**StorageMaster\_Задача 1**

**Due date**: Sunday, 14 October 2018, 00:00  
**Requested files**: Product.cs, Storage.cs, Vehicle.cs, AutomatedWarehouse.cs, DistributionCenter.cs, Engine.cs, Gpu.cs, HardDrive.cs, Ram.cs, Semi.cs, SolidStateDrive.cs, Truck.cs, Van.cs, Warehouse.cs, StorageMaster.cs, StartUp.cs ([Download](https://it-kariera.mon.bg/e-learning/mod/vpl/views/downloadrequiredfiles.php?id=1564))  
**Type of work**: Individual work  
**Настройки на оценките**: Максимална оценка: 36  
**Run**: Не. **Evaluate**: Да. **Evaluate just on submission**: Да  
**Automatic grade**: Да.

**ООП Подготовка за изпит – Storage Master**

**Преглед на задачата**

В този изпит трябва да построите проект за управление на складове, който има поддръжка за продукти, хранилище за съхранение на продукти и превозни средства за транспортиране на продукти от едно складово помещение до друго. Проектът се състои от модели и контролер клас, който управлява взаимодействието между складовите помещения (storage), превозните средства (vehicles) и продуктите (products).

**Настройка**

Създайте нов проект с името “StorageMaster”. Проектът трябва да има клас с името StartUp в namespace “StorageMaster”.

**Задача 1: Структура (36 точки)**

**Product**

Продуктът е **базов клас** за всички продукти и **не трябва да може да бъде инстанцииран**.

**Полета**

* Price – **реално число (double)**
  + **Ако се въведе отрицателна цена, хвърлете InvalidOperationException със съобщението “Price cannot be negative!”.**
* Weight – **реално число (double)**

**Конструктор**

Продуктът трябва да приема следните стойности при инициализирането:

double price, double weight

**Класове наследници**

Има няколко конкретни вида продукти:

* Gpu – винаги има **0.7** тежест
* HardDrive – винаги има **1** тежест
* Ram – винаги има **0.1** тежест
* SolidStateDrive – винаги има **0.2** тежест

Всеки тип продукт получава само своята цена при инициализация.

**Vehicle**

Превозното средство е базов клас за всички превозни средства и не трябва да може да бъде инстанцран.

**Полета**

* Capacity – **цяло число**
* Trunk – IReadOnlyCollection от продукти
* IsFull – булев
  + **Връща истина ако сумата от теглото на продуктите е равна или по-голяма от капацитета на превозното средство**
* IsEmpty – **булев**
  + **Връща истина ако превозното средство няма продукти в багажника**

**Конструктор**

Превозното средство трябва да примеа следните стойности при инициализация:

int capacity

**Поведение**

**void LoadProduct(Product product)**

Ако автомобилът вече е пълен, хвърлете InvalidOperationException със съобщението "Vehicle is full!".

Ако тази проверка преминава, продуктът се добавя към багажника на автомобила.

**Product Unload()**

Ако багаж**никът на автомобила е празен,**хвърлете InvalidOperationException със съобщението "No products left in vehicle!".

Ако тази проверка мине, последният продукт в багажника се отстранява от багажника на автомобила и се връща на повикващия.

**Класове наследници**

Има няколко конкретни вида превозни средства:

* Van – винаги има капацитет равен на 2
* Truck – винаги има капацитет равен на 5
* Semi – винаги има капацитет равен на 10

Конкретните класове на превозните средства не приемат никакви параметри при инициализация.

**Storage**

The Storage is a **base class** for any **storage** and it **should not be able to be instantiated**.

The **storage** is a building, which holds **products**. It also has a **garage** of vehicles with a **fixed length**. The **length**is determined by the **garage slots**of the storage.

**Полета**

* Name – **низ**
* Capacity – **цяло число**–максималното тегло на продуктите, с които може да работи устройството за съхранение
* GarageSlots – **цяло число –**броя гаражни слотове, които гаражът има
* IsFull – **булев**
  + Връща истина, ако сумата от теглата на продуктите е равна или по-голяма от капацитета за съхранение
* Garage – IReadOnlyCollection of **vehicles**
  + **Read-only** репрезентация на масив от гаражи.
* Products – IReadOnlyCollection от продукти.
  + **Read-only** репрезентация на **продуктите в склада**.

