automaten

Contents

link til git hub

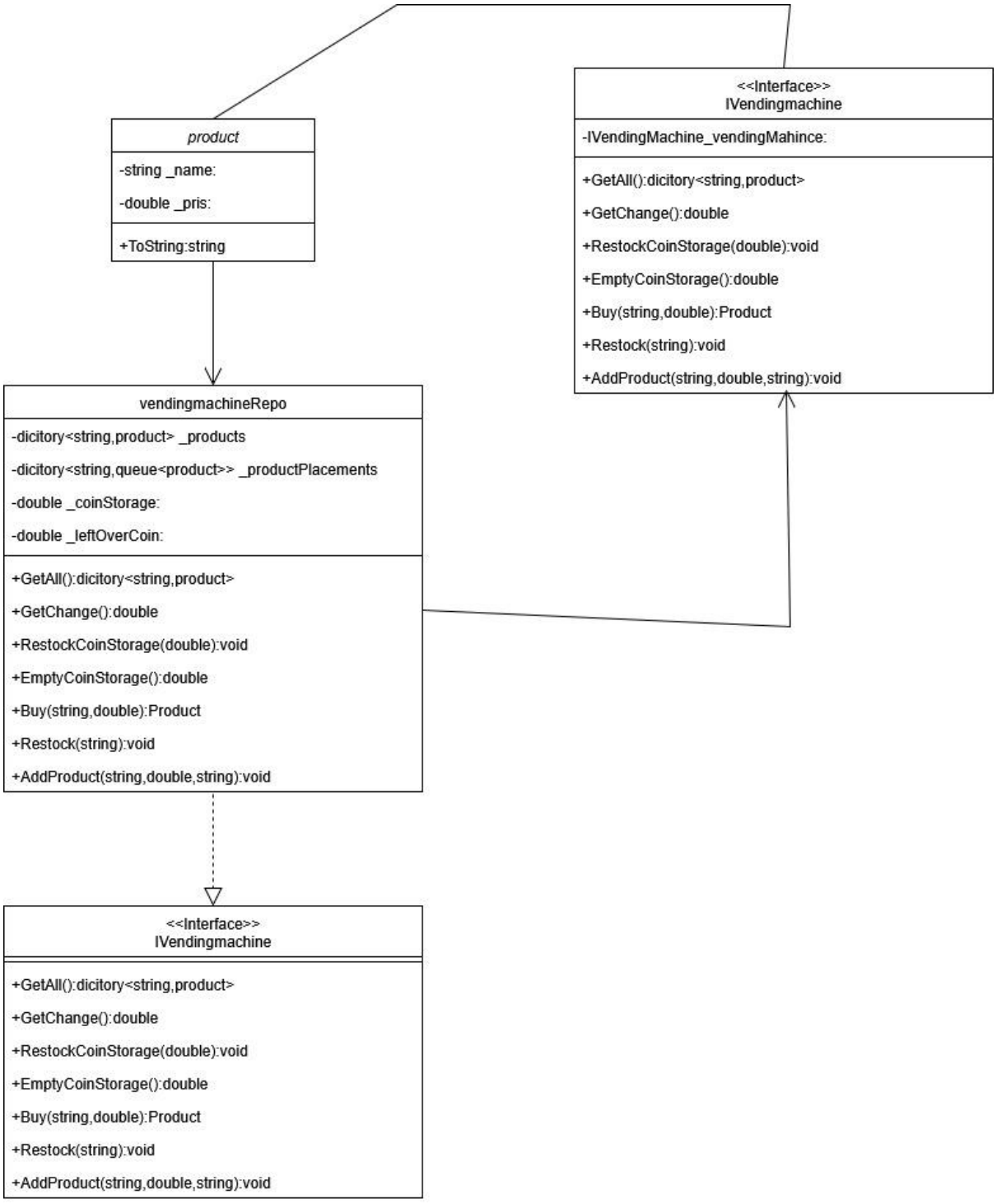
https://github.com/AureliaCrestfall/automat

User story's:

