

Name: ZISHNENDU SARKER

Roll: 2K19/CO/450

Java Programming

Lab assignment 10

28/11/2021

Java Program to Append Text to an Existing File

Theory: In Java, we can append a string in an existing file using FileWriter which has an option to open the file in append mode. Java FileWriter class is used to write character-oriented data to a file. It is the character-oriented class that is used for file handling in Java.

Code:

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;

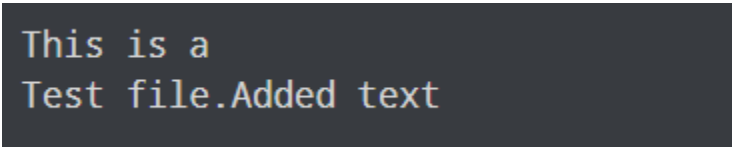
public class AppendFile {

    public static void main(String[] args) {

        String path = System.getProperty("user.dir") + "\\C:\\Users\\sristi\\Downloads\\java.txt";
        String text = "Added text";

        try {
            Files.write(Paths.get(path), text.getBytes(), StandardOpenOption.APPEND);
        } catch (IOException e) {
        }
    }
}
```

Output:



```
This is a
Test file.Added text
```

Learning Outcome:

In the above program, we use System's user.dir property to get the current directory stored in the variable path. Check Java Program to get the current directory for more information. Likewise, the text to be added is stored in the variable text. Then, inside a try-catch block we use Files' write() method to append text to the existing file. The write() method takes the path of the given file, the text to be written, and how the file should be open for writing. In our case, we used APPEND option for writing. Since the write() method may return an IOException, we use a try-catch block to catch the exception properly.

Java Program to Get the name of the file from the absolute path

Theory: The method java.io.File.getAbsolutePath () is used to obtain the absolute path name of a file or directory in the form of a string. This method requires no parameters.

Code:

```
import java.io.File;

class Main {

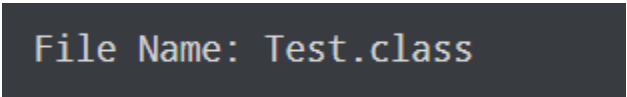
    public static void main(String[] args) {

        // link to file Test.class
        File file = new File("C:\\Users\\sristi\\Downloads\\java.txt\\Test.class");

        // get file name using getName()
        String fileName = file.getName();
        System.out.println("File Name: " + fileName);

    }
}
```

Output:



```
File Name: Test.class
```

Learning Outcome:

The absolute pathname of the file is obtained using the method java.io.File.getAbsolutePath() in the form of a string and is printed.

Java Program to Count Number of lines present in the file

Theory: At last, the Java NIO Files.lines is simple to use, and the performance isn't that much different, the best choice is to count the number of lines in a file. On Linux, the command `wc -l` is the fastest way to count the number of lines in a file.

Code:

```
import java.io.File;
import java.util.Scanner;

class Main {
    public static void main(String[] args) {

        int count = 0;

        try {
            // create a new file object
            File file = new File("input.txt");

            // create an object of Scanner
            // associated with the file
            Scanner sc = new Scanner(file);

            // read each line and
            // count number of lines
            while(sc.hasNextLine()) {
                sc.nextLine();
                count++;
            }
            System.out.println("Total Number of Lines: " + count);

            // close scanner
            sc.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
First Line  
Second Line  
Third Line
```

Learning Outcome:

In the above example, we have used the `nextLine()` method of the `Scanner` class to access each line of the file. Here, depending on the number of lines the file `input.txt` file contains, the program shows the output.