# High-Quality Code Exam – Theatre

A junior developer Huy Phuong Nguyen from Vietnam was once assigned to design and implement an information system for theatres and performances. The original problem statement is given below:

In the city there are several theatres. Each of them has a name and a timetable that holds all performances, their start date and time, their duration and ticket price. Your task is to model the theatres, performances and the timetable and create a console-based application that executes a sequence of commands:

* **AddTheatre(theatre)** – adds a theatre by given **theatre** name. The theatre name is unique among all theatres. Prints “**Theatre added**” as result or “**Error:** **Duplicate theatre**” in case of name conflict.
* **PrintAllTheatres()** – prints all theatres in the city, ordered by name, separated by comma and space, or "**No theatres**" in case of empty list of theatres.
* **AddPerformance(theatre, performance, date-and-time, duration, price)** – adds a **performance** to the specified **theatre** with the specified **date-and-time**, **duration** and ticket **price**. Performances are not allowed to overlap: at most one performance can take place at given moment of time for given theatre. Prints “**Performance added**” as a result or “**Error:** **Theatre does not exist**” in case of non-existing theatre or “**Error:** **Time/duration overlap**” in case of overlapping performances.
* **PrintAllPerformances()** – prints all performances in the city in format **(performance, theatre, date-and-time)**, ordered by theatre (as first criteria) and by date and time (as second criteria), separated by comma and space, or "**No performances**" in case of empty list of performances.
* **PrintPerformances(theatre)** – prints all performances for the specified theatre in format **(performance, date-and-time)**, ordered by date and time, separated by comma and space, or "**No performances**" in case of empty list of performances for the specified theatre. Print “**Error:** **Theatre does not exist**” in case of non-existing theatre.

The **input** could contain up to 50 000 commands so the efficiency is important. All strings in the commands (e.g. theatre names and performance titles) consist of alphabetical characters, numbers and spaces. Prices are real numbers with up to 2 digits after the decimal point "**.**" (e.g. 122.55, or 220). Date and time is always given and printed in 24-hours format: **dd.MM.yyyy HH:mm** (e.g. 23.01.2015 09:20). Empty lines in the input should be skipped.

The **output** should be printed at the console and should contain one text line corresponding to each command from the input.

|  |  |  |  |
| --- | --- | --- | --- |
| Sample Input  |  | | --- | | PrintAllPerformances()  PrintAllTheatres()  AddTheatre(Theatre Sofia)  AddTheatre(Theatre 199)  AddTheatre(Theatre Sofia)  PrintAllTheatres()  AddTheatre(Ivan Vazov National Theatre)  AddPerformance(Theatre 199, Duende, 20.01.2015 20:00, 1:30, 14.5)  AddPerformance(Theatre 199, Bella Donna, 20.01.2015 20:30, 1:00, 12)  AddPerformance(Theatre 199, Bella Donna, 20.01.2015 21:30, 1:00, 12)  AddPerformance(Theatre Sofia, Bella Donna, 20.01.2015 19:30, 1:00, 15)  AddPerformance(Theatre Sofia, Don Juan, 20.01.2015 20:31, 2:00, 14.60)  AddPerformance(Art Theatre, Don Juan, 28.12.2014 19:30, 1:45, 10)  AddPerformance(Theatre 199, Frozen, 18.01.2015 10:00, 1:00, 8)  AddPerformance(Theatre 199, Star Gate, 20.01.2015 22:00, 1:50, 6.45)  AddPerformance(Theatre 199, Don Juan, 18.01.2015 19:30, 2:00, 14.60)  AddPerformance(Theatre 199, I Pay, 18.01.2015 18:00, 1:30, 14.60)  PrintAllPerformances()  PrintPerformances(Ivan Vazov National Theatre)  PrintPerformances(Art Theatre)  PrintPerformances(Theatre 199)  AddPerformance(Theatre 199, Long Story, 18.01.2015 11:00, 12:00, 100)  AddPerformance(Theatre 199, Short Story, 18.01.2015 19:40, 0:10, 0.30) | | Sample Output  |  | | --- | | No performances  No theatres  Theatre added  Theatre added  Error: Duplicate theatre  Theatre 199, Theatre Sofia  Theatre added  Performance added  Error: Time/duration overlap  Error: Time/duration overlap  Performance added  Performance added  Error: Theatre does not exist  Performance added  Performance added  Performance added  Error: Time/duration overlap  (Frozen, Theatre 199, 18.01.2015 10:00), (Don Juan, Theatre 199, 18.01.2015 19:30), (Duende, Theatre 199, 20.01.2015 20:00), (Star Gate, Theatre 199, 20.01.2015 22:00), (Bella Donna, Theatre Sofia, 20.01.2015 19:30), (Don Juan, Theatre Sofia, 20.01.2015 20:31)  No performances  Error: Theatre does not exist  (Frozen, 18.01.2015 10:00), (Don Juan, 18.01.2015 19:30), (Duende, 20.01.2015 20:00), (Star Gate, 20.01.2015 22:00)  Error: Time/duration overlap  Error: Time/duration overlap | |

You are given the original source code from Huy Phuong designed to solve the above problem. Your task is to refactor it to improve its quality, fix any bugs, write unit tests, write some documentation and fix the performance bottlenecks.

## Problem 1. Code Refactoring

**Refactor the source code** to improve its quality following the best practices introduced in the course  
“[High-Quality Code](https://softuni.bg/courses/high-quality-code/)”. You are not allowed to modify the IPerformanceDatabase interface. Any other refactorings are welcome if they improve the code quality.

**36 score**

## Problem 2. Bug Fixing

**Debug the code**, **find and fix the bugs** in it.

**6 score**

## Problem 3. Unit Testing

Design and implement **unit tests for** **all methods of the** IPerformanceDatabase **interface**. Testing the other code is not required. The **code coverage** should be **at** **least 90% for the classes implementing** IPerformanceDatabase (you do not need to cover the class that parses the input commands and prints the output). Be sure to test all major execution scenarios + all interesting border cases and special cases. Use Visual Studio Team Test (VSTT) and VS code coverage.

**30 score**

## Problem 4. Code Documentation

**Document all the methods in the** IPerformanceDatabase **interface** and the IPerformanceDatabase **interface declaration** using C# XML documentation. Any other documentation is **not** required.

**5 score**

## Problem 5. Performance Bottlenecks

Find any **performance bottlenecks** and briefly describe them with a **comment in the code**.

**Fix the problems** if possible (and leave the bottlenecks descriptions in addition to the fixes).

**7 score**

## Problem 6. Correct Results in the Judge System

You are given an automated judge system to submit your solution (<http://judge.softuni.bg/Contests/65/High-Quality-Code-23-Jan-2015>). If your code is correct (all bugs are fixed) and runs fast enough (the performance bottlenecks are fixed), your solution will pass all the tests. The last 2 tests measure performance. The others measure correctness.

**16 score**