# Drones

## Preparation

Download the skeleton provided in Judge. **Do not** change the **StartUp** class or its **namespace**.

## Problem description

Your task is to create an airfield where drones can take of and land.

# Drone

You are given a class **Drone,** create the following fields:

* **Name: string**
* **Brand: string**
* **Range: int**
* **Available: boolean - true by default**

The class **constructor** should receive **(name, brand, range)**.

The class should also have a method:

* Override the **ToString()** method in the format:

**"Drone: {name}**

**Manufactured by: {brand}**

**Range: {range} kilometers"**

# Airfield

Next, a class named **Airfield**is given that has a **collection**(**drones**) of type **Drone**. The name of the collection should be **Drones**. All the entities of the **Drones** collection have the **same** properties. The **Airfield** has also some additional properties:

* **Name: string**
* **Capacity: int**
* **LandingStrip - double**

The **constructor** of the **Airfield** class should receive the **name, capacity, and landing strip**.

Implement the following features:

* Getter **Count** - returns the count of the drones in the airfield.
* string AddDrone(Drone drone) - **adds** a drone to the drone's collection **if** **there** **is** **room** for it. Before adding a drone, check:
  + - * If the **name** or **brand** are **null or empty**.
      * If the **range** is **NOT** between **5-15 kilometers**.

If the **name, brand,** or **range** properties are not valid, return: **"Invalid drone.".** If the airfield is full (there is no room for more drones), return **"Airfield is full.".** Otherwise, return: **"Successfully added {droneName} to the airfield."**

* bool RemoveDrone(string name) - removes a drone by **given name,** if such **exists return true**, otherwise **false.**
* int RemoveDroneByBrand(string brand) - removes **all drones** by the given **brand,** if such **exists, return how many drones were removed,** otherwise **0.**
* Drone FlyDrone(string name) method – **fly** (**set** its available **property** to **false** without removing it from the collection) the **drone** with the **given name if exists**. As a result **return** the **drone, or null if does not exist.**
* List<Drone> FlyDronesByRange(int range) method - **fly** and returns **all drones** which have a range **equal or bigger** to the given. **Return a list of all drones which have been flown.** The range will always be valid.
* **Report()** -returns information about the airfield and drones which are **not in flight** in the following format:
  + **"**Drones **available at {airfieldName}:  
    {**Drone**1}  
    {**Drone**2}  
    (…)**"

**Note: Do not use** "\n\r" **for a new line.**

## Constraints

* The **names** of the drones will be **always unique**.
* You will always have a drone added before receiving methods manipulating the Airfield’s drones.

## Examples

This is an example of how the **Airfield class** is **intended to be used**.

|  |
| --- |
| Sample code usage |
| // Initialize the repository (Airfield)  Airfield airfield = new Airfield("Heathrow", 10, 10.5);  // Initialize entity  Drone drone = new Drone("D20", "DEERC", 6);  //Print Drone  Console.WriteLine(drone);  // Drone: D20  // Manufactured by: DEERC  // Range: 6 kilometers  // Add Drone  Console.WriteLine(airfield.AddDrone(drone)); // Successfully added D20 to the airfield.  Console.WriteLine(airfield.Count); // 1  // Remove Drone  Console.WriteLine(airfield.RemoveDrone("DE51")); // False  Drone secondDrone = new Drone("CW4", "Cheerwing", 8);  Drone thirdDrone = new Drone("X5SW-V3", "Cheerwing", 7);  Drone fourthDrone = new Drone("X20", "Cheerwing", 4);  Drone fifthDrone = new Drone("EVO2", "Autel", 10);  Drone sixtDrone = new Drone("XL5-6S-FPV", "iFlight", 10);  // Add Drones  Console.WriteLine(airfield.AddDrone(secondDrone)); // Successfully added CW4 to the airfield.  Console.WriteLine(airfield.AddDrone(thirdDrone)); // Successfully added X5SW-V3 to the airfield.  Console.WriteLine(airfield.AddDrone(fourthDrone)); // Invalid drone.  Console.WriteLine(airfield.AddDrone(fifthDrone)); // Successfully added EVO2 to the airfield.  Console.WriteLine(airfield.AddDrone(sixtDrone)); // Successfully added XL5-6S-FPV to the airfield.  // Fly drone by name  Console.WriteLine(airfield.FlyDrone("CW4"));  // Drone: CW4  // Manufactured by: Cheerwing  // Range: 8 kilometers  Console.WriteLine("-----------------FlyDronesByRange-----------------");  List<Drone> flyDrones = airfield.FlyDronesByRange(10);  foreach (var droneToFly in flyDrones)  {  Console.WriteLine(droneToFly);  }  /\*  Drone: EVO2  Manufactured by: Autel  Range: 10 kilometers  Drone: XL5-6S-FPV  Manufactured by: iFlight  Range: 10 kilometers  \*/  // Remove drone by brand  Console.WriteLine(airfield.RemoveDroneByBrand("Cheerwing")); // 2  Console.WriteLine("----------------------Report----------------------");  Console.WriteLine(airfield.Report());  /\*  Drones available at Heathrow:  Drone: D20  Manufactured by: DEERC  Range: 6 kilometers  \*/ |

## Submission

Zip all the files in the project folder except **bin** and **obj** folders.