Commencé le	jeudi 19 octobre 2017, 09:52
État	Terminé
Terminé le	jeudi 19 octobre 2017, 11:02
Temps mis	1 heure 9 min
En retard	1 heure 9 min
Points	5,00/5,00
Note	**20,00** sur 20,00 (**100**%)

Description

These exercises are based on the web-log analyzer introduced in the detailed course material. A skeleton **`webloganalyzer.WeblogAnalyzer`** class is supplied in the *chapter07.jar* archive file.

The archive also contains some sample data in the *weblog.txt* file. To read the sample data, your should directory structure should look like

```
<run code from here>
  +--src
  |    +--webloganalyzer
  |    |    +--WeblogAnalyzer.java
  |    |       <other source code files>
  +--weblog.txt
  |--build
       +--classes
  |    |    +--WeblogAnalyzer.class
  |    |       <other bytecode files>
```

where *<run code from here>* is the directory from which you run your *java* command.

Question 1

Correct

Note de 1,00 sur 1,00

Explore the *webloganalyzer* project by creating a **`LogAnalyzer`** object and calling its **`analyzeHourlyData`** method. Follow that with a call to its **`printHourlyCounts`** method, which will print the results of the analysis. What are the busiest times of day?

Réponse : 18 ✓

Vérifier

Correct

Note pour cet envoi : 1,00/1,00.

Question 2

Correct

Note de 1,00 sur 1,00

Add a method **`busiestHour`** to **`LogAnalyzer`** that returns the busiest hour and its access count. You can do this by looking through the **`hourCounts`** array to find the element with the biggest count.

Hint: Do you need to check every element to see if you have found the busiest hour? If so, use a for loop.

Paste just your **LogAnalyzer** code into the Answer box and Check your work.

For example:

Test	Result
`LogAnalyzer analyzer = new LogAnalyzer();`	`18 is b`
`analyzer.analyzeHourlyData();`	
`int[] busiest = analyzer.busiestHour();`	
`System.out.println(busiest[0] + " is busiest hour with " + busiest[1] + " counts");`	

Réponse:

```
package webloganalyzer;

/**
 * Read web server data and analyse hourly access patterns.
 *
 * @author David J. Barnes and Michael Kölling.
 * @version    2016.02.29
 */
class LogAnalyzer {
    // Where to calculate the hourly access counts.
    private final int[] hourCounts;
    // Use a LogfileReader to access the data.
    private final LogfileReader reader;

    /**
     * Create an object to analyze hourly web accesses.
     */
    LogAnalyzer() {
```

Vérifier

	Test	Expected
✓	`LogAnalyzer analyzer = new LogAnalyzer();`	`18 is bu`
	`analyzer.analyzeHourlyData();`	
	`int[] busiest = analyzer.busiestHour();`	
	`System.out.println(busiest[0] + " is busiest hour with " + busiest[1] + " counts");`	

Passed all tests! ✓

Correct

Note pour cet envoi : 1,00/1,00.

Question 3

Correct

Note de 1,00 sur 1,00

Add a method **quietestHour** to **LogAnalyzer** that returns the number of the least busy hour and its access count.

Paste just your **LogAnalyzer** code into the Answer box and Check your work.

For example:

Test	Resu

`LogAnalyzer analyzer = new LogAnalyzer();`
`analyzer.analyzeHourlyData();`
`int[] quietest = analyzer.quietestHour();`
`System.out.println(quietest[0] + " is quietest hour with " + quietest[1] + " counts");`
	9 is

Réponse:

```
package webloganalyzer;

/**
 * Read web server data and analyse hourly access patterns.
 *
 * @author David J. Barnes and Michael Kölling.
 * @version    2016.02.29
 */
class LogAnalyzer {
    // Where to calculate the hourly access counts.
    private final int[] hourCounts;
    // Use a LogfileReader to access the data.
    private final LogfileReader reader;

    /**
     * Create an object to analyze hourly web accesses.
     */
    LogAnalyzer() {
```

Vérifier

	Test	Expec

✓	`LogAnalyzer analyzer = new LogAnalyzer();`
	`analyzer.analyzeHourlyData();`
	`int[] quietest = analyzer.quietestHour();`
	`System.out.println(quietest[0] + " is quietest hour with " + quietest[1] + " counts");`
		9 is

Passed all tests! ✓

Correct

Note pour cet envoi : 1,00/1,00.