**Конструктор**

Склада трябва да приема следните стойности при инициализирането:

string name, int capacity, int garageSlots, IEnumerable<Vehicle> vehicles

**Поведение**

**Vehicle GetVehicle(int garageSlot)**

Ако предоставеният номер на гаражното място е равен или по-голям от броя на гаражните слотове, хвърлете InvalidOperationException със съобщението "Invalid garage slot!".

Ако гаражът е празен, хвърлете InvalidOperationException със съобщението "No vehicle in this garage slot!"

Методът връща извлеченото превозно средство.

**int SendVehicleTo(int garageSlot, Storage deliveryLocation)**

Вземе превозното средство от определения слот гараж (използвайте валидацията на метода **GetVehicle**).

След това методът проверява дали има свободни гаражни слотове. **Свободният слот е означен с нулева (null) стойност.**

Ако няма свободно място за гараж, хвърлете **InvalidOperationException** със съобщението "No room in garage!".

След това гаражът в хранилището на източника се освобождава и автомобилът се добавя към първия свободен гаражен слот.

Методът връща гаражния слот, в който е добавен автомобилът, когато е прехвърлен.

**int UnloadVehicle(int garageSlot)**

Ако хранилището е пълно, хвърлете **InvalidOperationException** със съобщението "Storage is full!".

Вземете превозното средство от определения гаражен слот(използвайте вадидацията на GetVehicleметода).

След това, докато превозното средство се изпразни или се запълни, продуктите на превозното средство се разопаковат и се добавят към продуктите на хранилището.

Методът връща броя на разтоварените продукти.

**Child Classes**

Има няколко конкретни вида хранилища и всяка от тях има по подразбиране набор от превозни средства:

* AutomatedWarehouse – има капацитет равен на **1** и **2**гаражни слота
  + Превоздни средства по подразбиране: **1 Truck**
  + DistributionCenter – има капацитет равен на **2** и **5**гаражни слота
    - Превоздни средства по подразбиране: **3 Vans**
    - Warehouse – има капацитет равен на **10** and **10**гаражни слота
      * Превоздни средства по подразбиране: **3 Semi trucks**

Всеки тип хранилище получава **име** при **инициализирането**.

**Task 2: Business Logic (42 точки)**

**Контролер клас**

Бизнес логиката на програмата трябва да бъде концентрирана около няколко **команди**. Създайте клас, наречен **StorageMaster**, който ще задържа **основната функционалност**.

**Master Storage** следи регистъра за съхранение и групата продукти (продуктите в основното хранилище). Той също така следи текущото превозно средство (обяснено по-долу).

***Забележка: Класът StorageMaster НЕ трябва да се справя с изключенията! Тестовете са предназначени да очакват изключения, а не съобщения!***

Основната функционалност трябва да бъде постигната чрез **тези публични методи**:

|  |
| --- |
| StorageMaster.cs |
| public string AddProduct(string type, double price)  {    throw new **NotImplementedException**();  }    public string RegisterStorage(string type, string name)  {    throw new **NotImplementedException**();  }    public string SelectVehicle(string storageName, int garageSlot)  {    throw new **NotImplementedException**();  }    public string LoadVehicle(IEnumerable<string> productNames)  {    throw new **NotImplementedException**();  }    public string SendVehicleTo(string sourceName, int sourceGarageSlot, string destinationName)  {    throw new **NotImplementedException**();  }    public string UnloadVehicle(string storageName, int garageSlot)  {    throw new **NotImplementedException**();  }    public string GetStorageStatus(string storageName)  {    throw new **NotImplementedException**();  }    public string GetSummary()  {    throw new **NotImplementedException**();  } |

**Команди**

**Има няколко команди, които контролират бизнес логиката на приложението и трябва да го направите.**

**AddProduct Команд**

**Параметри**

* type – низ
* price – реално число (double)

**Функционалност**

Създава продукт и го добавя в продуктовия пул.