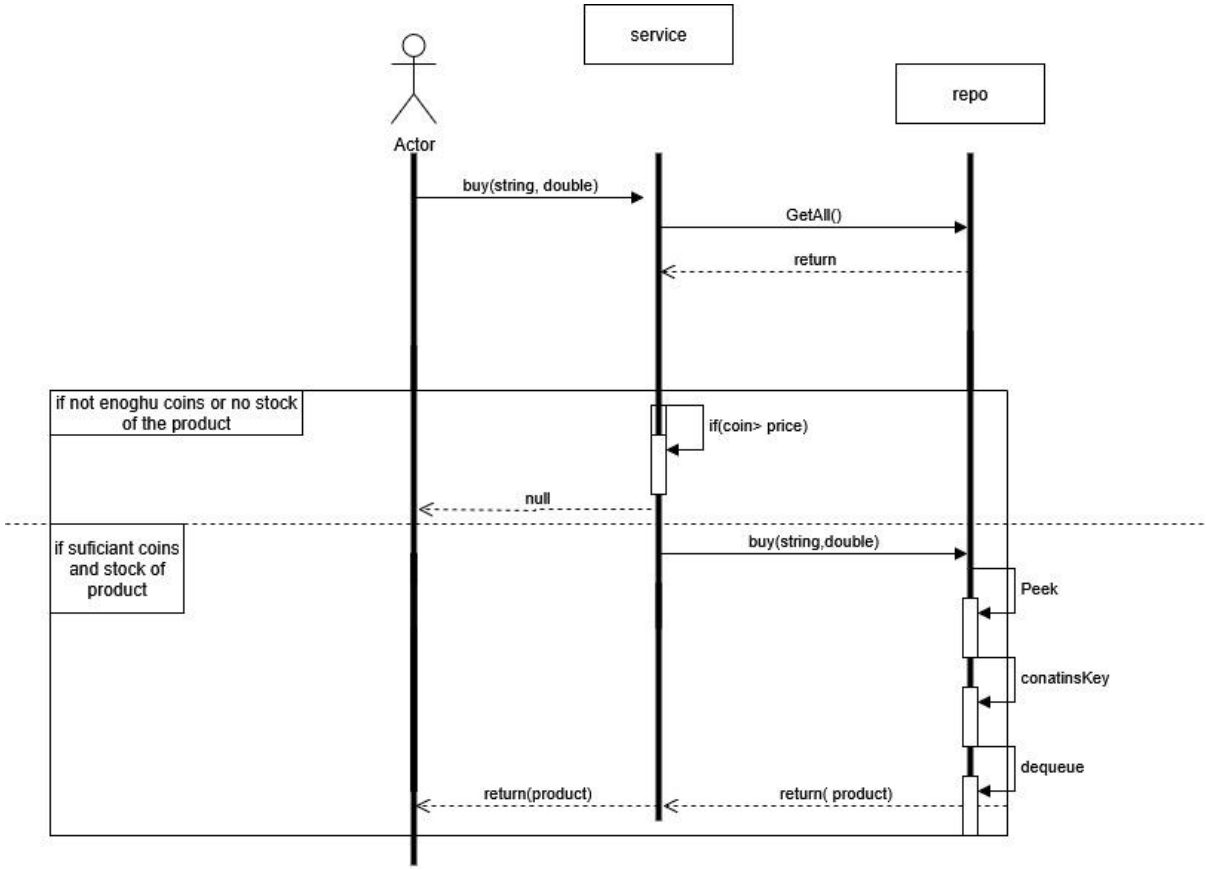
1. Som kunde vil jeg gerne kunne få penge tilbage hvis jeg putter for mange ind efter at have købt et produkt
2. Som admin vil jeg gerne kunne genopfule produkt
3. Som kunde vil jeg gerne kunne modtage produkt når jeg har købt det
4. Som admin vil jeg gerne tøme automaten få penge
5. Som kunde vil jeg gerne kunne vælge et produkt
6. Som kunde vil jeg gerne kunne få penge tilbage hvis jeg ikke har købt noget men har puttet penge i automaten
7. som kunde vil jeg gerne kunne putte flere mønter ind samme tidlig
8. som kunde vil jeg gerne kunne se produkterne i automaten

