that are the same, which would result in a score of 64 when comparing the sha256 codes of the 2 sentences. This was not OK, so I decided to write a code that firstly generates all the jokes into a file using the `BufferedWriter` class, similarly to how `BufferedReader` was done:

```
BufferedWriter jokes = new BufferedWriter(new FileWriter("Jokes.txt"));
```

This way there was no randomness involved, I used a nested loop to write a Jokes text file. The outer loop took a noun, while the inner loop went through all the verbs. The code wrote over 1.7 million jokes into a text file in just a few seconds, which really impressed me.

Midway through exploring `BufferedWriter` and trying to make it take 2 sentences for comparing their sha256 codes, I realized that it would be simpler to just make an array list of strings, and fill that list with the jokes from my Jokes.txt files that my `BufferedWriter` wrote. So at this point I have 3 array lists ready to go, a list for nouns, a list for verbs and a list for the jokes. The lists for nouns and verbs were not needed any more because they were used to create a Jokes.txt file, therefore I removed them from my final code, even though they did play a big role in my code initially.

Finally, it was time to use the Jokes.txt file with over 1.7 million jokes and attempt to find 2 jokes with the highest number of matches in their sha256 codes. I have taken a code that was provided in the final lecture of this module, which accepts a string and returns a sha256 code. This method contains lines of code and keywords that I still do not fully understand, but I have a basic idea of how it works after researching some of its key components, such as `MessageDigest` object. While trying to understand the sha256 method, I have also learned that `StringBuffer` is a very useful class that is used for manipulating strings in a more flexible way, such as adding characters or concatenations. I will definitely be using it in the future.

After confirming that the sha256 method works correctly, and it returns the hash code of the jokes I pass into it, it was time to create a method that would count the matches of 2 sha256 codes. It was a simple method that takes in 2 sha256 codes, loops through the whole hash code (64 times) and counts how many characters are the same at a specific index using `charAt()` method. The whole method is shown below:

```

public static int shaScore(String a, String b) // A method that takes in 2 sha256 codes and returns the int score
{
    int result = 0; // Initial variable to keep track of the score

    for (int pos1 = 0; pos1 < a.length(); pos1++) // a for loop that will loop 64 times, to go through the whole sha256 code.
    {
        if (a.charAt(pos1) == b.charAt(pos1)) // if 2 characters are the same from code a and code b
        {
            result++; // increment the score by 1
        }
    }

    return result; // return the final score
}

```

After all the code was done and ready to go, it was time to test a couple of strings and see if it works correctly and what score would I get. At first, I decided to just simply input 2 jokes and see what is the result. The output of my program was the first sentence, the second sentence, followed by the score of matches in the hash code between the 2 sentences. After ensuring that the program works as intended, it was time to increase the number of strings that the program will check. Another nested loop was created which would cycle through the whole list multiple times, taking 1 string from the outer loop and comparing it with every other string in the array list through the inner loop. There is an if statement that keeps track of the highest score so far, and the 2 sentences that belong to the current score. The whole structure looked like this:

```

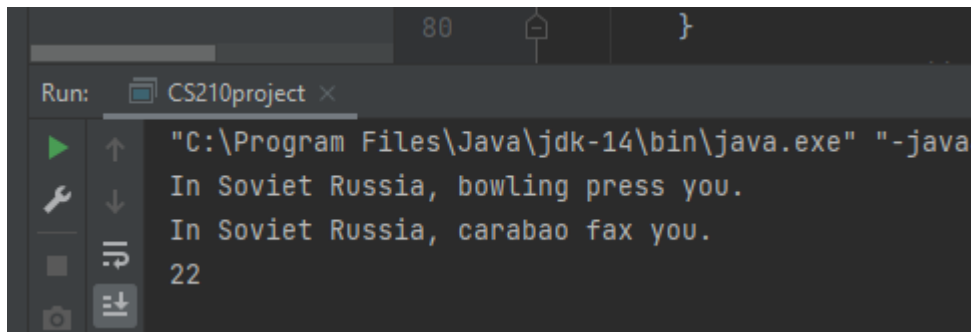
for (int pos1 = 0; pos1 < jokeList.size(); pos1++) // Outer loop for sentence 1
{
    for (int pos2 = pos1 + 1; pos2 < jokeList.size() - 1; pos2++) // Inner loop for sentence 2
    {
        sentence1 = jokeList.get(pos1); // Variable for sentence 1 from outer loop
        sentence2 = jokeList.get(pos2); // Variable for sentence 2 from inner loop

        temp = shaScore(sha256(sentence1), sha256(sentence2)); // Variable that holds the score between 2 hash codes

        if (temp > result) // If statement to keep track of the highest number of matches
        {
            result = temp; // Result is now the score between 2 hash codes
            finalSentence = sentence1 + "\n" + sentence2; // A string that will hold 2 jokes separated by a linebreak
        }
    }
}

```

I started off with just the first 10 jokes from the Jokes.txt file that was created earlier. Again, after pressing the run button in my IDE, my code reads the Jokes.txt file, adds each line into an array list using a loop, gets a temporary score using a combination of “shaScore” and “sha256” methods and checks if this is the highest score so far. The first result I got was a score of 7. For the second attempt, I have ramped up the number of line to 1000 and the result was a score of 12. Third attempt, number of lines is now 10 000, score is 13. Next is 50 000 lines, I started to notice that the program is starting to take longer and longer to output my score, which was 15 this time and took around a couple of minutes. Now it was the time to try and scan through larger number of jokes, 500 000 this time, I have realized that the time that it takes to scan through my jokes was growing exponentially, perhaps the combination of using lists first followed by a nested loop is not the most efficient way, and I was starting to worry that I would get some kind of error after many hours of waiting. It took 20 hours to scan through half a million jokes, and the result was as follows:



```
Run: CS210project x
"C:\Program Files\Java\jdk-14\bin\java.exe" "-java
In Soviet Russia, bowling press you.
In Soviet Russia, carabao fax you.
22
```

Sentence 1: In Soviet Russia, bowling press you.

