**Вендинг машини**

**Общ преглед**

Във вашата фирма постъпва проект за създаване на приложение, обслужващо набор от вендинг машини.

Вашият софтуер трябва да описва **машини** **(**VendingMachine**)** и **продукти (**Product**).**

Tрябва да реализирате функционалност, която да позволява създаване на продукти и машини, а в последствие реализиране на продажби – всичко това ще работи чрез **команди**, които вие ще получавате. Поредицата от команди приключва с „**End**”. За ваше удобство ще получите готов Program.cs файл, а вие ще трябва да реализирате само необходимите класове VendingMachine.cs и Product.cs

**Подзадача 1: Структура на Продукт – 15 точки**

**Product**

Всички продукти имат тип, име и цена:

type – низ, съставен от големи латински букви

name – низ, съставен от малки и големи латински букви, както и цифри, но без други специални знаци

                Уникална стойност, която различава всички продукти

price – число с плаваща запетая

Реализирайте конструктор:

* **Product(string type, string name, double price)** – този конструктор трябва да приема типа, името и цената на продукта.

За справка вижте следната схема за Product.cs:

|  |
| --- |
| Product.cs |
| public Product (**string**type**,** string name, double price)  {      //TODO: Добавете вашия код тук …  } |

**Командa за създаване на продукт**

Вашето приложение трябва да изпълнява следната команда за добавяне на продукти:

* **CreateProduct**<**тип**>  <**име**>  <**цена**> - тази команда има за цел да добави продукт с неговия тип, име и цена.

**Част от логиката на приложението е да се пази в променлива общия брой на всички поръчани продукти. За целта трябва да бъде добавен брояч, който да бъде увеличаван с 1 за всеки поръчан продукт.**

**Важно: Продукт се счита за поръчан, когато бъде изпълена команда за продажба от съответната вендинг машина.**

**Команда за извеждане на информация**

Вашето приложение във всеки един момент може да получи заявка да отпечата информация за продукт. Командата за това е следната:

* **PrintProductInfo**<**име**> - отпечатва информация за продукт във формат:  
  Product with type -**<тип>**and name -<**име>**
* Тази команда ще получава винаги валидни и съществуващи имена на продукти. За успешна реализация трябва да реализирате ваша версия на **ToString()**метода за класа **Product.**

**Подзадача 2: Структура на Вендинг машина – 15 точки**

**VendingMachine**

Всяка машина има: id, списък от продукти, обща стойност на продажбите, батерия.

id– низ, съставен от малки латински букви, уникална стойност за всяка машина

products– списък от **Product**

totalSalesAmount– число с плаваща запетая с първоначална стойност 0

battery– число с плаваща запетая. Всяка машина стартира с базисна стойност – 100 единици

Трябва да реализирате два констуктора:

* **VendingMachine(string id)** – този конструктор трябва да приема id.
* **VendingMachine(string id**, **List<Product> products)** - този конструктор трябва да приема id и списък от продукти.
* За справка вижте следната схема за VendingMachine.cs:

|  |
| --- |
| VendingMachine.cs |
| public VendingMachine(string id)  {        //TODO: Добавете вашия код тук …  }      public VendingMachine(string id, List<Product> products)  {        //TODO: Добавете вашия код тук …  } |

**Команда за създаване на вендинг машина**

Вашето приложение трябва да изпълнява следната команда за създаване на машина:

* **CreateVendingMachine**<**id>** - тази команда има за цел да създаде машина с даденото ID.
* **CreateVendingMachine**<**id, списък от продукти**>- тази команда има за цел да създаде машина с даденото ID и списък от продукти.

Уверете се, че всички полета имат валидни състояния!

**Команда за извеждане на информация**

Вашето приложение във всеки един момент може да получи заявка да отпечата информация за машина. Командата за това е следната:

* **PrintMachineInfo<ID> -**отпечатва информация за машина във формат:

Machine: <ID> has the following available products:

Product with type - <ProductType> and name - <ProductName>

Product with type - <ProductType> and name - <ProductName>

…/всички налични продукти, в реда, в който са постъпили в машината/

---- With total sales amount: <TotalSalesAmount>.

* Тази команда ще получава винаги валидни и съществуващи имена на машини. За успешна реализация трябва да реализирате ваша версия на **ToString()**метода за класа **VendingMachine.**Всички продукти трябва да бъдат изпечатани по дадения формат /ToString() на всеки от тях/, в реда, в който са постъпили в машината. Oчаква се обема на всички продажби **да бъде форматиран до два знака** след десетичния разделител.

