## High-Quality Code Exam – Air Conditioner Testing System

As an expansion program for your company, you were assigned to a new branch tasked with coming up with new software products. Luckily for you, your new partner **Yakushiba Nadowara** has some revolutionary new ideas, last week he proposed to your chief to create an Air Conditioner Tester System to establish yourself in the Japanese market. Since the Japanese hold quality to a high standard having a product which can efficiently test the new Air Conditioners will be a big hit on the market. To that idea your colleague has already written the main code for the system, however he did a little “localization” to make it more “user friendly” for the local Japanese market. After seeing his “improvements” you decide it’s better for the future of the project if you take over and refactor it.

Your task is to **refactor the code**, using all best practices in **object-oriented design** and **object-oriented programming**, **SOLID** principles, and **design patterns**. You have to **improve the code quality**. You also have to **fix any bugs** that maybe left, and **improve the general performance** (execution speed) of the code. Since your colleague didn’t understand from unit testing he didn’t **write** any **unit tests**, so they’re also left for you.

You are given the original code and the design document, specifying the task at hand. Your colleague also left you two sample test cases to check how the application works. These documents are provided below.

**Ensure the application follows the design document strictly.**

## Overview

The Air Conditioner Tester System holds information about different types of **air conditioners** and **Reports.**

There are three types of air conditioners:

* **Stationary air conditioners** which have a **manufacturer**, a **model**, **power usage** and a **required energy efficiency rating.**
* **Car air conditioners** which have a **manufacturer**, a **model** and **volume** **covered.**
* **Plane air conditioners** which have a **manufacturer**, a **model**, **volume covered** and **electricity used**.

**Reports** have a **manufacturer**, a **model** and a **mark** (Passed or Failed).

Keep in mind an air conditioner’s **manufacturer** and **model** are **unique**, **duplicates are not allowed** even for different types of air conditioners.

The system contains methods to **register** and **test** air conditioners. When an **air conditioner** is **registered** a new entry is made with its **manufacturer** and **model** anddepending on the type of air conditioner either **power usage** and **required** **energy efficiency rating** for **stationary air conditioners,** **volume covered** for **car air conditioners** or **volume covered** and **electricity used** for **plane air conditioners**.

The system can also **test** registered air conditioners, **testing** a registered air conditioner produces a **Report** that holds the air conditioner’s **manufacturer**, its **model** and a **mark** which signifies if the air conditioner meets the expected quality. The ways to calculate whether a given type of air conditioner passes or fails the testing are given bellow:

**For stationary air conditioners:**

If the energy efficiency rating of the air conditioner is better than or equal to the **required energy efficiency rating**, the air conditioner passes the test, otherwise it fails. The energy efficiency rating can be determined from the **power usage** using the following table (E being the worst rating and A being the best):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | E | D | C | B | A |
| Power Usage(KW/h) | >2000 | 1501-2000 | 1251-1500 | 1000-1250 | <1000 |

**For car air conditioners:**

If the **square root** of the **volume covered** is **less than** **3** the test **fails**, otherwise it passes.

**For plane air conditioners:**

If the **electricity usage** divided by the **square root** of the **volume covered** is **less than** **150** the test **passes**, otherwise it fails.

The system also implements ways to track its own status with methods for finding an **air conditioner by its manufacturer and model**,finding a **Report by its manufacturer and model**, finding **all reports from a specified manufacturer** and getting the system **status**.

## System design

The core of the system is **the engine**, it reads **commands** from the standart input (console). In case a **null** or an **empty line** is read the engine finishes excecution, otherwise it **trims the input** and passes it to a **controller**. In case of an **incorrect command (incorrect name, incorrect amount of parameters passed or incorrect parameter format)** an **InvalidOperationException** with the message "Invalid command" is thrown, in case of a **correct command** it’s passed to the controller. If **any kind of exception is thrown,** either because of an incorrect command or a correct command, the engine catches it and prints the exception’s message on the standart output(console) instead.

