Assignment 2 - Time Difference

Description

Write a program that accepts user input of two time values and computes the difference between them. The times are read into the program as integers of at most 6 digits (without leading zeros): **HHMMSS**, where **HH** represents hours according to a 24-hour clock (0-23), **MM** represents minutes (0-59) and **SS** represents seconds (0-59). The difference should be displayed in the same format.

The difference can be either positive or negative. It is obtained by subtracting the second time from the first time - make sure that you do not subtract the first time from the second, as this will produce incorrect answers.

For the purposes of this assignment, you may assume that the inputs are correct--that is, the input values will be valid positive integers no more than 6 digits in length and the hours, minutes and seconds will be within the proper ranges (e.g., seconds will be within 0-59).

Please note that you are not being asked to convert an AM/PM 12-hour format to a 24- hour format.

Example Output

```
Enter first time: 230000
Enter second time: 210001
Time difference: 15959
Enter first time: 123245
Enter second time: 112955
Time difference: 10250
Enter first time: 23245
Enter second time: 32815
Time difference: -5530
Enter first time: 245
Enter second time: 235
Time difference: 10
```

These example cases will not be the only cases run by the grader so be sure to try your own to confirm all is working as it should.

Design and Implementation

Before any calculations can begin, input from the user must be gathered. A Scanner object will be used to gather times from the user. Recall that a Scanner object can be used more than once, so you should not need to create two different Scanner objects in order to get two different inputs. Please do not use the *Swing* package in Java or the option pane object, as they can make it difficult for running the grading scripts.

How to find the time difference is up to you, but the following describes one such algorithm that you could use:

. **1** Isolate the different parts of each time: hours, minutes and seconds (*hint: use the mod operator % and/or integer division*).

. **2** Compute the times in terms of seconds (1 hour = 3600 seconds, 1 minute = 60 seconds).

. **3** Calculate the difference between the times in terms of seconds.

. **4** Isolate hours, minutes and seconds from the difference value (*again, use the mod operator % and/or integer division, but with a different base ... hint, hint, hint*).

. **5** Create the new integer time value in the form **HHMMSS** and output it.

Since the algorithm above uses mathematical techniques in order to find the correct answer, the algorithm that you choose should do the same. **PLEASE NOTE:** It will not be acceptable to use string manipulation techniques (parsing a string for certain numbers, concatenating different numbers into an output, etc.) as a way of finding the answer.

Be careful about when to have zeroes between non-zero values. For example, there is a material difference between “10250” (1 hour, 2 minutes, 50 seconds) and “1250” (12 minutes, 50 seconds).Make sure that there is only one negative sign if the time difference is negative. Avoiding string manipulation will in turn help you to avoid problems where the output might be “-55-30” instead of “-5530”.

Formatting and Style

However you design the user prompt is fine, but try to make sure that the user is not confused about when or where to enter input. For instance, if nothing is printed to the screen (such as “Enter first time: ”) then the user might be confused about what to do. Also, it is a good idea to make sure that the input and output are on separate lines.

In your code, keep in mind that you will likely use some constants (such as 3600 for the number of seconds in an hour). It would be a good idea to create final int type variables for these constants and then use them where they are needed; it will help to make the code more readable and it avoids the issue of “magic numbers”.

Lastly, although it might seem clever to handle all of the calculations in a “one-liner”, try instead to break apart the calculations through several lines so that your code reads better. Your code should be written in such a way that it provides a clear idea of how the program’s flow works (i.e., a lot of comments will not be necessary).

Grading Rubric

Points	**Criteria**
2.5	Gathers user input correctly
4	Calculates the time difference correctly
2.5	Outputs the correct answers in HHMMSS format
1	Good style demonstrated in the code; sensible formatting