# Practice problem sets

1. **File Concatenation** :
   You are tasked with writing a Java program to concatenate the contents of two input text files and save the result into an output file. The program should take the names of the two input files and the output file from the user. It should then read the contents of the two input files and concatenate them line by line into the output file.

   Your program should adhere to the following requirements:

   - Prompt the user to enter the names of the two input text files and the output text file.
   - Open the input files and read their contents line by line.
   - Concatenate the contents of the two input files, placing the contents of the second file below the contents of the first file.
   - Write the concatenated content into the output file, ensuring each line from the input files is preserved in the output file.
   - Close all file streams after reading and writing data.

   Note: Make sure you have two input text files (input1.txt and input2.txt) with some content, and the output file (output.txt) will contain the concatenated content of the two input files.

2. **File Copy with Progress Display**
   You have been tasked to write a Java program that copies the contents of one binary file to another binary file while displaying the progress of the file copy in terms of the percentage completed.

   The program should have the following specifications:

   - Declare two string variables source and destination to hold the names of the source and destination binary files, respectively. Initialize them with the file names for example : "src.mp4" and "copy.mp4", respectively.

   - Create an InputStream object and an OutputStream object to read from the source file and write to the destination file, respectively.

   - Display the size of the source file (you can use the available() method of the InputStream object).

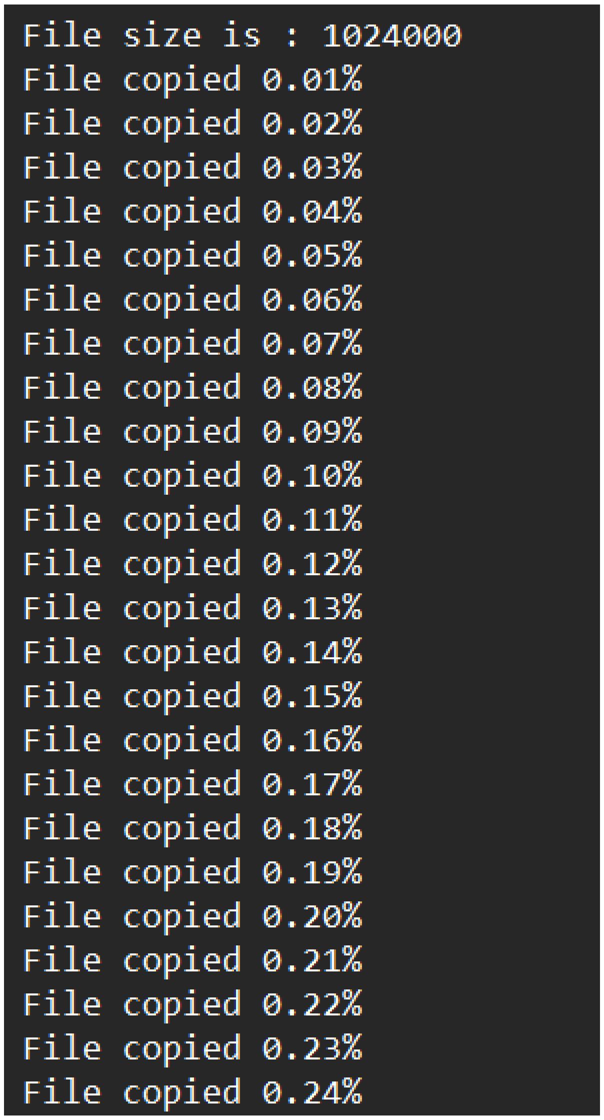
- Use a while loop to read bytes from the source file using the read() method and write them to the destination file using the write() method. Track the number of bytes copied so far in an integer variable named copiedBytes.

- Inside the loop, calculate the percentage of the file copied, and display the progress in the format "File copied XX.XX%".

- Ensure that the InputStream and OutputStream are properly closed after copying is completed.

Your task is to implement the program following the specifications above.

Sample output for the progress bar :



```
File size is : 1024000
File copied 0.01%
File copied 0.02%
File copied 0.03%
File copied 0.04%
File copied 0.05%
File copied 0.06%
File copied 0.07%
File copied 0.08%
File copied 0.09%
File copied 0.10%
File copied 0.11%
File copied 0.12%
File copied 0.13%
File copied 0.14%
File copied 0.15%
File copied 0.16%
File copied 0.17%
File copied 0.18%
File copied 0.19%
File copied 0.20%
File copied 0.21%
File copied 0.22%
File copied 0.23%
File copied 0.24%
```

Test the program with a binary file of your choice and observe the progress messages indicating the percentage of the file copied at each iteration.

Note: The program should work with any binary file and not just limited to "src.mp4" and "copy.mp4". Make sure to handle any potential file I/O exceptions gracefully.

3. **Character Frequency Counter Program**
   You are tasked with writing a Java program that reads a text file and calculates the frequency of each character present in the file. The program should then print the characters along with their respective counts.

   Your program should adhere to the following requirements:

   - Prompt the user to enter the name of the text file from which to read and count the characters.

   - Open the specified text file and read its contents character by character.

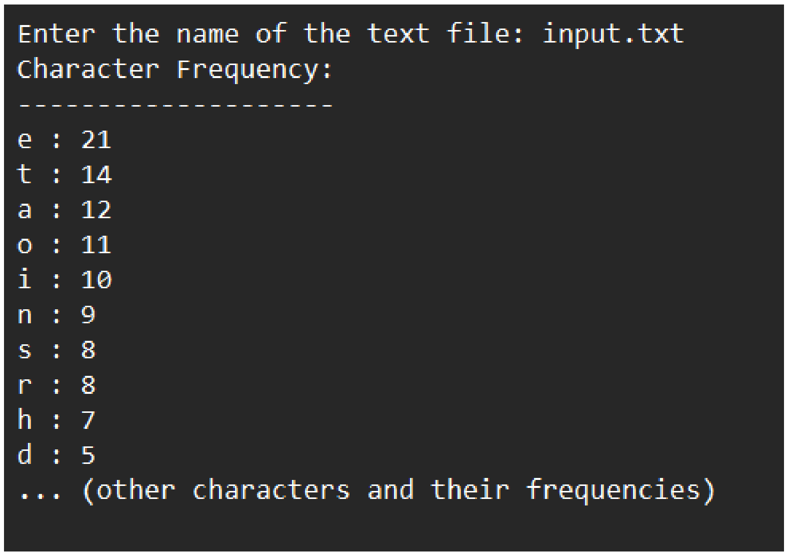
   - Use a data structure (for example : array) to keep track of the character frequencies.

   - For each character read from the file, update its frequency count.

   - After reading the entire file, print each character along with its frequency count.

   - Ensure you handle any potential file I/O exceptions gracefully.

Sample output :



```
Enter the name of the text file: input.txt
Character Frequency:
--------------------
e : 21
t : 14
a : 12
o : 11
i : 10
n : 9
s : 8
r : 8
h : 7
d : 5
... (other characters and their frequencies)
```

Your task is to implement the program following the specifications above.

Test the program with different text files to verify its accuracy in counting the character frequencies.

Note: Make sure to provide appropriate error messages and handle cases where the specified file is not found or is empty.