A sample command is shown below:

CommandName (value1,value2,...)

**Command names** will consist only of **Latin letters**, **values** will consist only of **Latin letters, numbers, spaces and dashes** (-). **Values** will be seperated by a **single coma**(,). The **values** will be surrounded by exactly **2 brackets** – an opening bracket right before the first value and a closing bracket right after the last. The **command name** and the brackets with values will be seperated by a **single space**. All commands given will be in the format specified above.

The engine delegates all actions to **a controller which contains all information about the air conditioner tester system**.

**Models** are classes containing information about the real-world objects the system works with. The system should support **Stationary Air Conditioners, Car Air Conditioners, Plane Air Conditioners** and **Reports**. Not all models are valid. The validation rules for the models are given below:

* An air conditioner’s manufacturer must be at least 4 symbols long.
* An air conditioner’s model must be at least 2 symbols long.
* A stationary air conditioner’s required energy efficiency rating must be a valid rating from “A” to “E”.
* A stationary air conditioner’s power usage must be a positive (non-zero) integer.
* A car or a plane air conditioner’s volume covered must be a positive (non-zero) integer.
* A plane air conditioner’s electricity used must be a positive (non-zero) integer.

In case the validation fails, the system throws an **ArgumentException** with an appropriate message.

The **controller** contains the main business logic of the application. It contains a **database** and all the **actions**. An **action** is a public method which either returns a **string result** or throws an **exception** and can optionally accept parameters. The **database** class stores all the relevant information needed for the operation of the air conditioner tester system.

In case of a **correct command the controller** returns either a success message (if everything went as expected), or throws an exception (if there was any problem executing the command).

The **controller** checks the validity of the current action. For example, if a command for testing an air conditioner model which is not registered is received, the system will reject the request. If the validation check fails, the system throws a **NonExistantEntryException** with the message **"The specified entry does not exist."**.

## System functionality

The air conditioner tester system contains the following commands:

* **RegisterStationaryAirConditioner (<manufacturer>,<model>,<energyEfficiencyRating>,<powerUsage>)**Registers a new air conditioner with the specified manufacturer, model, required energy efficiency rating and power usage.

|  |  |  |
| --- | --- | --- |
| **Case** | **Message** | **Exception** |
| Success | Air Conditioner model [model] from [manufacturer] registered successfully. | None |
| The required energy efficiency rating is not between the required values. | Energy efficiency rating must be between "A" and "E". | ArgumentException |
| The length of the manufacturer name is too short. | Manufacturer's name must be at least [min manufacturer’s length] symbols long. | ArgumentException |
| The length of the model name is too short. | Model's name must be at least [min model's length] symbols long. | ArgumentException |
| The power usage is zero or a negative number. | Power Usage must be a positive integer. | ArgumentException |
| The air conditioner is already registered. | An entry for the given model already exists. | DuplicateEntryException |

An example output for registering a stationary air conditioner with manufacturer “Toshiba”, model “EX100”, required energy efficiency rating “B” and power usage 1000 KW/h is given bellow:  
Air Conditioner model EX100 from Toshiba registered successfully.

* **RegisterCarAirConditioner (<manufacturer>,<model>,<volumeCoverage>)**Registers a new air conditioner with the specified manufacturer, model and volume covered.

|  |  |  |
| --- | --- | --- |
| **Case** | **Message** | **Exception** |
| Success | Air Conditioner model [model] from [manufacturer] registered successfully. | None |
| The length of the manufacturer name is too short. | Manufacturer's name must be at least [min manufacturer’s length] symbols long. | ArgumentException |
| The length of the model name is too short. | Model's name must be at least [min model's length] symbols long. | ArgumentException |
| The volume covered is zero or a negative number. | Volume Covered must be a positive integer. | ArgumentException |
| The air conditioner is already registered. | An entry for the given model already exists. | DuplicateEntryException |

An example output for registering an air conditioner with manufacturer “Toshiba”, model “C60” and Volume Covered - 9 is given bellow:   
Air Conditioner model C60 from Toshiba registered successfully.

