LA6 (Pair project)
Encoder and Decoder

Due Date (a one-week LA)
11/14/19@ 11:59pm

Concepts

- Binary Files
- Data Structures
- Pair Programming

Problem Specification

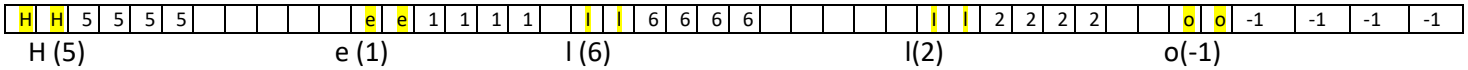
Communication privacy is very important nowadays. So, in many situations, messages are usually encoded before being sent out to the recipients.

You are to write a Java program to encode and decode a text file. The **encoder** reads a message stored in a plain text file, encodes it, and stores it in a binary file. The **decoder** reads the binary file, decodes the message and prints it to the console.

The encoding algorithm works as follows:

- Each character **c** in the message is followed by a **randomly-generated** number **n** ranging from 1 to 20. **n** is the number of bytes of random data between **c** and the next character in the message. So, after writing **c** followed by **n** to the file, there should be **n** byte locations (with random data) before the next character **c** is written.
- The last character is followed by the number -1 which indicates the end of the message.
- Each character stored in the binary file occupies 2 bytes while each integer occupies 4 bytes. Random data is stored between the integer following each character and the next character.

The following figure shows the structure of the binary file with the message "**Hello**" stored in it. Each cell represents one byte. The empty cells represent random data stored between the integer following each character and the next character. Note that each character in the figure occupies two byte locations while each integer occupies 4 byte locations.



Design Phase

You are required to write an **Encoder** class that implements the following interface.

```
public interface IEncoder {
        // Given the paths of an input file and an output file, the encoder
        // will read the message from the input file, encode the message
        // as described above, and store it into a binary file with the
        // given path.
        // Include code to handle the IOException.
        public void encode(String inputFileName, String outputFilePath);
}
```

You are also required to have a **Decoder** class that implements the following interface.

```
public interface IDecoder {
        // Given the file path of the binary file, the decoder will read the file, decode the message and
        // print it to the console.
        // Include code to handle the IOException.
        public void decode(String filePath);
}
```

Pseudocode

Write pseudocode for the required methods (as specified above) and for any other methods you may want to use in your code.

*Note: **Correct pseudocode development** will be worth **40%** of the total LA grade.*

You will also need to **generate Javadoc** for your project. Follow the steps shown in class (a copy of these steps can be found on the Content page in Elearning, or can otherwise be provided to you by your Lab instructor). If you follow the steps accurately, a "**doc**" folder (for Javadoc) should be created in your project.

Implementation Phase

Using the pseudocode developed, write the Java code for your assignment. This is a one-week assignment.

Testing Phase

To test your program, you can use the file input.txt provided in the dropbox or make up your own text file.

NOTE: You are required to use the Main class provided below to run your program

```
public class Main {
      public static void main(String[] args) {
            IEncoder encoder = new Encoder();
            IDecoder decoder = new Decoder();
```

```
            String inputFileName = "input.txt";
            String encodedFileName = inputFileName+".encode";
            encoder.encode(inputFileName,encodedFileName);
            decoder.decode(encodedFileName);

        }
}
```

Example Input (input.txt):
In this tutorial we are going to see how to use RandomAccessFile in order to to
read and write data to a File (in random positions). The RandomAccessFile class
treats the file as an array of Bytes. You can write your data in any position of
the "array". To do that, it uses a pointer that holds the current position
(you can think of the file pointer as a cursor in a text editor).

The RandomAccessFile does that using:

getFilePointer() to get the current position of the pointer
seek(int) to set the position of the pointer
read(byte[] b) to reads up to b.length bytes of data from the file into an array of bytes
write(byte[] b) to write b.length bytes from the specified byte array to the file, starting at the current file pointer position

Example Output:
In this tutorial we are going to see how to use RandomAccessFile in order to to
read and write data to a File (in random positions). The RandomAccessFile class
treats the file as an array of Bytes. You can write your data in any position of
the "array". To do that, it uses a pointer that holds the current position
(you can think of the file pointer as a cursor in a text editor).

The RandomAccessFile does that using:

getFilePointer() to get the current position of the pointer
seek(int) to set the position of the pointer
read(byte[] b) to reads up to b.length bytes of data from the file into an array of bytes
write(byte[] b) to write b.length bytes from the specified byte array to the file, starting at the current file pointer position

Note that both the input and the output are the same, in line with the requirements.

Additional Requirements

Pair Programming

You should each have either a GItLab account, and should also have a partner with whom you will be working on this assignment.

There should be visible evidence of both partners in a team contributing to the work required for this assignment (each person's commits should reflect in the team's repository). You will be graded partially on how well you use your GIT repository. The "commits" on GitHub or GitLab will be reviewed to determine if there was input from both team members. There should be at least two commits from each person on the team. In addition, in the pseudocode document, a **brief** explanation should be included explaining what each team member worked on.

You must add your lab instructor (with Maintainer access) to your GIT repository to provide access for grading purposes. For additional help outside of the explanations provided by your lab instructors, you may consult the "Git Guide" document which your lab instructors will make available to you. It includes steps specifying how to perform various tasks on Git.

If you experience any problems with Git, get in touch with your lab instructor at the earliest opportunity. A zip file containing your submission needs to be submitted to Elearning only if you can't get Git working (you will be penalized for this, so you need to make every effort to work with Git). If your code is in the repository, you will not have to upload it to Elearning. However, you still need to keep to the due date for submission – note that the most recent date/time of modification can be seen in Git.

Coding Standards

You must adhere to all conventions in the CS 1120 Java coding standard (available on Elearning for your Lab). This includes the use of white spaces and indentations for readability, and the use of comments to explain the meaning of various methods and attributes. Be sure to follow the conventions also for naming classes, variables, method parameters and methods.

Assignment Submission

- The final version of your project should be uploaded to the GitLab or GitHub repository (a Git push) by the due date / time.

- The pseudocode should be submitted to the appropriate folder on Elearning. Only one pseudocode document is required for each team / pair and the names of the team members should be clearly indicated on the report.

NOTE: The eLearning folder for LA submission will remain open beyond the due date but will indicate how many days late an assignment was submitted where applicable. The dropbox will be inaccessible seven days after the due date by which time no more credit can be received for the assignment.

The penalty for late submissions as stated in the course syllabus will be applied in grading any assignment submitted late.