Guided Lab - 303.13.2 - New I/O

Introduction:

Java has provided a second **I/O system** called ***New I/O*** **(NIO)**:

- It supports a buffer-oriented, channel-based approach for I/O operations.
- NIO was developed to allow Java programmers to implement high-speed I/O without using the custom native code.

Objective

In this lab, we will demonstrate how to use Java NIO.

Learning Objective

Bythe end of this lab learner will be able to use NIO

Begin

Example One: Multiple source channels (input files) to a single output channel.

In this example, we will read the file content from two different files and write their combined output into a single separate file.

- In total, we will create three channels:
    - We will create two channels for the **source file**.
    - We will create one channel for the **destination file**.

Instructions:

Click on the links below to download a dummy file for this lab:

- ➔ File1.txt
- ➔ File2.txt

Create a class named **NioExample** and write the code below.

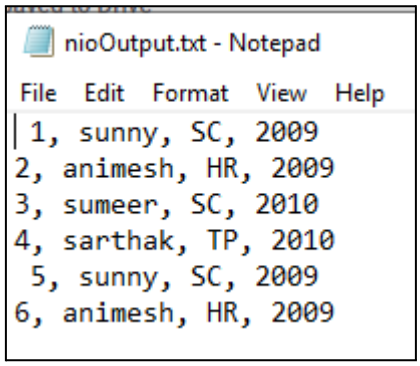
💡 ***Note: Do not forget to change the path or location for both files, file1.txt, and file2.txt.***

```
import java.io.*;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
public class NioExample {
   public static void main(String[] args) throws IOException
   {
    // initializing two files in Array;
 String[] inputFiles = {"C:/Downloads/file1.txt",
               "C:/Downloads/file2.txt"};

  // Specify out file with path location
//Files contents will be written in these files
String outputFile = "C:/Downloads/nioOutput.txt";

       // Get Channel for destination or outputFile
       FileOutputStream  fos = new FileOutputStream(outputFile);
       FileChannel targetChannel =  fos.getChannel();

       for(int i =0; i < inputFiles.length; i++)
       {
           // Get channel for inputFiles
          FileInputStream fis = new FileInputStream(inputFiles[i]);
          FileChannel inputchannel =  fis.getChannel();
          long size = inputchannel.size();
          ByteBuffer buf = ByteBuffer.allocate((int)size);
          System.out.print((char) buf.get());
          while (inputchannel.read(buf)> 0) {
             buf.flip();
             while (buf.hasRemaining()) {
             //  System.out.print((char) buf.get());
                 targetChannel.write(buf);
               }
           }
```

```
            //fis.close();
        }
    }}
```

Run your program

After executing your program**, a *nioOutput.txt*** file will be created in your computer per the specified path location.



Example Two: Java NIO FileChannel transferTo() and transferFrom().

As in normal Java applications, where IO happens mostly between an input source and an output target, it happens in NIO as well, and we may need to transfer data from ***one channel to another channel*** frequently.

Bulk transfers of file data from one place to another are so common that a couple of optimization methods have been added to the FileChannel class to make it even more efficient.

Data Transfer between Channels

Java NIO provides two methods for transferring the data between two channels:

- FileChannel.transferTo()
- FileChannel.transferFrom()

The **transferTo()** and **transferFrom()** methods allow us to cross-connect one channel to another. This eliminates the need to pass the data through an intermediate buffer.

These methods exist only in the ***FileChannel*** class; therefore, one of the channels involved in a **channel-to-channel** transfer must be a FileChannel.

Create a class named ***NioExampleTwo*** and write the code below.

💡 ***Note: Do not forget to change the path location for both files, file1.txt, and file2.txt.***

```
import java.io.*;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.channels.WritableByteChannel;
public class NioExampleTwo{
       public static void main(String[] args) throws IOException {
           // TODO Auto-generated method stub

           //Input files
String[] inputFiles = {"C:/Downloads/file1.txt", "C:/Downloads/file2.txt"};

           //Files contents will be written in these files
           String outputFile = "C:/Downloads/OutputExampleTwo.txt";
           //Get channel for output file

           FileOutputStream fos = new FileOutputStream(outputFile);
           WritableByteChannel targetChannel = fos.getChannel();
           for (int i = 0; i < inputFiles.length; i++)
           {
               //Get channel for input files
               FileInputStream fis = new FileInputStream(inputFiles[i]);
               FileChannel inputChannel = fis.getChannel();

               //Transfer data from input channel to output channel
               inputChannel.transferTo(0, inputChannel.size(), targetChannel);

               //close the input channel
               inputChannel.close();
               fis.close();
           }

           //finally close the target channel
           targetChannel.close();
           fos.close();
       }
   }
```

Run your program

After executing your program, the ***OutputExampleTwo.txt*** file will be created in your computer per the specified path location.

Submission Instructions:

Include the following deliverables in your submission -

- Submit your source code using the Start Assignment button in the top-right corner of the assignment page in Canvas.

CANVAS STAFF USE ONLY: Canvas Submission Guideline:

Instructions for Canvas Assignment Creation
Assignment Name: GLAB - 303.13.2 - NIO **Points:** 100 **Assignment Group:** Module 303: Java SE Review (Not Graded) **Display Grade As:** Complete/Incomplete **Do not count this assignment towards the final grade:** Checked **Submission Types:** Files Uploads **Everything else is the default.**