Homework #4

CS 4120/5220 – Adv Appl Development in Java

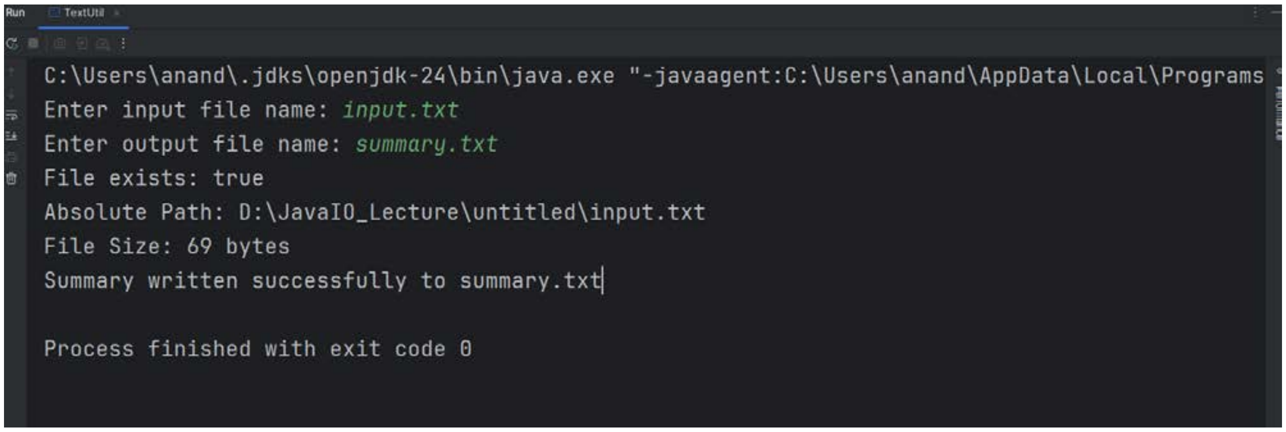
Fall 2025

100 Points
Due: 10/31/2025, 11:59 PM

Instructions: Submit all required project files, including source code, as a single compressed archive (ZIP file). Ensure that your project is well-organized, with appropriate folder structures for source files, resources, and any required dependencies. Include a README file if necessary to explain how to run the program. Do not copy and paste code from external sources. Clearly comment on your code to explain important logic. Convert any written explanations, analysis, or required screenshots into a single PDF file and include it in the ZIP archive. Upload the ZIP file to Brightspace before the due date and time.

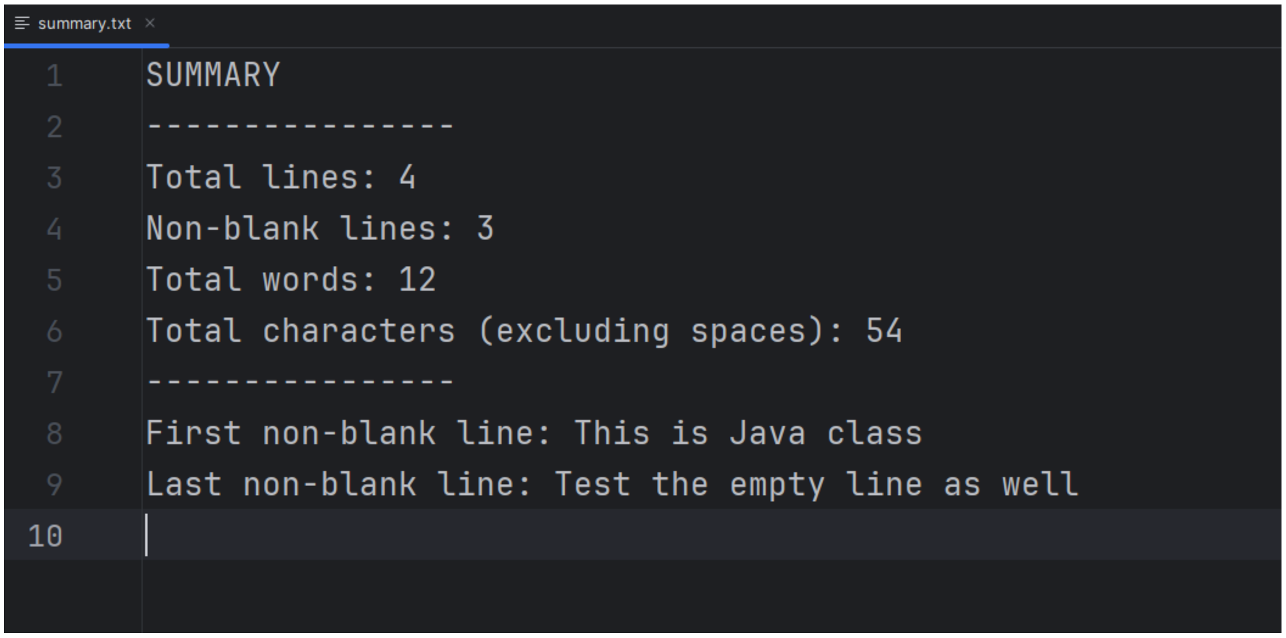
1. Write a Java program that analyzes a given text file and creates a formatted summary report.
    a. Prompt the user to enter an input file name and an output file name.
    b. Use the File class to:
        i. Check if the input file exists and is readable.
        ii. Display the file's absolute path and size in bytes.
    c. Use BufferedReader to read the file **line by line** until end-of-file.
    d. For each line:
        i. Count total characters (excluding spaces).
        ii. Count total words (split by whitespace).
        iii. Count the number of non-blank lines.
    e. Write the summary to the **output file** using BufferedWriter, including:
        i. Total lines, total non-blank lines, total words, and total characters.
        ii. A separator line (e.g., "----------------").
        iii. The first and last non-blank lines from the input file.
    f. Ensure you call flush() before closing the writer.
    g. Handle all exceptions gracefully using try-with-resources.