Ако типът на продукта е невалиден, хвърлете **InvalidOperationException** със съобщението "Invalid product type!"

Връша "Added {type} to pool".

**RegisterStorage Команда**

**Параметри**

* type – низ
* name – низ

**Функционалност**

**Създава** хранилище и го **добавя в регистъра за хранилището**.

Ако типът хранилище е **невалиден**, хвърлете **InvalidOperationException** със съобщението "Invalid storage type!"

Връща "Registered {storageName}".

**SelectVehicle Команда**

**Параметри**

* storageName – низ
* garageSlot – цяло число (int)

**Функционалност**

Задава **текущото превозно** средство на превозното средство **в слота за гараж на хранилището**. **Текущото** **превозно** **средство** е превозното средство, с което ще взаимодейства методът **LoadVehicle**.

Връща "Selected {vehicleType}".

**LoadVehicle Команда**

**Параметри**

* productNames – IEnumerable<string>

**Функционалност**

Зарежда текущото превозно средство **с възможно най-много** от предоставените типове продукти, **без да препълва превозното средство**.

Методът преминава през **всяко от имената на продуктите** и изпълнява **следните операции**:

Ако в продуктовия пул **няма елементи с това име**, хвърлете **InvalidOperationException** със съобщение "{name} is out of stock!".

Ако има, **последният продукт с това име** в пула се отстранява от пула и се **зарежда в превозното средство**.

Връща "Loaded {loadedProductsCount}/{productCount} products into {vehicleType}".

**Забележка: productCount е само броят продукти, получени от командата като параметър.**

**SendVehicleTo Команда**

**Параметри**

* sourceName – низ
* garageSlot – цяло число (int)
* destinationName – низ

**Функционалност**

Ако **началното хранилище** **или** **дестинационното хранилище** не съществуват, хвърлете **InvalidOperationException** със съобщението "Invalid source storage!" or "Invalid destination storage!"

След това методът получава превозното средство от хранилището в предоставения гаражен слот и го изпраща на **дестинационното хранилище**.

Връща "Sent {vehicleType} to {destinationName} (slot {destinationGarageSlot})".

**UnloadVehicle Команда**

**Параметри**

* storageName – низ
* garageSlot – цяло число (int)

**Функционалност**

Методът получава превозното средство в гаража на склада. Тогава превозното средство се разтоварва в склад.

Методът връща "Unloaded {unloadedProductsCount}/{productsInVehicle} products at {storageName}".

**GetStorageStatus Команда**

**Параметри**

* storageName – низ

**Функционалност**

Методът взима хранилище по дадено име от регистъра за хранилища и извършва известна агрегация върху него:

Продуктите за съхранение се отчитат, **групират по име**, **сортират се по брой на продуктите** (**низходящи**), след това по име на продукт (**възходящо**).

След това се извлича името на всеки автомобил в гаража. Ако в този гараж няма превозно средство, поставете “empty”.

Командата произвежда два реда:

Първият ред е **стоковият формат**: “Stock ({0}/{1}): [{2}]”. Първият параметър е сумата от теглото на продуктите, вторият параметър е капацитетът на хранилището. Третият параметър е информацията за акциите, описана по-горе, разделена със запетаи.

Вторият ред е гаражният формат: “Garage: [{0}]”. Единственият параметър е имената на превозните средства (и празните гаражни слотове), разделени със знак "|".

Методът връща тези две линии, разделени с **нов ред**.

За пример проверете секцията В/И.

**GetSummary Команда**

**Функционалност**

Методът взима **всички складове** в складовия регистър, **подредени по сумата от цената на продуктите им** (**низходяща**). За всеки един се получава низ в следния формат:

|  |
| --- |
| {storageName}:  Storage worth: ${totalMoney:F2} |

Методът връща всички форматирани низове, разделени с нови линии.

**Task 3: Вход / Изход (22 точки)**

**Вход**

* Ще получите команди, докато не получите "**END**" като команда.

