# High-Quality Code Exam – Theatre

This document defines the evaluation criteria for the "Theatre" exam from the "High-Quality Code" course.

## Code Refactoring

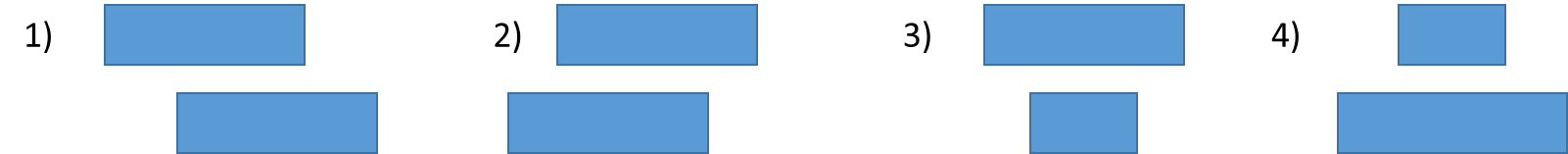
* Fix the **incorrect formatting** in the code correctly (3 score)
* Merge the unneeded classes Class1 and Class2 with the class holding the Main() method (2 score)
* Split the code into **separate files** (each class in separate file) (2 score)
* Fix the **incorrect names** in the code: (5 score)
  + Huy-Phuong, NhàHát, universal, s2, sortedDictionaryStringSortedSetPerformance, BuổIDiễNDatabase, kiemTra, ss, ee, ss2, ee2, asfd, tt, t2, pp, thoiGian, ThoiGian, BuoiDien, tr23, tr32, e2, gia, chiHuyLine, chiHuy, chiHuyParts, chiHuyResult, chiHuyParts1, chiHuyParams, chiHuyParams1, tmp, ps, result2, result1, …
* Fix the visibility of classes / class members (public **/** private **/** protected **/** internal**)** members (2 score)
* Remove the unneeded comments: (2 score)
  + The authors name and contacts
  + The comment "*Do not modify the interface members*"
  + All "*TODO*" comments
* Remove the **duplicated code for parsing the parameters** in Main() method (3 score)
* Split the Main() method into several smaller methods with clear intent (5 score)
  + Method parsing the parameters
  + Methods for each of the commands
  + See the author's solution for an example
* Extract **constants**: (3 score)
  + DateTimeFormat = "dd.MM.yyyy HH:mm";
  + DurationFormat = "hh':'mm";
  + NumberFormat = "f2";
  + Command result messages like "*Theatre added*" and "*Duplicate theatre*"
* Improve the logic that checks for **overlapping time intervals** (2 score)
  + It is putoriginally in the method kiemTra()
  + The method could be split into pieces to improve code quality
* Fix the ToString() method in the Performance class (originally named gia) (2 score)
  + This method should be either deleted (because the main program does not use it) or refactored to correctly represent a performance as string.
  + The method is designed to simplify the unit testing, but can be avoided.
* Fix all **StyleCop** issues (5 score)

## Bug Fixing

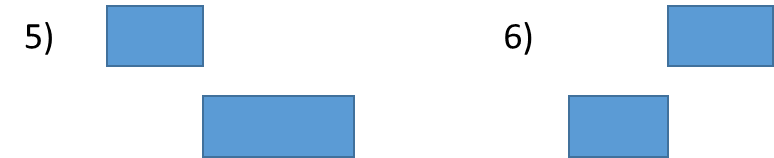
* Incorrect culture (locale): new CultureInfo("vi-VN") 🡪 CultureInfo.InvariantCulture (2 score)
* Incorrect command name: case "PrintAllTheaters" 🡪 case "PrintAllTheatres" (2 score)
* In AddTheatre(…) the first check is inverted: if (! ….ContainsKey) 🡪 if (….ContainsKey) (2 score)

## Required Unit Tests

* AddTheatre(theatre) – add a theatre, check sorting an duplicates (5 score)
  + Add a few theatres, list the theatres and check whether the list is correct
  + Add a theatre twice and except DuplicateTheatreException
  + Give less score if only the count of theatres is asserted without the **content** and the **correct** **order**
* PrintAllTheatres() (4 score)
  + Add a few theatres and assert they are correctly added (correct count, correct order, correct names)
  + Give less score if only the count of theatres is asserted only without the content and the order
  + Try duplicating theatres and assert that duplicated theatres are printed only once
* AddPerformance(…) usual cases (6 score)
  + Add a few performances and assert they are correctly added (correct count, order and data)
  + Assert the order of the performances is correct, not just the count (use PrintAllPerformances())
  + Add performance for non-existing theatre and expect TheatreNotFoundException
* AddPerformance(…) **overlapping** (6 score)
  + We have 4 overlapping scenarios between a new time interval and an existing time intervals:



* + We should check the border cases when start == end



* PrintAllPerformances()tests (2 score)
  + Add a few non-overlapping performances and assert theiy are printed in the correct order (sorted by theatrte, then by date and time)
  + Test with zero performances
* PrintPerformances(theatre)tests (4 score)
  + Add a few non-overlapping performances and assert theiy are printed in the correct order
  + Try non-existign theatre and expect TheatreNotFoundException
  + Test with zero performances for a certain theatre
* **Code coverage** >= 90% (3 score)

## Code Documentation

* 5 score, ~ 1 score per documented method

## Performance Issues

* PrintAllTheatres() 🡪 theatres are added and removed many times to a LinkedList<string> without any need
  + Problem found: 1 score
  + Problem fixed: +2 score
  + The judge test #15 (the test before the last) indicates whether the problem is fixed.
* PrintAllPerformances() 🡪 the performances are concatenated in a string, which is slow
  + Problem found: 2 score
  + Problem fixed: +2 score
  + The judge test #16 (the last test) indicates whether the problem is fixed.

## Judge System

* 16 tests, each gives 1 score