Sample execution output:



```
C:\Users\anand\.jdks\openjdk-24\bin\java.exe "-javaagent:C:\Users\anand\AppData\Local\Programs
Enter input file name: input.txt
Enter output file name: summary.txt
File exists: true
Absolute Path: D:\JavaIO_Lecture\untitled\input.txt
File Size: 69 bytes
Summary written successfully to summary.txt

Process finished with exit code 0
```



```
SUMMARY
----------------
Total lines: 4
Non-blank lines: 3
Total words: 12
Total characters (excluding spaces): 54
----------------
First non-blank line: This is Java class
Last non-blank line: Test the empty line as well

```

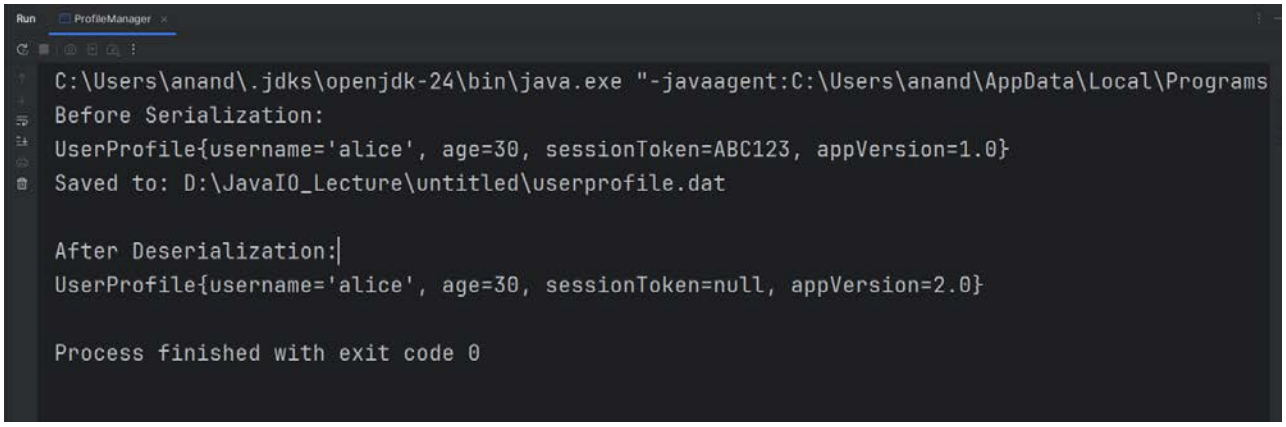
2. Write a Java program to demonstrate saving and restoring object data using **serialization**: **(50 points)**
    a. Create a class UserProfile that implements Serializable and contains:
        i. String username
        ii. int age
        iii. transient String sessionToken (should not be serialized)
        iv. static String appVersion = "1.0" (shared among all objects)
    b. Create a driver class ProfileManager with the following operations:
        i. Create a new UserProfile object with hardcoded values.

ii. Save the object into a file (userprofile.dat) using ObjectOutputStream.
iii. Before deserialization, change UserProfile.appVersion to "2.0".
iv. Read the object back using ObjectInputStream.
v. Display all fields before and after serialization.

Expected Behavior:

i. The transient field (sessionToken) is not restored after deserialization.
ii. The static field (appVersion) reflects the current class value, not the stored value.
iii. The other fields (username, age) should retain their values.

Sample execution output:



```
C:\Users\anand\.jdks\openjdk-24\bin\java.exe "-javaagent:C:\Users\anand\AppData\Local\Programs
Before Serialization:
UserProfile{username='alice', age=30, sessionToken=ABC123, appVersion=1.0}
Saved to: D:\JavaIO_Lecture\untitled\userprofile.dat

After Deserialization:
UserProfile{username='alice', age=30, sessionToken=null, appVersion=2.0}

Process finished with exit code 0
```

Additional Requirements:

i. Use ObjectOutputStream and ObjectInputStream.
ii. Include comments explaining why transient and static behave differently.
iii. Handle all file and I/O exceptions properly.