По-долу можете да видите формата, в който всяка команда ще бъде дадена във входа:

* AddProduct {type} {price}
* RegisterStorage {type} {name}
* SelectVehicle {storageName} {garageSlot}
* LoadVehicle {productName1} {productName2} {productNameN}
* SendVehicleTo {sourceName} {sourceGarageSlot} {destinationName}
* UnloadVehicle {storageName} {garageSlot}
* GetStorageStatus {storageName}

**Изход**

**Отпечатайте изхода от всяка команда**, когато бъде издадена. Когато се получи **крайната команда**, отпечатайте изхода от командата **GetSummary**.

Ако InvalidOperationException бъде хвърлен по време на изпълнение на някоя от командите, отпечатайте:

* "Error:" плюс съобщението на изключението
* Командите винаги ще бъдат в предоставения формат.

**Ограничения**

**Примери**

|  |
| --- |
| **Вход** |
| RegisterStorage DistributionCenter SofiaDistribution  RegisterStorage Warehouse AmazonWarehouse  AddProduct Gpu 1200  AddProduct SolidStateDrive 205  AddProduct HardDrive 70  AddProduct HardDrive 120  SelectVehicle SofiaDistribution 0  LoadVehicle HardDrive Gpu  SendVehicleTo SofiaDistribution 0 AmazonWarehouse  UnloadVehicle AmazonWarehouse 3  END |
| **Изход** |
| Registered SofiaDistribution  Registered AmazonWarehouse  Added Gpu to pool  Added SolidStateDrive to pool  Added HardDrive to pool  Added HardDrive to pool  Selected Van  Loaded 2/2 products into Van  Sent Van to AmazonWarehouse (slot 3)  Unloaded 2/2 products at AmazonWarehouse  AmazonWarehouse:  Storage worth: $1320.00  SofiaDistribution:  Storage worth: $0.00 |

|  |
| --- |
| **Вход** |
| AddProduct HardDrive -20  RegisterStorage InvalidStorage LoshHackerStorage  RegisterStorage Warehouse GoodHackerStorage  SelectVehicle GoodHackerStorage 0  LoadVehicle HardDrive  SendVehicleTo LoshHackerStorage 0 GoodHackerStorage  SendVehicleTo GoodHackerStorage 0 LoshHackerStorage  END |
| **Изход** |
| Error: Price cannot be negative!  Error: Invalid storage type!  Registered GoodHackerStorage  Selected Semi  Error: HardDrive is out of stock!  Error: Invalid source storage!  Error: Invalid destination storage!  GoodHackerStorage:  Storage worth: $0.00 |

|  |
| --- |
| **Вход** |
| RegisterStorage DistributionCenter AmazonDistribution  RegisterStorage Warehouse AmazonWarehouse  AddProduct HardDrive 80  AddProduct HardDrive 70  AddProduct HardDrive 120  AddProduct Gpu 800  SelectVehicle AmazonDistribution 0  LoadVehicle SolidStateDrive  LoadVehicle HardDrive Gpu HardDrive  SendVehicleTo AmazonDistribution 0 AmazonWarehouse  GetStorageStatus AmazonWarehouse  UnloadVehicle AmazonWarehouse 3  GetStorageStatus AmazonWarehouse  END |
| **Изход** |
| Registered AmazonDistribution  Registered AmazonWarehouse  Added HardDrive to pool  Added HardDrive to pool  Added HardDrive to pool  Added Gpu to pool  Selected Van  Error: SolidStateDrive is out of stock!  Loaded 3/3 products into Van  Sent Van to AmazonWarehouse (slot 3)  Stock (0/10): []  Garage: [Semi|Semi|Semi|Van|empty|empty|empty|empty|empty|empty]  Unloaded 3/3 products at AmazonWarehouse  Stock (2.7/10): [HardDrive (2), Gpu (1)]  Garage: [Semi|Semi|Semi|Van|empty|empty|empty|empty|empty|empty]  AmazonWarehouse:  Storage worth: $990.00  AmazonDistribution:  Storage worth: $0.00 |