# Object-Oriented Programming – Practical Exam

## Problem 1 – Furniture

A furniture manufacturer keeps track of their **companies** and **furniture**: **tables** and **chairs**. Each furniture piece has **model**, **material**, **price** in dollars, and **height** in meters. Each table has **length** and **width** in meters. Chairs are three types: **normal**, **adjustable** and **convertible**. Each chair has **number of legs**. Each adjustable chair can **adjust** its height. Each convertible chair can **convert** its state and be easily movable. Each company has **name**, **registration number** and **catalog of furniture**. Companies can **add** or **remove** furniture to their catalogs. Companies can **find** furniture by model. Companies can **show catalogs** of all furniture they offer.

### Design the Class Hierarchy

Your **task** is to **design an object-oriented class hierarchy** to model the furniture manufacturer, companies and all types of furniture **using the best practices for object-oriented design (OOD) and object-oriented programming (OOP)**. **Avoid duplicated code though abstraction, inheritance, and polymorphism and encapsulate correctly all fields.**

You are given few C# **interfaces** that you should **obligatory** implement and use as a basis of your code:

|  |
| --- |
| namespace FurnitureManufacturer.Interfaces  {  publicinterfaceICompany  {  string Name { get; }  string RegistrationNumber { get; }  ICollection<IFurniture> Furnitures { get; }  void Add(IFurniture furniture);  void Remove(IFurniture furniture);  IFurniture Find(string model);  string Catalog();  }  publicinterfaceIFurniture  {  string Model { get; }  string Material { get; }  decimal Price { get; set; }  decimal Height { get; }  }  publicinterfaceIChair : IFurniture  {  int NumberOfLegs { get; }  }  publicinterfaceITable : IFurniture  {  decimal Length { get; }  decimal Width { get; }  decimal Area { get; }  }  publicinterfaceIAdjustableChair : IChair  {  void SetHeight(decimal height);  }  publicinterfaceIConvertibleChair : IChair  {  bool IsConverted { get; }  void Convert();  }  } |

All your furniture should implement IFurniture. Tables should implement ITable, chairs should implement IChair, adjustable chairs should implement IAdjustableChair and convertible chairs should implement IConvertibleChair. Companies should implement ICompany.

**Furniture** validity rules:

* Model cannot be empty, null or with less than 3 symbols.
* Price cannot be less or equal to $0.00.
* Height cannot be less or equal to 0.00m.

**Table** validity rules:

* Can calculate area by the following formula: length \* width.

**Adjustable chair** validity rules:

* Can change the height to a new valid one.

**Convertible chair** validity rules:

* Has two states – converted and normal.
* States can be changed by converting the chair from one to another.
* Converted state sets the height to 0.10m.
* Normal state returns the height to the initial one.
* Initial state is normal.

**Company** validity rules:

* Name cannot be empty, null or with less than 5 symbols.
* Registration number must be exactly 10 symbols and must contain only digits.
* Adding duplicate furniture is allowed.
* Removing furniture removes the first occurance. If such is not found, nothing happens.
* Finding furniture by model gets the first occurance. If such is not found, return null. Searching is case insensitive.

Companies should only be created through the ICompanyFactory implemented by a class named **CompanyFactory**. Furniture should only be created through the IFurnitureFactory implemented by a class named **FurnitureFactory**. Both classes are in the **FurnitureManufacturer.Engine.Factories** namespace.

The company catalog method returns the information about the available furniture in the following form:

|  |
| --- |
| ***(company name)*–*(number of furniture/”no”)(“furniture”/”furnitures”)***  (*information about furniture)*  (*information about furniture)*  (*information about furniture)* |

The listed furniture added to a certain company (through the **Add(…)** method) should be ordered by price then by model. If the company has no furniture added, print **“no furnitures”** (yes, we know “furnitures” is not a valid word, but we do not care, obey the requirements :D ). If the company has 1 piece of furniture, print **“1 furniture”** and show its information on a separate line. If the company has more than 1 piece of furniture, print its number and list each one’s information on a separate line. All decimal type fields should be printed “as is”, without any formatting or rounding.