Question 4

Correct

Note de 1,00 sur 1,00

Add a method **busiestTwoHours** to **LogAnalyzer** that finds which two-hour period is the busiest. Return the value of the first hour of this period and their total access count.

Paste just your **LogAnalyzer** code into the Answer box and Check your work.

For example:

Test

`LogAnalyzer analyzer = new LogAnalyzer();`
`analyzer.analyzeHourlyData();`
`int[] busiest = analyzer.busiestTwoHours();`
`System.out.println(busiest[0] + " starts busiest two hours with " + busiest[1] + " total cou`

Réponse:

```
package webloganalyzer;

/**
 * Read web server data and analyse hourly access patterns.
 *
 * @author David J. Barnes and Michael Kölling.
 * @version    2016.02.29
 */
class LogAnalyzer {
    // Where to calculate the hourly access counts.
    private final int[] hourCounts;
    // Use a LogfileReader to access the data.
    private final LogfileReader reader;

    /**
     * Create an object to analyze hourly web accesses.
     */
    LogAnalyzer() {
```

Vérifier

	Test

✓	`LogAnalyzer analyzer = new LogAnalyzer();`
	`analyzer.analyzeHourlyData();`
	`int[] busiest = analyzer.busiestTwoHours();`
	`System.out.println(busiest[0] + " starts busiest two hours with " + busiest[1] + " total cou`

Passed all tests! ✓

Correct

Note pour cet envoi : 1,00/1,00.

Question 5

Correct

Note de 1,00 sur 1,00

Modify your **LogAnalyzer** to analyze log data by day counts as well. You'll need to add the methods **analyzeDailyData** and **printDailyCounts**, and whatever else is necessary. You may also find it necessary to modify **LogEntry** somewhat.

Note: daily information starts with one as the first day, not zero as for hourly data.

Paste your **LogAnalyzer** and **LogEntry** code into the Answer box and Check your work.

For example:

Test	Result

`LogAnalyzer analyzer = new LogAnalyzer();`
`analyzer.analyzeDailyData();`
`analyzer.printDailyCounts();`
	`Day: Count`
	`1: 69`
	`2: 108`
	`3: 120`
	`4: 106`
	`5: 35`

	6: 271
	7: 122
	8: 186
	9: 154
	10: 123
	11: 149
	12: 58
	13: 82
	14: 152
	15: 103
	16: 159
	17: 149
	18: 78
	19: 42
	20: 58
	21: 99
	22: 81
	23: 92
	24: 242
	25: 75
	26: 46
	27: 53
	28: 103
	29: 229
	30: 94
	31: 311

Réponse:

```java
package webloganalyzer;

/**
 * Read web server data and analyse hourly access patterns.
 *
 * @author David J. Barnes and Michael Kölling.
 * @version    2016.02.29
 */
class LogAnalyzer {
    // Where to calculate the hourly access counts.
    private final int[] hourCounts;
    private final int[] dayCounts;
    // Use a LogfileReader to access the data.
    private final LogfileReader reader;

    /**
     * Create an object to analyze hourly web accesses.
     */
```

Vérifier

	Test	Expected	Got	
✓	LogAnalyzer analyzer = new LogAnalyzer();	Day: Count	Day: Count	✓
	analyzer.analyzeDailyData();	1: 69	1: 69	
	analyzer.printDailyCounts();	2: 108	2: 108	
		3: 120	3: 120	
		4: 106	4: 106	
		5: 35	5: 35	
		6: 271	6: 271	
		7: 122	7: 122	
		8: 186	8: 186	
		9: 154	9: 154	
		10: 123	10: 123	
		11: 149	11: 149	

		12: 58	12: 58	
		13: 82	13: 82	
		14: 152	14: 152	
		15: 103	15: 103	
		16: 159	16: 159	
		17: 149	17: 149	
		18: 78	18: 78	
		19: 42	19: 42	
		20: 58	20: 58	
		21: 99	21: 99	
		22: 81	22: 81	
		23: 92	23: 92	
		24: 242	24: 242	
		25: 75	25: 75	
		26: 46	26: 46	
		27: 53	27: 53	
		28: 103	28: 103	
		29: 229	29: 229	
		30: 94	30: 94	
		31: 311	31: 311	

Passed all tests! ✓

Correct

Note pour cet envoi : 1,00/1,00.