



CodeCheck Report: trainingMN3P82-M94

[Check out Codility training tasks](#)

Test Name:

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
ParkingBill	5 min	100%

Total score

100%

Tasks Details

Elementary	1. ParkingBill	Task Score	Correctness	Performance
	Given two strings representing times of entry and exit from a car parking lot, find the cost of the ticket according to the given billing rules.			
		100%	100%	Not assessed

Task description

You parked your car in a parking lot and want to compute the total cost of the ticket. The billing rules are as follows:

- The entrance fee of the car parking lot is 2;
- The first full or partial hour costs 3;
- Each successive full or partial hour (after the first) costs 4.

You entered the car parking lot at time E and left at time L. In this task, times are represented as strings in the format "HH:MM" (where "HH" is a two-digit number between 0 and 23, which stands for hours, and "MM" is a two-digit number between 0 and 59, which stands for minutes).

Write a function:

```
class Solution { public int solution(String E, String L); }
```

that, given strings E and L specifying points in time in the format "HH:MM", returns the total cost of the parking bill from your entry

Solution

Programming language used:	Java 8
Total time used:	5 minutes
Effective time used:	5 minutes
Notes:	not defined yet

Task timeline

08:29:1908:34:03

Code: 08:34:03 UTC, java, final, score: 100

[show code in pop-up](#)

at time E to your exit at time L. You can assume that E describes a time before L on the same day.

For example, given "10:00" and "13:21" your function should return 17, because the entrance fee equals 2, the first hour costs 3 and there are two more full hours and part of a further hour, so the total cost is 2 + 3 + (3 * 4) = 17. Given "09:42" and "11:42" your function should return 9, because the entrance fee equals 2, the first hour costs 3 and the second hour costs 4, so the total cost is 2 + 3 + 4 = 9.

Assume that:

- strings E and L follow the format "HH:MM" strictly;
- string E describes a time before L on the same day.

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

Copyright 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Test results - Codility

```
1 // you can also use imports, for example:
2 // import java.util.*;
3 import java.text.ParseException;
4 import java.text.SimpleDateFormat;
5 import java.util.Date;
6
7 // you can write to stdout for debugging purposes,
8 // System.out.println("this is a debug message");
9
10 class Solution {
11     public int solution(String E, String L) {
12         // Implement your solution here
13         int totalCost = 0;
14
15         SimpleDateFormat format = new Simp
16         long difference=0;
17         try {
18             Date date1 = format.parse(
19             Date date2 = format.parse(
20             difference = date2.getTime
21         } catch (ParseException e) {
22             // TODO Auto-generated cat
23             e.printStackTrace();
24         }
25
26         int diffMinutes = (int)(long)(difference /
27         int diffHours = (int)(long)differe
28
29         if(diffMinutes>0) {
30             diffHours+=1;
31         }
32         int totalBillableHours=diffHours;
33         totalCost=2+3+((totalBillableHours
34
35         return totalCost;
36     }
37 }
```

Analysis summary

The solution obtained perfect score.

Analysis

expand all		Example tests
▶	example1	✓ OK
	first example test	
▶	example2	✓ OK
	second example test	
expand all		Correctness tests
▶	under_ten_minutes	✓ OK
	very short parking times, answer is always 5	
▶	random_under_hour	✓ OK
	short parking times, answer is always 5	
▶	equal_hours_small	✓ OK
	parking for short time, always with complete hours	
▶	random	✓ OK
	randomly generated test cases	
▶	mixed	✓ OK
	medium and short intervals	
▶		

Test results - Codility

equal_hours_big	✓ OK
long parking time, for complete hours	
▶ random_big	✓ OK
randomly generated parking times, for at least 20 hours	
▶ maximum_result	✓ OK
test cases giving maximum results or almost maximum results	