You may use the following for reference:

|  |
| --- |
| "{0} - {1} - {2} {3}",  this.Name,  this.RegistrationNumber,  this.Furnitures.Count != 0 ? this.Furnitures.Count.ToString() : "no",  this.Furnitures.Count != 1 ? "furnitures" : "furniture" |

Look into the example below to get better understanding of the printing format.

The table information should be in the following form:

|  |
| --- |
| "Type: {0}, Model: {1}, Material: {2}, Price: {3}, Height: {4}, Length: {5}, Width: {6}, Area: {7}", this.GetType().Name, this.Model, this.Material, this.Price, this.Height, this.Length, this.Width, this.Area |

The normal and adjustable chair information should be in the following form:

|  |
| --- |
| "Type: {0}, Model: {1}, Material: {2}, Price: {3}, Height: {4}, Legs: {5}", this.GetType().Name, this.Model, this.Material, this.Price, this.Height, this.NumberOfLegs |

The convertible chair information should be in the following form:

|  |
| --- |
| "Type: {0}, Model: {1}, Material: {2}, Price: {3}, Height: {4}, Legs: {5}, State: {6}", this.GetType().Name, this.Model, this.Material, this.Price, this.Height, this.NumberOfLegs, this.IsConverted ? "Converted" : "Normal" |

The Type is either “Table“, or “Chair”, or “**Adjustable**Chair” or “**ConvertibleChair**”. The convertible chair state is either “**Converted**” or “**Normal**”. All decimal type fields should be printed “**as is**”, without any formatting or rounding.

All properties in the above interfaces are mandatory (cannot be null or empty).

If a null value is passed to some mandatory property, you should use **defensive programming** to prevent unwanted results.

### Additional Notes

To simplify your work you are given an engine that executes a sequence of commands read from the console using the classes and interfaces in your project. Please put your classes in namespace **FurnitureManufacturer.Models**.Implement the**CompanyFactory**and **FurnitureFactory** class in the namespace **FurnitureManufacturer.Engine.Factories**.

You are only **allowed to write classes in theFurnitureManufacturer.Modelsnamespace**. You are **not allowed to modify the existing interfaces and classes except the CompanyFactoryand FurnitureFactoryclasses**. You may delete the **DeleteMe.cs** file. ☺

Currentimplemented commands the engine supports are:

* **CreateCompany (name) (registration number)**– adds a company with given name and registration number. Duplicate names are not allowed. As a result the command returns “**Company (name) created**”.
* **AddFurnitureToCompany (company name) (furniture model)**– searches for furniture and adds it to an existing company’s catalog. As a result the command returns “**Furniture (furniture model) added to company (company name)**”.
* **RemoveFurnitureFromCompany (company name) (furniture model)** – searches for furniture and removes it from an existing company’s catalog.As a result the command returns “**Furniture (furniture model) removed fromcompany (company name)**”.
* **FindFurnitureFromCompany (company name) (furniture model)** – searches for furniture in an existing company’s catalog.If found the engine prints the furniture’s ToString() method.
* **ShowCompanyCatalog (company name)** – searches for a company and invokes it’s Catalog() method.
* **CreateTable (model) (material) (price) (height) (length) (width)**– creates a table with given model, material, price, height, length and width. Duplicate models are not allowed. As a result the command returns “**Table (model) created**”.
* **CreateChair(model) (material) (price) (height) (legs) (type)** – creates a chair by given model, material, price, height, legs and type. Type can be “**Normal**”, “**Adjustable**” and “**Convertible**”.Duplicate models are not allowed.As a result the command returns “**Chair(model) created**”.
* **SetChairHeight(model) (height)** – searches for a chair by model and sets its height, if the chair is adjustable.As a result the command returns “**Chair (model) adjusted to height (height)**”.
* **ConvertChair (model)** – searches for a chair by model and converts its state, if the chair is convertible. As a result the command returns “**Chair (model) converted**”.

In case of invalid operation or error, the engine returns appropriate text messages.