**Подзадача 3: Логика – 40 точки**

**Kоманди**

Вашето приложение трябва да реализира следните команди:

* **RechargeMachine <ID> -**Презареждане на машината – нивото на батерията се вдига до първоначалната си стойност – 100 единици.
* **ClearMachineSales <ID> -**Стойността на продажбите следва да бъде занулена.
* **AddProductToMachine <ID> <име на продукт>**-Тази команда добавя продукт към дадената машина
* **CheckProductQuantity <ID> <тип продукт> -**Следва да се отпечата точния брой на продукти от определен тип в дадената машина.
* **RemoveProductByName <ID> <име на продукт> -**Командата премахва продукт с даденото име от машината със съответното id.
* **AddProduct <ID> <име на продукт> -** Командата добавя продукт с даденото име в машината със съответното id.
* **GetMostExpensiveProduct <ID> <име на продукт> -**Изпечатва се най-скъпият продукт, наличен в машината със съответното id. Гарантира се, че няма да има два продукта с еднакви цени.
* **SellProduct <ID> <име на продукт> -**Продажба на продукт. При всяка продажба нивото на батерията намалява. Стойността, с която се намалява, е равна на **цената на продукта  \* 0.8 + 2**. От списъка с продукти премахнете продаденото. Увеличете броя на тоталните продажби на продукти през класа Product с една единица, към TotalSalesAmount добавете стойността на продукта.

**Важно:** Трябва да се уверите, че разполагате с достатъчно батерия. В случай, че нивото е по-ниско, продажбата не се осъществява. Допълнително пояснения – в секцията за Валидации.

* **PrintMachineInfo <ID> -**Изпечатване на машината със съответното ID. Вика се метода ToString(). Форматирането е описано по-горе.
* **PrintProductInfo <име на продукт> -** Изпечатване на продукта с даденото име. Вика се метода ToString(). Форматирането е описано по-горе.
* **GetBatteryLevelOfMachine <ID> -**Изпечатва се текущото ниво на батерията на дадената машина.
* **GetMachineTotalSalesAmount<ID> -**Изпечатва се текущия обем на продажбите на дадената машина

**Подзадача 4: Бонус логика – 20 точки**

Трябва да реализирате още няколко команди:

* **RemoveAllProductsOfGivenType**<**ID**> <**тип продукт**>**-**Премахват се всички налични продукти от машината, които отговарят на посочения тип.
* **PrintInfoAboutAllProductsByType**<**ID**>**-** Изпечатва се информация за всички продукти от съответната машина. Следва да подредите продуктите по брой в нарастващ ред и след това по тип по азбучен ред. За всеки тип се изпечатва следният текст на нов ред –

Type: **<ProductsType>** has total of - **<Брой продукти от този тип>** products.

* **GetTotalProductSales -**Изпечатва общият брой на **продадени** продукти

**За безпроблемната работа на всички изброени команди от задачи 3 и 4 подзадача трябва да реализирате Product.cs и VendingMachine.cs, по аналогичен начин на показаното по-долу:**

|  |
| --- |
| Product.cs |
| public Product(string type, string name, double price)  {     //TODO: Добавете вашия код тук …  }    public static void IncreaseOrdersCount()  {     //TODO: Добавете вашия код тук …  }    public override string ToString()  {     //TODO: Добавете вашия код тук …  } |

|  |
| --- |
| VendingMachine.cs |
| public VendingMachine(string id) : this(id, new List<Product>())  {  //TODO: Добавете вашия код тук …  }    public VendingMachine(string id, List<Product> products)  {  //TODO: Добавете вашия код тук …  }    public void Recharge()  {  //TODO: Добавете вашия код тук …  }        public void ClearSales()  {   //TODO: Добавете вашия код тук …  }  public int CheckProductQuantityOfGivenType(string type)  {  //TODO: Добавете вашия код тук …  }    public void AddProduct(Product product)  {  //TODO: Добавете вашия код тук …  }    public void RemoveProduct(string productName)  {  //TODO: Добавете вашия код тук …  }    public Product GetMostExpensiveProduct()  {  //TODO: Добавете вашия код тук …  }    public string SellProduct(string productName)  {  //TODO: Добавете вашия код тук …  }    public override string ToString()  {   //TODO: Добавете вашия код тук …  }    public void RemoveAllProductsOfGivenType(string type)  {  //TODO: Добавете вашия код тук …  }    public string GetInfAboutAllProductsByType()  {  //TODO: Добавете вашия код тук …  } |

**Забележка:**Освен горепосочените методи трябва да реализирате и необходимите свойства за всеки от класовете. Възможно е да е удачно да реализирате допълнителни полета, свойства и методи, по ваша преценка.

**Подзадача 5: Валидация – 10 точки**

Освен всичко останало вие трябва да направите и валидация!

