Task 3: Handshake Digest

Due 30 Nov 2022 by 17:00 **Points** 0 **Submitting** a website url
Available 17 Nov 2022 at 12:00 - 30 Nov 2022 at 17:00

This assignment was locked 30 Nov 2022 at 17:00.

This assignment is about hashing. We will use hashing to compute *digests*, or fingerprints, of data exchanged during handshake negotiations between client and server. Those digests will be used for integrity protection and for authentication.

The hashing algorithm we use is SHA-256. There is already a class in JCA that suits our needs very well, the MessageDigest class, so there is not much implementation work you need to do to support digest computations.

HandshakeDigest Class

Implement the HandshakeDigest class. When an instance is created by calling the constructor, which does not take any parameters, it is initialised for SHA-256 hashing. The constructor looks like this:

```
public HandshakeDigest()
```

The HandshakeDigest class has two methods:

```
public void update(byte[] input)
```

The input to a hash function is a sequence of bytes that can be of any length. The hash value is computed in an iterative way, where the update method "feeds" the hash function with more input data.

```
public byte[] digest()
```

The digest method returns the final digest, which is the hash value computed over the data given through the one or more calls to the update method.

The FileDigest program

The HandshakeDigest program is a Java program that takes a file name as parameter, computes a HandshakeDigest of the (as a SHA-256 hash) of the file's content, and prints the result to the system output. So in a terminal window, you would run the program in the following way:

```
$ java FileDigest <filename>
```

where <filename> denotes the name of a file. The program reads the input file as binary data, which means that you need to pay attention to how you deal with the input file in your program; you should the input file as binary data, that is, as a stream of bytes.

Digest Output

The FileDigest program prints the hash of the file to system output. Since the hash is a binary value, it is not well-suited to be printed directly to a terminal, and therefore the program encodes the hash as text before it is printed.

There are many ways to represent binary values as text. A straight-forward way could be to print the hash as a (very large) integer value, for instance in decimal or hexadecimal notation. But that is not very efficient. If we print an integer value as a hexadecimal string, it means that we use two characters for each byte in the integer. A 4-byte integer, for instance, would be printed as 8 characters (which corresponds to 8 bytes).

Instead, the HandshakeDigest program prints the hash as a Base64-encoded string. Base64 is an encoding scheme that is widely used in applications such as Web and email for encoding binary data. If the file is called "input.txt", the output would look something like this:

```
$ java FileDigest input.txt
spqYNYFYnslkBbtblz2xmXIweT8ZrV6imRxwx7yPm0o=
```

Assignment

Implement the HandshakeDigest class as a subclass of MessageDigest.

Implement the FileDigest program to compute the hash of a file, as described above.

Testing

In your KTH GitHub repository, you will find an example data file and a file with the corresponding SHA-256 hash. Run your program and make sure that the result is identical to the content of the file with the hash.

A reference implementation of FileDigest is available here as a jar file: **FileDigest.jar (https://canvas.kth.se/courses/36226/files/6039422?wrap=1) (https://canvas.kth.se/courses/36226/files/6039422/download?download_frd=1)** . Compare the output of your program with the output of the reference implementation. On the command line, run it like this:

```
$ java -jar FileDigest.jar <filename>
```

Submission

Submit two files: HandshakeDigest.java and FileDigest.java.

- Commit your changes in git and push to your personal repository on KTH GitHub.
    - **Note:** the file should not contain any package declarations. If your IDE inserts package declarations for you, you need to remove them.
- Submit the URL of your personal KTH GitHub repository here, in this assignment, to notify us about the submission.

Important: We will not automatically check your personal KTH GitHub repository for updates. Whenever you make an update that you want us know about, you have to make a submission here. Submit the URL of your personal KTH GitHub repository.