### Sample Input

|  |
| --- |
| **CreateCompany AcademyDivani 1234567890**  **CreateCompany AcademyHladilnici 0987654321**  **ShowCompanyCatalog AcademyDivani**  **CreateTable JustMasa wooden 123.4 0.50 0.45 0.65**  **CreateChair KendoStol leather 99.99 1.20 5 Normal**  **CreateChair SitefinityDivan leather 111.56 0.80 4 Adjustable**  **CreateChair AJAXControlsTaburetka plastic 80.00 1.00 3 Convertible**  **CreateChair SitefinityShtyrkel leather 111.56 0.80 4 Normal**  **ShowCompanyCatalog AcademyHladilnici**  **AddFurnitureToCompany AcademyHladilnici JustMasa**  **AddFurnitureToCompany AcademyHladilnici SitefinityShtyrkel**  **AddFurnitureToCompany AcademyHladilnici JustMasa**  **AddFurnitureToCompany AcademyHladilnici SitefinityDivan**  **ShowCompanyCatalog AcademyHladilnici**  **ShowCompanyCatalog AcademyDivani**  **AddFurnitureToCompany AcademyDivani JustMasa**  **AddFurnitureToCompany AcademyDivani KendoStol**  **AddFurnitureToCompany AcademyDivani AJAXControlsTaburetka**  **AddFurnitureToCompany AcademyDivani SitefinityDivan**  **ShowCompanyCatalog AcademyDivani**  **ShowCompanyCatalog AcademyHladilnici**  **RemoveFurnitureFromCompany AcademyHladilnici JustMasa**  **ShowCompanyCatalog AcademyHladilnici**  **FindFurnitureFromCompany AcademyHladilnici JustMasa**  **FindFurnitureFromCompany AcademyHladilnici SitefinityDivan**  **RemoveFurnitureFromCompany AcademyDivani SitefinityDivan**  **RemoveFurnitureFromCompany AcademyDivani SitefinityDivan**  **ShowCompanyCatalog AcademyDivani**  **FindFurnitureFromCompany AcademyDivani SitefinityDivan**  **FindFurnitureFromCompany AcademyDivani AJAXControlsTaburetka**  **FindFurnitureFromCompany AcademyDivani KendoStol**  **CreateCompany KenovAndSonBiura 6666666666**  **CreateChair PeshoBiuro plastic 0.99 0.67 4 Adjustable**  **AddFurnitureToCompany KenovAndSonBiura PeshoBiuro**  **SetChairHeight PeshoBiuro 1.11**  **FindFurnitureFromCompany KenovAndSonBiura PeshoBiuro**  **CreateChair GoshoFotiol wooden 1.99 0.95 1 Convertible**  **AddFurnitureToCompany KenovAndSonBiura GoshoFotiol**  **ConvertChair GoshoFotiol**  **FindFurnitureFromCompany KenovAndSonBiura GoshoFotiol**  **ConvertChair GoshoFotiol**  **FindFurnitureFromCompany KenovAndSonBiura GoshoFotiol**  **ConvertChair GoshoFotiol**  **ShowCompanyCatalog KenovAndSonBiura** |

