Java BufferedWriter Class

Java BufferedWriter class is used to provide buffering for Writer instances. It makes the performance fast. It inherits [Writer](https://www.javatpoint.com/java-writer-class) class. The buffering characters are used for providing the efficient writing of single [arrays](https://www.javatpoint.com/array-in-java), characters, and [strings](https://www.javatpoint.com/java-string).

1. **public** **class** BufferedWriter **extends** Writer

|  |  |
| --- | --- |
| void newLine() | It is used to add a new line by writing a line separator. |
| void write(int c) | It is used to write a single character. |
| void close() | It is used to closes the input stream |

File myObj = **new** File(System.*getProperty*("user.dir")+"\\reports\\FinalProductResult.txt");

FileWriter fw=**new** FileWriter(myObj) to write in the file

BufferedWriter wr = **new** BufferedWriter(fw);

String finalResult=mystore.getFinalProductName().getText()+System.*lineSeparator*()+mystore.getProductOfColour().getText()+System.*lineSeparator*()+

mystore.getQuantityLabel().getText()+" "+mystore.getQuantityValue().getText()+System.*lineSeparator*()+mystore.getTotalLabel().getText()+" "+mystore.getTotalValue().getText();

//System.out.println(finalResult);

wr.write(finalResult);

wr.close();

READ text from textFile:

1. File file=**new** File("Demo.txt");    //creates a new file instance
2. FileReader fr=**new** FileReader(file);   //reads the file
3. BufferedReader br=**new** BufferedReader(fr);

String line=br.readLine(); // help to read the line

While(line!=”null”)

{

Syso(line)

line=br.readLine();

}

* How to identify in JAVA file and Folder:

package com.journaldev.files;

import java.io.File;

public class CheckDirectoryOrFile {

public static void main(String[] args) {

File file = new File("/Users/pankaj/source.txt");

File dir = new File("/Users/pankaj");

File notExists = new File("/Users/pankaj/notafile");

System.out.println("/Users/pankaj/source.txt is file?"+file.isFile());

System.out.println("/Users/pankaj/source.txt is directory?"+file.isDirectory());

System.out.println("/Users/pankaj is file?"+dir.isFile());

System.out.println("/Users/pankaj is directory?"+dir.isDirectory());

System.out.println("/Users/pankaj/notafile is file?"+notExists.isFile());

System.out.println("/Users/pankaj/notafile is directory?"+notExists.isDirectory());

}

}

Output of the above program is:

/Users/pankaj/source.txt is file?true

/Users/pankaj/source.txt is directory?false

/Users/pankaj is file?false

/Users/pankaj is directory?true

/Users/pankaj/notafile is file?false

/Users/pankaj/notafile is directory?false

* To get all the file present in directory and read all the content present in file:

The class named **File** of the java.io package represents a file or directory (pathnames) in the system. This class provides various methods to perform various operations on files/directories.

To get the list of all the existing files in a directory this class provides five different methods to get the details of all files in a particular folder −

* String[] list()
* File[] listFiles()
* String[] list(FilenameFilter filter)
* File[] listFiles(FilenameFilter filter)
* File[] listFiles(FileFilter filter)

#### The ListFiles() method

This method returns an array holding the objects (abstract paths) of all the files (and directories) in the path represented by the current (File) object.

The following Java program prints the name, path and, size of all the files in the path D:\\ExampleDirectory.

mport java.io.File;

import java.io.IOException;

import java.util.Scanner;

public class ListOfFiles {

   public static void main(String args[]) throws IOException {

      //Creating a File object for directory

      File directoryPath = new File("D:\\Demo");

      //List of all files and directories

      File filesList[] = directoryPath.listFiles();

      System.out.println("List of files and directories in the specified directory:");

      Scanner sc = null;

      for(File file : filesList) {

         System.out.println("File name: "+file.getName());

         System.out.println("File path: "+file.getAbsolutePath());

         System.out.println("Size :"+file.getTotalSpace());

         //Instantiating the Scanner class

         sc= new Scanner(file);

         String input;

         StringBuffer sb = new StringBuffer();

         while (sc.hasNextLine()) {

            input = sc.nextLine();

            sb.append(input+" ");

         }

         System.out.println("Contents of the file: "+sb.toString());

         System.out.println(" ");

      }

   }

}

#### Output

List of files and directories in the specified directory:

File name: samplefile1.txt

File path: D:\Demo\samplefile1.txt

Size :262538260480

Contents of the file: Contents of the sample file 1

File name: samplefile2.txt

File path: D:\Demo\samplefile2.txt

Size :262538260480

Contents of the file: Contents of the sample file 2

File name: samplefile3.txt

File path: D:\Demo\samplefile3.txt

Size :262538260480

Contents of the file: Contents of the sample file 3

File name: samplefile4.txt

File path: D:\Demo\samplefile4.txt

Size :262538260480

Contents of the file: Contents of the sample file 4

File name: samplefile5.txt

File path: D:\Demo\samplefile5.txt

Size :262538260480

Contents of the file: Contents of the sample file 5

TESTNG :

<http://makeseleniumeasy.com/2018/06/10/testng-tutorials-22-test-annotation-games-of-priority-of-methods-in-testng/>

Map:

Put method:

### **No Duplicate Key on HashMap**

You cannot store duplicate keys in HashMap. However, if you try to store duplicate key with another value, it will replace the value.

1. **import** java.util.\*;
2. **public** **class** HashMapExample2{
3. **public** **static** **void** main(String args[]){
4. HashMap<Integer,String> map=**new** HashMap<Integer,String>();//Creating HashMap
5. map.put(1,"Mango");  //Put elements in Map
6. map.put(2,"Apple");
7. map.put(3,"Banana");
8. map.put(1,"Grapes"); //trying duplicate key
10. System.out.println("Iterating Hashmap...");
11. **for**(Map.Entry m : map.entrySet()){
12. System.out.println(m.getKey()+" "+m.getValue());
13. }
14. }
15. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=HashMapExample2)

Iterating Hashmap...

1 Grapes

2 Apple

3 Banana

http://makeseleniumeasy.com/2018/06/10/testng-tutorials-22-test-annotation-games-of-priority-of-methods-in-testng/

1. Yes, you can set null as key in Java HashMap.

Map<String,String>map = new HashMap<>();

      map.put("Football", "A");

      map.put("Squash", "B");

      map.put("Cricket", "C");

      map.put("Hockey", "D");

      map.put("Rugby", "E");

      map.put("Golf", "F");

      map.put("Archery", "G");

      System.out.println("Size of HashMap = " + map.size());

      map.put(null, "H");

      System.out.println("Updated Size of HashMap = " + map.size());

      System.out.println("For null = " + map.get(null));

Output:

Size of HashMap = 7

Updated Size of HashMap = 8

For null = H