* **RegisterPlaneAirConditioner (<manufacturer>,<model>,<volumeCoverage>, <electricityUsed>)**Registers a new air conditioner with the specified manufacturer, model, volume covered and electricity used.

|  |  |  |
| --- | --- | --- |
| **Case** | **Message** | **Exception** |
| Success | Air Conditioner model [model] from [manufacturer] registered successfully. | None |
| The length of the manufacturer name is too short. | Manufacturer's name must be at least [min manufacturer's length] symbols long. | ArgumentException |
| The length of the model name is too short. | Model's name must be at least [min model's length] symbols long. | ArgumentException |
| The volume covered is zero or a negative number. | Volume Covered must be a positive integer. | ArgumentException |
| The electricity used is zero or a negative number. | Electricity Used must be a positive integer. | ArgumentException |
| The air conditioner is already registered. | An entry for the given model already exists. | DuplicateEntryException |

An example output for registering an air conditioner with manufacturer “Hitachi”, model “P320”, Volume Covered - 25 and Electricity Used - 750 is given bellow:   
Air Conditioner model P320 from Hitachi registered successfully.

* **TestAirConditioner (<manufacturer>,<model>)**Tests an air conditioner with the specified manufacturer and model and produces a Report containg manufacturer, model and mark.

|  |  |  |
| --- | --- | --- |
| **Case** | **Message** | **Exception** |
| Success | Air Conditioner model [model] from [manufacturer] tested successfully. | None |
| There is no registered air conditioner with the specified manufacturer or model. | The specified entry does not exist. | NonExistantEntryException |
| A Report for the specified model already exists. | An entry for the given model already exists. | DuplicateEntryException |

An example output for testing an air conditioner with manufacturer “Toshiba”, model “EX100”, required energy efficiency rating “B” and power usage 1000 KW/h is given bellow:  
Air Conditioner model EX100 from Toshiba tested successfully.

* **FindAirConditioner (<manufacturer>,<model>)**Finds an air conditioner by a given manufacturer and model (check the examples bellow the table to understand how to print each type of air conditioner.

|  |  |  |
| --- | --- | --- |
| **Case** | **Message** | **Exception** |
| Success | Air Conditioner ====================  Manufacturer: [manufacturer] Model: [model]  **<for stationary AC’s only>** Required energy efficiency rating: [required energy efficiency rating] Power Usage(KW / h): [power usage]  **</stationary>**  **<for car AC’s only>**  Volume Covered: [volume covered]  **</car>**  **< for plane AC’s only>**  Volume Covered: [volume covered]  Electricity Used: [electricity used]  **</plane>** ==================== | None |
| There is no registered air conditioner with the specified manufacturer or model. | The specified entry does not exist. | NonExistantEntryException |

An example output for a stationary air conditioner with manufacturer “Toshiba”, model “EX100”, required energy efficiency rating “B” and power usage 1000 KW/h is given bellow:  
Air Conditioner  
====================

Manufacturer: Toshiba  
Model: EX100  
Required energy efficiency rating: B  
Power Usage(KW / h): 1000  
====================

An example of a car air conditioner with manufacturer “Toshiba”, model “C60”, Volume Covered – 9 would be:

Air Conditioner  
====================

Manufacturer: Toshiba  
Model: C60  
Volume Covered: 9  
====================

An example of a plane air conditioner with manufacturer “Hitachi”, model “P320”, Volume Covered - 25 and Electricity Used - 750 would be:

Air Conditioner  
====================

Manufacturer: Hitachi  
Model: P320  
Volume Covered: 25  
Electricity Used: 750  
====================

* **FindReport (<manufacturer>,<model>)**Finds a report by a given manufacturer and model.

|  |  |  |
| --- | --- | --- |
| **Case** | **Message** | **Exception** |
| Success | Report ====================  Manufacturer: [manufacturer] Model: [model] Mark: [mark] ==================== | None |
| There is no Report with the given manufacturer or model. | The specified entry does not exist. | NonExistantEntryException |