Sentence 2: In Soviet Russia, carabao fax you.

The score of matches: 22.

These might not be the funniest jokes that my program has output, but I was quite happy with the score I got.

I decided to do one more final run, and go for 750 000 jokes to be scanned from the other side of the text file. Around 45 hours later, I got the score of 22 again. I was satisfied with my final result, even though I was hoping I could get a score of 25 or higher. It was time to let my computer rest after it worked tirelessly through multiple nights.

Overall it was a fun project to work on, I have learned a couple of new concepts such as `BufferedReader/BufferedWriter` and how they can be used to read and write files, which I didn't know was possible before. I've expanded my knowledge of working with lists and learned a bit about `StringBuilder` and `StringBuffer` which will definitely be useful in my future programs. In the future, if I had to do the same or similar project, I would try and make it more efficient by eliminating array lists all together to avoid the program doing double work and taking a lot of time to run through all the text lines. For a person that invested in bitcoin, it was very interesting for me to finally have a basic idea of how bitcoin mining actually works. The project ignited an interest inside me to explore cryptography and perhaps consider taking a cryptography module in the future.

Appendix with all the code:

```
import java.io.*;
import java.security.MessageDigest;
import java.util.*;

public class CS210project
{
    public static void main (String args [])
    {
        ArrayList<String> jokeList = new ArrayList(); // An array
list that will hold lines from Jokes.txt text file

        int temp = 0; // Variable that will hold the temporary score
between 2 hash codes
        int result = -1; // Variable for final score. Initial value is
-1, so it can be replaced with any positive int.

        String sentence1 = ""; // String that holds the first sentence
that will be checked
        String sentence2 = ""; // String that holds the second sentence
```

```

that will be checked
    String finalSentence = ""; // Final string that will print out
2 sentences with the highest score

    try
    {
        BufferedReader jokes = new BufferedReader(new
FileReader("Jokes.txt")); // A class that reads a text file from
computer

        jokeList.add(jokes.readLine()); // Adds the next line from
the text file into an array list of strings
        while (jokes.readLine() != null) // A loop that adds the
next line from the text file into an array list of strings
        {
            jokeList.add(jokes.readLine()); // Adds the next line
from the text file into an array list of strings
        }
    }
    catch (IOException e) // Catches an exception if there was an
error reading the file
    {
        e.printStackTrace(); // Prints the error
    }

    for (int pos1 = 0; pos1 < jokeList.size(); pos1++) // Outer
loop for sentence 1
    {
        for (int pos2 = pos1 + 1; pos2 < jokeList.size() -1;
pos2++) // Inner loop for sentence 2
        {
            sentence1 = jokeList.get(pos1); // Variable for
sentence 1 from outer loop
            sentence2 = jokeList.get(pos2); // Variable for
sentence 2 from inner loop

            temp = shaScore(sha256(sentence1), sha256(sentence2));
// Variable that holds the score between 2 hash codes

            if (temp > result) // If statement to keep track of
the highest number of matches
            {
                result = temp; // Result is now the score between
2 hash codes

                finalSentence = sentence1 + "\n" + sentence2; //
A string that will hold 2 jokes separated by a linebreak
            }
        }
    }

    System.out.println(finalSentence + "\n" + result); // Print
the 2 sentences followed by their score for hash code
}

    public static int shaScore(String a, String b) // A method that
takes in 2 sha256 codes and returns the int score
    {
        int result = 0; // Initial variable to keep track of the score

```

```

        for (int pos1 = 0; pos1 < a.length(); pos1++)    // A for loop
that will loop 64 times, to go through the whole sha256 code.
        {
            if (a.charAt(pos1) == b.charAt(pos1))    // If 2 characters
are the same from code a and code b
            {
                result++;    // Increment the score by 1
            }
        }
        return result;    // Return the final score
    }

    public static String sha256(String input)    // Method that takes in
a string and outputs a sha-256 code in a form of a string
    {
        try
        {
            MessageDigest mDigest = MessageDigest.getInstance("SHA-
256");

            byte [] salt = "CS210+".getBytes("UTF-8");
            mDigest.update(salt);

            byte [] data = mDigest.digest(input.getBytes("UTF-8"));
            StringBuffer sb = new StringBuffer();

            for (int pos1 = 0; pos1 < data.length; pos1++)
            {
                sb.append(Integer.toString((data[pos1] & 0xff) + 0x100,
16).substring(1));
            }

            return sb.toString();
        }
        catch (Exception e)
        {
            return(e.toString());
        }
    }
}

```