Не допускайте създаването на:

* Продукт-типът трябва да бъде текст създаден само от главни букви, ако не -  **Invalid type!**
* Продукт-името трябва да бъде с дължина над 2 символа - **Invalid name!**
* Цената на продукта трябва да положително число, ако не - **Invalid price!**
* ID на вендинг машина трябва да бъде текст, съставен само от малки букви и с дължина по-голяма от 3 символа, ако не - **Invalid machine id!**
* SaleProduct – Валидацията, следва да проверява дали има достатъчно батерия, ако няма изведете съобщение, ако не - **Out of battery!**

При невалидни данни хвърлете грешка от тип **ArgumentException(“message”)**

**Допълнително информация свързана с валидацията**

Командите **CreateProduct, CreateVendingMachine**могат да съдържат информация, която да наруши правилата от по-горе. Въпреки това, имената на продуктите и машините в тези команди **винаги** ще бъдат низове, съдържащи малки и/или големи латински букви, без каквито и да е други символи.

На методите **RechargeMachine()**, **ClearMachineSales**(), **AddProductToMachine(), CheckProductQuantity(), RemoveProductByName**(), **AddProduct**(), **GetMostExpensiveProduct**(), **SellProduct(), PrintMachineInfo**(), **PrintProductInfo(), GetBatteryLevelOfMachine**(), **GetMachineTotalSalesAmount(), RemoveAllProductsOfGivenType**(), **PrintInfoAboutAllProductsByType**(), **GetTotalProductSales()**

**ВИНАГИ**ще се подават продукт/машина, които съществуват.

**Ако получите команда, която се опитва да извърши операция, която нарушава валидацията, хвърлете изключение с текст, посочен за съответната валидация!**

**Вход / Изход**

**Вход**

* Програмата ще получава множество редове с информация. Всеки ред представлява команда. Самият вход се обработва изцяло от примерния Program.cs.
* Всички команди приключват с въвеждането на End

**Изход**

За някои от командите не е нужно да извеждате нищо. За други е необходимо форматиране на изход – напр. ToString(), PrintInfoAboutAllProductsByType(), SellProduct().

**Ограничения**

* Всички цели числа ще бъдат в диапазона **–10000** до **+10000**
* Всички числа с плаваща запетая ще бъдат въвеждани с до два знака след запетаята.
* Всички имена няма да съдържат интервал

**Примери**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| CreateProduct SWEET twix 2.59  CreateVendingMachine iddd  PrintProductInfo twix  End | Product with type - SWEET and name - twix |
| CreateProduct SWEET twix 2.59  CreateVendingMachine iddd  AddProductToMachine iddd twix  CreateProduct SWEET kitkat 1.27  AddProductToMachine iddd kitkat  PrintMachineInfo iddd  End | Machine: iddd has the following available products:  Product with type - SWEET and name - twix  Product with type - SWEET and name - kitkat  ---- With total sales amount: 0.00. |
| CreateProduct SWEET twix 2.59  CreateVendingMachine iddd  AddProductToMachine iddd twix  PrintProductInfo twix  CreateProduct SWEET kitkat 1.27  AddProductToMachine iddd kitkat  PrintMachineInfo iddd  GetMostExpensiveProduct iddd  SellProduct iddd twix  GetBatteryLevelOfMachine iddd  RechargeMachine iddd  GetBatteryLevelOfMachine iddd  SellProduct iddd kitkat  GetMachineTotalSalesAmount iddd  GetTotalProductSales  ClearMachineSales iddd  CheckProductQuantity iddd SWEET  RemoveProduct iddd twix  RemoveAllProductsOfGivenType iddd SWEET  PrintInfoAboutAllProductsByType iddd  CreateProduct malaktip twixx 2.59  CreateProduct KUSOIME t 2.59  CreateProduct GOLQMTIP otricatelnacena -5  CreateVendingMachine GOLQMOID  CreateVendingMachine i  CreateProduct VEGAN veganfood 444.50  AddProductToMachine iddd veganfood  SellProduct iddd veganfood  End | Product with type - SWEET and name - twix  Machine: iddd has the following available products:  Product with type - SWEET and name - twix  Product with type - SWEET and name - kitkat  ---- With total sales amount: 0.00.  Machine's with ID = iddd most expensive product is: Product with type - SWEET and name - twix  Machine's with ID = iddd sold twix for 2.59lv.  Machine's with ID = iddd has battery level = 95.93  Machine's with ID = iddd has battery level = 100.00  Machine's with ID = iddd sold kitkat for 1.27lv.  3.86  2  0  Invalid type!  Invalid name!  Invalid price!  Invalid machine id!  Invalid machine id!  Out of battery! |

**Точки**

Разбивката по подзадачи е следната:

1. **15** точки, като трябва да имате и задължително реализиран ToString() метод
2. **15** точки, като трябва да имате и задължително реализиран ToString() метод
3. **40** точки
4. **20** точки
5. **10** точки

Общ брой точки: **100**