An example output for a Report with manufacturer “Toshiba”, model “EX100” and a mark “Passed” is given bellow:  
Report  
====================

Manufacturer: Toshiba  
Model: EX100  
Mark: Passed  
====================

An example output for a Report with manufacturer “Hitachi”, model “P320” and a mark “Failed” is given bellow:  
Report  
====================

Manufacturer: Hitachi  
Model: P320  
Mark: Failed  
====================

* **FindAllReportsByManufacturer (<manufacturer>)**Finds all Reports for a given manufacturer.

|  |  |  |
| --- | --- | --- |
| **Case** | **Message** | **Exception** |
| Success | **In case there are no reports:**  No reports.  **In case there are reports, displays them sorted by model in ascending order in the format:**  Reports from [Manufacturer]:  Report ====================  Manufacturer: [manufacturer] Model: [model] Mark: [mark] ==================== | None |

An example output for the command in a system which has 2 Reports from manufacturer “Toshiba” is given bellow:

Reports from Toshiba:

Report  
====================  
Manufacturer: Toshiba  
Model: C60  
Mark: Passed  
====================  
Report  
====================  
Manufacturer: Toshiba  
Model: EX100  
Mark: Passed  
====================  
**Status ()**Prints a message displaying the status of the system (percentage of air conditioners tested). Note that the percentage should be **rounded to exactly two decimal places**.

|  |  |  |
| --- | --- | --- |
| **Case** | **Message** | **Exception** |
| Success | Jobs complete: <percent of registered air conditioners tested> | None |

An example output for a system that has 3 registered air conditioners of which 2 have been tested would be:   
Jobs complete: 66.67%

Model the system and all entities using the best established practices in object-oriented design and object-oriented programming.

The input should be read from the console. The output is written to the console. The input and output formats have been specified above.

## Sample Input 1

|  |
| --- |
| RegisterStationaryAirConditioner (Toshiba,EX1000,B,1000)  RegisterPlaneAirConditioner (Hitachi,P320,25,750)  RegisterCarAirConditioner (Toshiba,C60,9)  TestAirConditioner (Toshiba,EX1000)  TestAirConditioner (Hitachi,P320)  FindReport (Toshiba,EX1000)  Status ()  RegisterStationaryAirConditioner (Toshiba,WH70,A,780)  TestAirConditioner (Toshiba,WH70)  FindAllReportsByManufacturer (Toshiba) |

## Sample Output 1

|  |
| --- |
| Air Conditioner model EX1000 from Toshiba registered successfully.  Air Conditioner model P320 from Hitachi registered successfully.  Air Conditioner model C60 from Toshiba registered successfully.  Air Conditioner model EX1000 from Toshiba tested successfully.  Air Conditioner model P320 from Hitachi tested successfully.  Report  ====================  Manufacturer: Toshiba  Model: EX1000  Mark: Passed  ====================  Jobs complete: 66.67%  Air Conditioner model WH70 from Toshiba registered successfully.  Air Conditioner model WH70 from Toshiba tested successfully.  Reports from Toshiba:  Report  ====================  Manufacturer: Toshiba  Model: EX1000  Mark: Passed  ====================  Report  ====================  Manufacturer: Toshiba  Model: WH70  Mark: Passed  ==================== |

## Sample Input 2

|  |
| --- |
| TestAirConditioner (Toshiba,EX1000)  Register (Hitachi,C50,1000,5000)  RegisterCarAirConditioner (Tos,WX500,8)  RegisterCarAirConditioner (Toshiba,WX500,8)  RegisterPlaneAirConditioner (Toshiba,WX500,8,1000)  RegisterAirConditioner (Toshiba,WX500,8)  RegisterPlaneAC (Hitachi,HZ200,25,700)  RegisterPlaneAirConditioner (Hitachi,HZ200,-20,500)  RegisterPlaneAirConditioner (Hitachi,HZ200,25,1000)  TestAirConditioner (Toshiba, WX500)  FindAirConditioner (Toshiba,WX500)  RegisterCarAirConditioner (Toshiba,WX500,8)  RegisterPlaneAirConditioner (Hitachi,HZ220,20,-500)  TestAirConditioner (Hitachi,HZ200)  TestAirConditioner (Hitachi,HZ200)  RegisterCarAirConditioner (Toshiba,WX500,8)  FindReport (Hitachi,HZ200)  FindReport (Toshiba,WX500)  RegisterStationaryAirConditioner (Fujitsu,FU90,C,1473,15)  Status () |