### Sample Output

|  |
| --- |
| **Company AcademyDivani created**  **Company AcademyHladilnici created**  **AcademyDivani - 1234567890 - no furnitures**  **Table JustMasa created**  **Chair KendoStol created**  **Chair SitefinityDivan created**  **Chair AJAXControlsTaburetka created**  **Chair SitefinityShtyrkel created**  **AcademyHladilnici - 0987654321 - no furnitures**  **Furniture JustMasa added to company AcademyHladilnici**  **Furniture SitefinityShtyrkel added to company AcademyHladilnici**  **Furniture JustMasa added to company AcademyHladilnici**  **Furniture SitefinityDivan added to company AcademyHladilnici**  **AcademyHladilnici - 0987654321 - 4 furnitures**  **Type: AdjustableChair, Model: SitefinityDivan, Material: Leather, Price: 111.56, Height: 0.80, Legs: 4**  **Type: Chair, Model: SitefinityShtyrkel, Material: Leather, Price: 111.56, Height: 0.80, Legs: 4**  **Type: Table, Model: JustMasa, Material: Wooden, Price: 123.4, Height: 0.50, Length: 0.45, Width: 0.65, Area: 0.2925**  **Type: Table, Model: JustMasa, Material: Wooden, Price: 123.4, Height: 0.50, Length: 0.45, Width: 0.65, Area: 0.2925**  **AcademyDivani - 1234567890 - no furnitures**  **Furniture JustMasa added to company AcademyDivani**  **Furniture KendoStol added to company AcademyDivani**  **Furniture AJAXControlsTaburetka added to company AcademyDivani**  **Furniture SitefinityDivan added to company AcademyDivani**  **AcademyDivani - 1234567890 - 4 furnitures**  **Type: ConvertibleChair, Model: AJAXControlsTaburetka, Material: Plastic, Price: 80.00, Height: 1.00, Legs: 3, State: Normal**  **Type: Chair, Model: KendoStol, Material: Leather, Price: 99.99, Height: 1.20, Legs: 5**  **Type: AdjustableChair, Model: SitefinityDivan, Material: Leather, Price: 111.56, Height: 0.80, Legs: 4**  **Type: Table, Model: JustMasa, Material: Wooden, Price: 123.4, Height: 0.50, Length: 0.45, Width: 0.65, Area: 0.2925**  **AcademyHladilnici - 0987654321 - 4 furnitures**  **Type: AdjustableChair, Model: SitefinityDivan, Material: Leather, Price: 111.56, Height: 0.80, Legs: 4**  **Type: Chair, Model: SitefinityShtyrkel, Material: Leather, Price: 111.56, Height: 0.80, Legs: 4**  **Type: Table, Model: JustMasa, Material: Wooden, Price: 123.4, Height: 0.50, Length: 0.45, Width: 0.65, Area: 0.2925**  **Type: Table, Model: JustMasa, Material: Wooden, Price: 123.4, Height: 0.50, Length: 0.45, Width: 0.65, Area: 0.2925**  **Furniture JustMasa removed from company AcademyHladilnici**  **AcademyHladilnici - 0987654321 - 3 furnitures**  **Type: AdjustableChair, Model: SitefinityDivan, Material: Leather, Price: 111.56, Height: 0.80, Legs: 4**  **Type: Chair, Model: SitefinityShtyrkel, Material: Leather, Price: 111.56, Height: 0.80, Legs: 4**  **Type: Table, Model: JustMasa, Material: Wooden, Price: 123.4, Height: 0.50, Length: 0.45, Width: 0.65, Area: 0.2925**  **Type: Table, Model: JustMasa, Material: Wooden, Price: 123.4, Height: 0.50, Length: 0.45, Width: 0.65, Area: 0.2925**  **Type: AdjustableChair, Model: SitefinityDivan, Material: Leather, Price: 111.56, Height: 0.80, Legs: 4**  **Furniture SitefinityDivan removed from company AcademyDivani**  **Furniture SitefinityDivan removed from company AcademyDivani**  **AcademyDivani - 1234567890 - 3 furnitures**  **Type: ConvertibleChair, Model: AJAXControlsTaburetka, Material: Plastic, Price: 80.00, Height: 1.00, Legs: 3, State: Normal**  **Type: Chair, Model: KendoStol, Material: Leather, Price: 99.99, Height: 1.20, Legs: 5**  **Type: Table, Model: JustMasa, Material: Wooden, Price: 123.4, Height: 0.50, Length: 0.45, Width: 0.65, Area: 0.2925**  **Furniture SitefinityDivan not found**  **Type: ConvertibleChair, Model: AJAXControlsTaburetka, Material: Plastic, Price: 80.00, Height: 1.00, Legs: 3, State: Normal**  **Type: Chair, Model: KendoStol, Material: Leather, Price: 99.99, Height: 1.20, Legs: 5**  **Company KenovAndSonBiura created**  **Chair PeshoBiuro created**  **Furniture PeshoBiuro added to company KenovAndSonBiura**  **Chair PeshoBiuro adjusted to height 1.11**  **Type: AdjustableChair, Model: PeshoBiuro, Material: Plastic, Price: 0.99, Height: 1.11, Legs: 4**  **Chair GoshoFotiol created**  **Furniture GoshoFotiol added to company KenovAndSonBiura**  **Chair GoshoFotiol converted**  **Type: ConvertibleChair, Model: GoshoFotiol, Material: Wooden, Price: 1.99, Height: 0.10, Legs: 1, State: Converted**  **Chair GoshoFotiol converted**  **Type: ConvertibleChair, Model: GoshoFotiol, Material: Wooden, Price: 1.99, Height: 0.95, Legs: 1, State: Normal**  **Chair GoshoFotiol converted**  **KenovAndSonBiura - 6666666666 - 2 furnitures**  **Type: AdjustableChair, Model: PeshoBiuro, Material: Plastic, Price: 0.99, Height: 1.11, Legs: 4**  **Type: ConvertibleChair, Model: GoshoFotiol, Material: Wooden, Price: 1.99, Height: 0.10, Legs: 1, State: Converted** |

