# Data Structures Exam: Judge

## Description

You need to implement a structure for an online judge system used to test students. You are given a Submission class, which has the following properties:

* Int Id – unique id for each submission.
* SubmissionType Type – An enumeration specifying the programming language of the submission.
* Int UserId – Id of the user who submitted the submission.
* Int ContestId – Id of the contest to which the submission was submitted.
* Int Points – The points which the submission has earned.

SubmissionType is an enumeration of programming languages that will be given in the skeleton, the possible values are:

* CSharpCode
* JavaCode
* JavaScriptCode
* PhpCode

You need to support the following operations (and they should be **fast**):

* AddUser(int userId) – Adds a User to the Judge system, attempting to add the same User twice should result in no change happening (the User should not be added again and no exception should be thrown). HashSet<User>
* AddContest(int contestId) – Adds a Contest to the Judge system, attempting to add the same Contest twice should result in no change happening (the contest should not be added again and no exception should be thrown). HashSet<Contest>
* AddSubmission(Submission submission) – Adds a Submission to the Judge system. The submission carries info about which Contest and User it belongs to. Any attempt to add a Submission with a non-existing User or non-existing Contest should throw an InvalidOperationsException. HashSet<Submission>
* GetUsers() – Should return all users in the system sorted by **Id** in **ascending** order.
* GetContests() – Should return all contests in the system sorted by **Id** in **ascending** order.
* GetSubmissions() – Should return all submissions in the system sorted by **Id** in **ascending** order.
* DeleteSubmission(int submissionId) – Should delete the submission with the given id from the system. Dictionary<int,Submission>
* ContestsBySubmissionType(SubmissionType submissionType) – Returns all contests which contain submissions with the given submissionType. For example all contests which have submissions written in Java. Dictionary<Submission,Contest>
* SubmissionsInContestIdByUserIdWithPoints(int points, int contestId, int userId) – Returns all submissions from the specified contest, by the specified user which have the specified amount of points. For example all submissions in contest 3, by user 4 with exactly 50 points. ??????
* SubmissionsWithPointsInRangeBySubmissionType(int minPoints, int maxPoints, SubmissionType submissionType) – Returns all submissions with the given submissionType, which have points between minPoints (inclusive) and maxPoints (inclusive). For example all CSharpCode submissions which have between 20 and 30 points.
* ContestsByUserIdOrderedByPointsDescThenBySubmissionId(int userId) – Returns all contests in which the user has submissions, ordered by submission points in descending order as a first criteria and by submission Id in ascending order as a second criteria. In other words return the contests in which the user has performed the best first. For example if you have 4 submissions:  
  s1 with id 1, 30 points in contest c1
  + s2 with id 2, 28 points in contest c2
  + s3 with id 3, 30 points in contest c3
  + s4 with id 4, 29 points in contest c1

The correct order returned should be c1, c3, c2. Submission s4 and s1 are in the same contest and s1 has more points so clearly s1 is the best submission for the user in contest c1. Submission s2 has less points than submissions s1 and s3, so c2 is clearly the last contest. Submissions s1 and s3 have the same points but s1 has a lower Id so contest c1 should be first in the result, this leaves contest c3 as the second. Dictionary<int,Submission>

The equivalent exception in java is UnsupportedOperationException.

## Input / Output

You are given a **Visual Studio C# project skeleton** (unfinished project) / **IntelliJ Java project** holding the interface IJudge, the unfinished classes Judge and Submission. **Tests** covering the Judge **functionality** and **performance**.

Your task is to **finish this class** to make the tests run correctly.

* You are **not allowed to change the tests**.
* You are **not allowed to change the interface**.
* You can add to the Judge class, but don't remove anything.
* You can edit the Judge class if it implements the IJudge interface.

## Interface

The interface IJudge in C# looks like the code below:

|  |
| --- |
| public interface IJudge  {  void AddContest(int contestId);  void AddSubmission(Submission submission);  void AddUser(int userId);    void DeleteSubmission(int submissionId);  IEnumerable<Submission> GetSubmissions();  IEnumerable<int> GetUsers();  IEnumerable<int> GetContests();  IEnumerable<Submission> SubmissionsWithPointsInRangeBySubmissionType(int minPoints, int maxPoints, SubmissionType submissionType);  IEnumerable<int> ContestsByUserIdOrderedByPointsDescThenBySubmissionId(int userId);  IEnumerable<Submission> SubmissionsInContestIdByUserIdWithPoints(int points, int contestId, int userId);  IEnumerable<int> ContestsBySubmissionType(SubmissionType submissionType);  } |

## Submission

Submit an archive (.zip) of the source code. Your code **mustn't** contain namespaces/packages.

## Scoring

Each implemented method brings you a specific amount of points, some of the points are awarded for correct behavior, others for performance. **The performance tests might not work on your PC**. You need to cover all tests in each group to receive points. Bellow is a breakdown of all points by methods:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Correct Behavior | Performance | Total |
| Overall | 70 | 130 | 200 |