## Sample Output 2

|  |
| --- |
| The specified entry does not exist.  Invalid command  Manufacturer's name must be at least 4 symbols long.  Air Conditioner model WX500 from Toshiba registered successfully.  An entry for the given model already exists.  Invalid command  Invalid command  Volume Covered must be a positive integer.  Air Conditioner model HZ200 from Hitachi registered successfully.  The specified entry does not exist.  Air Conditioner  ====================  Manufacturer: Toshiba  Model: WX500  Volume Covered: 8  ====================  An entry for the given model already exists.  Electricity Used must be a positive integer.  Air Conditioner model HZ200 from Hitachi tested successfully.  An entry for the given model already exists.  An entry for the given model already exists.  Report  ====================  Manufacturer: Hitachi  Model: HZ200  Mark: Failed  ====================  The specified entry does not exist.  Invalid command  Jobs complete: 50.00% |

## Problem 1. Code Refactoring

**Refactor the source code** to improve its quality following the best practices introduced in the course  
“[High-Quality Code](https://softuni.bg/courses/high-quality-code/)”. You may refactor anything, as long as it improves the code quality. You may create as many classes, interfaces, enumerations, structures, etc. as you wish.

**25 score**

## Problem 2. StyleCop

Make StyleCop run without any errors on your code (ignore all documentation-related errors). Use the following StyleCop settings:



**4 score**

## Problem 3. Bug Fixing

**Debug the code** and fix any bugs you find.

**4 score**

## Problem 4. Code Documentation

**Document the following interfaces and methods** using C# XML documentation:

* The interface for the **Report** model (declaration + members)
* The controller method implementing the **RegisterPlaneAirConditioner** action
* The controller method implementing the **FindAirConditioner** action
* The controller method implementing the **Status** action

Any other documentation is **not** required. Each documentation gives 1 score.

**7 score**

## Problem 5. Unit Testing

Design and implement **unit tests for** **the following methods:**

* The method implementing the **RegisterStationaryAirConditioner** action.
* The method implementing the **FindAllReportsByManufacturer** action.
* The method implementing the **Status** action.

**Hint:** In Windows a new line is actually “\r\n”

Any other code is not required to be tested. The **code coverage** should be **at** **least 80% for the specified methods** (you do not need to cover the class that parses the input commands and prints the output). Be sure to test **all major execution scenarios** + all interesting **border cases** and **special cases**. Use Visual Studio Team Test (VSTT) and VS code coverage.

**30 score**

## Problem 6. Performance Bottlenecks

Find any **performance bottlenecks** and briefly describe them with the following **comment in the code**:

**// PERFORMANCE: <your description of why you think this is a performance bottleneck>**

**Fix the problems** if possible (and leave the bottleneck descriptions in addition to the fixes).

**4 score**

## Problem 7. Correct Results in the Judge System

You are given an automated judge system to submit your solution. If your code is correct (all bugs are fixed) and runs fast enough (the performance bottlenecks are fixed), your solution will pass all the tests. The last 2 tests measure performance. The others measure correctness.

**16 score**

## Problem 8. Mocking

Test the controller action **TestAirConditioner** using mocking. You are free to do it manually or use a framework (like Moq for example).

Note that you may need to make your code testable first.

**10 score**

## Problem 9\*. Dependency Injection

Try to decouple the implementation of the command execution from the controller using dependency injection.

**10 score (bonus)**