## Problem 2 – Infestation

The Zemyans (“Zemya” coming from the Bulgarian word, meaning the same as “Terra” in latin) have captured some renegade Grez. The Grez are a slimy alien race, which increase their ranks through a process called infestation. Now the Zemyans want to simulate the Grez interactions with their own units, to have a better fighting chance when the invasion comes.

You are given an API, which supports some basic unit interactions, based on early Zemyan designs, before the Grez showed up. You need to extend the API to match the current situation.

There are some simple rules the API supports:

* There are three main components of the API: the HoldingPen, the Unit and the Supplement
  + A HoldingPen contains units in an isolated environment and executes basic interactions between them. It can also add new units, provide them with supplements and report the their status
  + A Unit represents any active thing in the game – e.g. a Person, a Dog, a Tank, etc.
    - Units can interract with each other and their interactions depend on several characteristics (discussed later)
  + A Supplement is an object, which quantatively changes the basic characteristics of an object. E.g. a Weapon is a kind of a Supplement, which makes a trained user more dangerous
* Units
  + Every unit has an Id (a string name) which uniquely identifies it
  + Every unit has base Health, Power and Aggresiveness
    - Base meaning the value before adding supplements, which can increase the aforementioned
  + The more health a unit has, the more attacks it can survive
  + The more power a unit has, the more potent its attacks are
  + The more aggressive a unit is, the more likely it is to attack another unit
  + Every unit has a set of active supplements (usually empty when first created)
  + Every unit has a classification (type) either Biological, Mechanical or Psionic (i.e. telepathic)
* Supplements
  + Every Supplement can give a bonus (positive or negative) to a units Health, Power and Aggresiveness
  + The API currently has NO implemented supplements, but has infrastructure which can be extended to include supplements
* The HoldingPen is the place where all operations (commands from the console) are parsed and executed

### Commands

There are four commands the HoldingPen supports:

* Insertion command – createsUnits
  + Syntax: insertDog *dogName*
  + Inserting (creating) a Unit requires its type and its id
  + Syntax: insertDog*sharo*– creates a Dog with the id “sharo”
* Proceed command – forces all units to engage in interactions
  + Syntax: proceed
  + During a proceed command, each unit receives information about all the other units in the HoldingPen
  + During a proceed command, each unit can interact exactly once
    - The unit gives information to the HoldingPen about the interaction it wishes to execute and the HoldingPen takes care of the actual interaction
  + Interactions are three types – attack, infest and the passive interaction (i.e. the unit does nothing)
* Supplement command – adds a supplement to an existing unit
  + Syntax: supplement SupplementType *targetUnitId*
  + Creates a supplement of the desired type (e.g. AggressionInhibitor) and adds it to the unit with the provided id
  + Syntax: supplement AggressionInhibitor *sharo–* adds an AggressionInhibitor to the Dog sharo (created in the previous examples)
* Status command – prints information about all units in the HoldingPen
  + Syntax: status
  + The base class Unit overrides the ToString() method appropriately, describing its state. You need not concern yourself with this operation
  + Note: The ToString() command prints the object class name and its supplements’ class names – be sure when you are creating units and supplements to use the names exactly as they are described below

### Tasks

You are tasked with extending the API by implementing several commands and object types. You are **not allowed to edit any existing class from the original code of the API**. You **areNOTallowed to edit the Main method**.You are **only allowed to edit the InitializePen()** method in the **Program** class.

