# Data Structures Exam: Microsystem

## Description

You have to implement a structure that keeps track of the Microsystem store for computers. Your structure will have to support the following functionalities:

* **CreateComputer(computer)**– you have tocreate a new computer and add it to the store. If there is already a computer with that number, throwArgumentException
* **Contains(int number)** – checks if a computer with the provided number exists in the store.
* **Count() –** returns the count of computers in the store
* **GetComputer(number) –** returns the computer with the given number. If there isn’t such throw ArgumentException.
* **Remove(int number) –** removes the computer with the provided number. If there isn’t such throw ArgumentException
* **RemoveWithBrand(brand) –** removes all computers with the given brand. If there aren’t any throw **ArgumentException**
* **UpgradeRam(ram, number) –** finds the computer with the given number and sets its ram to the given one (only if the given one is bigger). If there isn’t a computer with the provided number throw**ArgumentException**
* **GetAllFromBrand(brand) –** finds all computers with the provided brand. Order them by price descending. If there aren’t any return empty collection.
* **GetAllWithScreenSize(screenSize) –** finds all computers with screen size equal to the given. Order them by number descending. If there aren’t any return empty collection.
* **GetAllWithColor(color) –** finds all computers with the same color as the given. Order them by price descending.If there aren’t any return empty collection.
* **GetInRangePrice(minPrice, maxPrice) –** finds all computers with price between the given inclusive. Order them by price descending. If there aren’t any return empty collection.

*Feel free to override Equals() and GetHashCode() if necessary.*

## Input/Output

You are given a **Visual Studio C# project skeleton** (unfinished project) / **IntelliJ Java project** holding the interface IMicrosystems, the classesMicrosystemandComputer.**Tests** covering the Microsystems**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 Computer class, but don't remove anything.
* You can edit the Microsystems class if it implements the IMicrosystems interface.

## Interface

The interface IMicrosystemin C# looks like the code below:

|  |
| --- |
| publicinterfaceIMicrosystem  {  void CreateComputer(Computer computer);  bool Contains(int number);  int Count();  void Remove(int number);  void RemoveWithBrand(Brand brand);  void UpgradeRam(int ram, int number);  IEnumerable<Computer> GetAllFromBrand(Brand brand);  IEnumerable<Computer> GetAllWithScreenSize(int screenSize);  IEnumerable<Computer> GetAllWithColor(string color);  IEnumerable<Computer> GetInRangePrice(decimal minPrice, decimalmaxPrice);  } |

The interface Microsystemin Java looks like the code below:

|  |
| --- |
| **public interface Microsystem {**  **void createComputer(Computer computer);**  **boolean contains(int number);**  **int count();**  **Computer getComputer(int number);**  **void remove(int number);**  **void removeWithBrand(Brand brand);**  **void upgradeRam(int ram, int number);**  **Iterable<Computer> getAllFromBrand(Brand brand);**  **Iterable<Computer> getAllWithScreenSize(double screenSize);**  **Iterable<Computer> getAllWithColor(String color);**  **Iterable<Computer> getInRangePrice(double minPrice, double maxPrice);**  **}** |

## 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 | 50 | 100 | 150 |