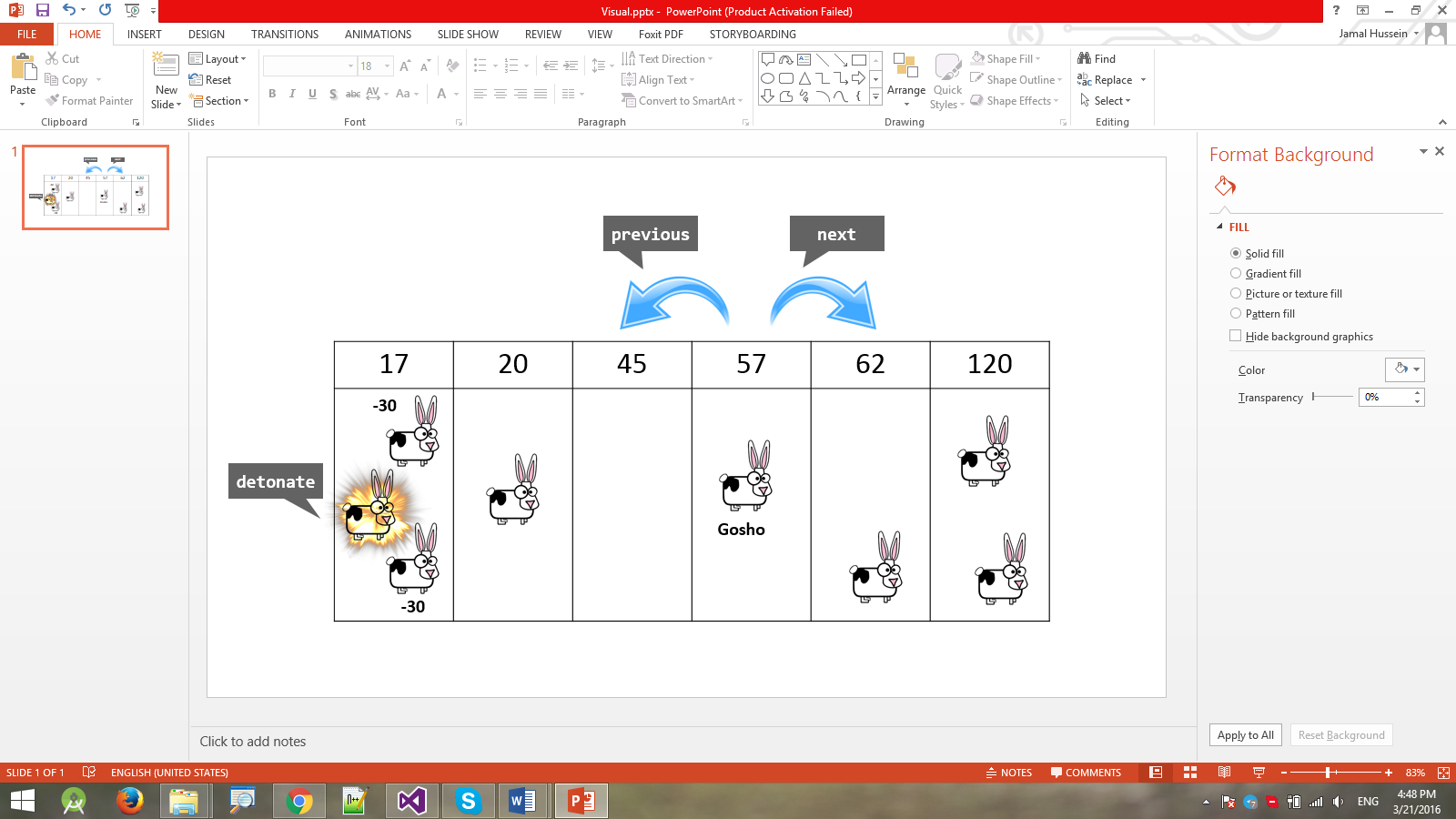
# Bunny Wars – Data Structures Exam

**Bunny Wars** is an interactive game. The rules are as follows:

* There are **several rooms** in the game, each with a **unique id**. Rooms are situated according to their **id** in ascending order.
* Each room can hold **several bunnies**.
* A bunny has a **unique** **name**, **health**, **score** and **team**. All bunnies have **health = 100** and **score = 0**   
  by default.
* Bunnies can jump to the **next** or **previous** room.
* Bunnies can **detonate**, causing damage to all bunnies of other teams in the room (**detonating** does not delete the detonated bunny – it’s still in the room after the command).