* Catalists are Supplements, which improve the Health, Power and Aggression of a unit. Implement:
  + A PowerCatalyst – has a PowerEffect of 3
  + A HealthCatalyst – has a HealthEffect of 3
  + An AggressionCatalyst– has an AggressionEffect of 3
* Implement a Tank
  + A Tank is a type of Unit
  + The Tank has a base Power of 25, a base Health of 20, and a base Aggression of 25
  + The Tank is classified as a Mechanical Unit.
* Implement a Marine
  + The Marine is a type of Human
  + It has the same base Power, Health, Aggression
  + It has a supplement by default – WeaponrySkill
    - The WeaponrySkill does not directly affect any of the properties of the Marine
    - The WeaponrySkill cannot be added with the supplement command. This means you cannot create it through the console, it doesn’t mean you can’t use the AddSupplement method.
  + When a Marine attacks, it always picks a target, such that:
    - The target’s Power is less than or equal to the Marine’sAggression
    - If there is more than one such target, the marine picks the one with the highest Health
* Implement a Weapon Supplement
  + A Weapon is a Supplement, which increases the Power of a Unit by 10 and its Aggression by 3, but only if the Unit already has a WeaponrySkill Supplement. If not, the Weapon Supplement does not have any effect.
* Implement an InfestationSpores Supplement
  + The InfestationSpores Supplement has an AggressionEffect of 20 and a PowerEffect of -1
  + The InfestationSpores Supplement does not accumulate like the other Supplements – even if two or more Infestations are added, the total AggressionEffect stays 20 (the same goesfor PowerEffect).
  + The InfestationSpores Supplement cannot be added with the supplement command
* Implement a Parasite
  + The Parasite is a type of Unit, which can Infest
    - Infesting is equivalent to adding an InfestationSpores Supplement to the target
  + The Parasite is classified as a Biological Unit.
  + The Parasite has all base values set to 1
  + When a Parasite is offered to Interact, it always tries to find a Unit to infest
    - The target Unit can be any unit different than itself
    - If there are multiple such units, the Parasite picks the one with the least Health
* Implement a Queen
  + The Queen is a type of Unit, which can infest
    - Infesting is equivalent to adding an InfestationSpores Supplement to the target
  + The Queen has a base Health of 30 and all of its other base values are set to 1
  + The Queen is classified as a Psionic Unit
  + The Queen interacts in the same way as the Parasite
* **Infesting has some requirements**
  + A Biological unit can only be infested by another Biological unit
  + A Mechanical unit can only be infested by a Psionic unit
  + A Psionic unit can only be infested by another Psionic unit
  + There is some code in the API reflecting these rules, seek it out

### Input and Output Data

You should not concern yourself with handling input and output data – the engine does it for you. You should only consider how to implement the required commands. See the existing API code for hints. Also:

* The names in the commands will always consist of upper and lowercase English letters only.
* If for some reason a command is illegal, just skip it
* The ReactTo method needs to be implemented specifically for ONLY two of the above tasks – most supplements don’t need to react (at least in this task)

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| **insert Dog Sharo**  **insert Parasite Paro**  **proceed**  **status**  **insert Tank Tanio**  **proceed**  **status**  **insert Queen Murphy**  **proceed**  **status**  **insert Marine Marin**  **supplement Weapon Marin**  **supplement AggressionCatalyst Marin**  **supplement AggressionCatalyst Marin**  **proceed**  **status**  **end** | **Dog Sharo (Biological) [Health: 4, Power: 4, Aggression: 22, Supplements: [InfestationSpores]]**  **Tank Tanio (Mechanical) [Health: 20, Power: 25, Aggression: 25, Supplements: []]**  **Tank Tanio (Mechanical) [Health: 20, Power: 24, Aggression: 45, Supplements: [InfestationSpores]]**  **Queen Murphy (Psionic) [Health: 5, Power: 1, Aggression: 1, Supplements: []]**  **Tank Tanio (Mechanical) [Health: 20, Power: 24, Aggression: 45, Supplements: [InfestationSpores, InfestationSpores]]**  **Marine Marin (Biological) [Health: 10, Power: 14, Aggression: 10, Supplements: [WeaponrySkill, Weapon, AggressionCatalyst, AggressionCatalyst]]** |