Your task is to design a data structure in C# or Java that supports the commands listed below in a **fast and efficient way**.

* Add roomId– adds a room to the structure. Rooms have unique ids. Rooms should be situated according to **their id in ascending order**.
  + If a room with the given Id exists the command should throw an exception.
* Add bunnyName teamId roomId– creates a new bunny with the given **name** and **team id** and adds it to the **specified room**.
  + There can be a maximum of 5 teams in the game, teamId will always be in the range [0..4]**.**
  + If a bunny with the given name already exists or a room with the given Id does not exist, the command should throw an exception.
* BunnyCount – returns the total amount of bunnies in the structure.
* RoomCount – returns the total amount of rooms in the structure.
* Remove roomId– removes the given **room** from the game. **All bunnies in the room** are also removed from the game.
  + If a room with the given **Id** does not exist, the command should throw an exception.
* Next bunnyName– moves the specified bunny in the **next room** (to the right of the current room). If a bunny is in the last room (the room with the biggest Id) and receives this command he will move to the first room (the room with the lowest Id).
  + If a bunny with the given name does not exist, the command should throw an exception.
* Previous bunnyName – moves the specified bunny in the **previous room** (to the left of the current room). If a bunny is in the first room (the room with the lowest Id) and receives this command he will move to the last room.
  + If a bunny with the given name does not exist, the command should throw an exception.
* Detonate bunnyName– detonates the bunny, causing all bunnies from other teams in the same room to suffer 30 damage to their health (their health is reduced by 30).
  + If a bunny with the given name does not exist, the command should throw an exception.
  + If a bunny falls to 0 or less health as a result of the detonation, it should be removed from the game.
  + For each removed enemy bunny, the detonated bunny should gain +1 score.
* ListBunniesByTeam teamId - returns all bunnies from the specified team in (sorted by name in descending order).
* ListBunniesBySuffix suffix - returns all bunnies ending with the **specified suffix** (sorted by the ASCII code of the reversed name in ascending order as a first criteria and by length in ascending order as a second criteria)**.** Example **Tpen < apen < aapen < bapen < bpen.**

### Input and Output

You are given a **Visual Studio C# project skeleton** (unfinished project) / **IntelliJ Java project** holding the interface IBunnyWarsStructure, the unfinished class BunnyWarsStructure and **tests** covering its **functionality** and its **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**.

### Constraints

* All names:
  + Consist of **Latin letters and digits**.
  + Have length in the range [1...100].
* Room IDs will be integer values in the range [-231..231].
* All string matching operations are **case-sensitive**.

### Submissions

Submit an archive (.zip) of the source code + external libraries.

# Scoring

Each implemented method brings you a specific amount of points, some of the points are awarded for correct behavior, others for performance. You need to cover all tests in a given group in order to receive points. Bellow is a breakdown of all points by methods:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Correct Behaviour | Performance | Total |
| Add Room | 2 | 4 | 6 |
| Add Bunny | 3 | 6 | 9 |
| BunnyCount | 1 |  | 1 |
| RoomCount | 1 |  | 1 |
| Remove | 4 | 10 | 14 |
| Next | 3 | 8 | 11 |
| Previous | 3 | 8 | 11 |
| Detonate | 4 | 10 | 14 |
| ListBunniesByTeam | 4 | 10 | 14 |
| ListBunniesBySuffix | 5 | 14 | 19 |
| Overall: | 30 | 70 | 100 |