9. som admin vil jeg gerne kunne tilføje eller fjene produkter i automaten

klassedigram



system sekvens diagram


service
repo
Actor
buy(string, double)
GetAll()
return
if not enoghu coins or no stock of the product
if(coin> price)
null
if suficiant coins and stock of product
buy(string,double)
Peek
conatinsKey
dequeue
return(product)
return(product)

kode

model

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace automat.Model
{
    29 references | Martin Egholm Sørensen, 9 hours ago | 2 authors, 4 changes
    internal class Product
    {
        /// <summary>
        /// class for creation of products for the vending machine
        /// </summary>
        string _name;
        double _pris;
        /// <summary>
        /// instanciring of variables
        /// </summary>
        2 references | Martin Egholm Sørensen, 9 hours ago | 2 authors, 3 changes
        public string Name
        {
            get { return _name; }
            set { _name = value; }
        }
        5 references | Martin Egholm Sørensen, 1 day ago | 2 authors, 2 changes
        public double Pris
        {
            get { return _pris; }
            set { _pris = value; }
        }
```

```
/// <summary>
/// unchanined constorter for product
/// </summary>
/// <param name="pris"> for assinging a price to the object</param>
/// <param name="name">for assinging a name to the object</param>
    5 references | Martin Egholm Sørensen, 9 hours ago | 2 authors, 2 changes
    public Product(double pris, string name)
    {
        Pris = pris;
        Name = name;

    }
    0 references | Martin Egholm Sørensen, 1 day ago | 1 author, 1 change
    public override string ToString()
    {
        return Name;
    }

}
}
```

Ivendingmachine

```
using automat.Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace automat.Repo
{
    3 references | Martin Egholm Sørensen, 9 hours ago | 2 authors, 4 changes
    internal interface IVendingMachine
    {
        /// <summary>
        /// contract for vendingmachineRepo so it has to include the following metodes
        /// </summary>

        3 references | Martin Egholm Sørensen, 9 hours ago | 1 author, 2 changes
        Dictionary<string, Product> GetAll();
        2 references | Martin Egholm Sørensen, 1 day ago | 1 author, 1 change
        void RestockCoinStoage(double newcoin);
        2 references | Martin Egholm Sørensen, 1 day ago | 2 authors, 2 changes
        double EmptyCoinStoage();
        2 references | Martin Egholm Sørensen, 9 hours ago | 1 author, 1 change
        Double GetChange();
        2 references | Martin Egholm Sørensen, 1 day ago | 1 author, 1 change
        Product Buy(string product, double coin);
        2 references | Martin Egholm Sørensen, 1 day ago | 1 author, 1 change
        void Restock(string productPlace);
        2 references | Martin Egholm Sørensen, 1 day ago | 1 author, 1 change
        void AddProduct(string newProductName, double newProductPrise, string productPlace);


    }
}
```

vendingmachinerepo

```cs
using automat.Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace automat.Repo
{
    3 references | crestfall, Less than 5 minutes ago | 2 authors, 5 changes
    internal class VendingMachineRepo:IVendingMachine
    {
        double _coinStoage;
        Dictionary<string, Queue<Product>> _produtPlacements;
        Dictionary<string, Product> _products;
        double _leftOverCoin;

        1 reference | Martin Egholm Sørensen, 9 hours ago | 1 author, 1 change
        double LeftOverCoin
        {
            get{ return _leftOverCoin; }
            set { _leftOverCoin = value; }
        }

        13 references | Martin Egholm Sørensen, 1 day ago | 1 author, 1 change
        Dictionary<string, Product> Products
        {
            get { return _products; }
            set { _products = value; }
        }
        4 references | Martin Egholm Sørensen, 9 hours ago | 1 author, 1 change
        double coinStoage
        {
            get { return _coinStoage; }
            set { _coinStoage = value; }
        }
    }
```

```cs
    19 references | Martin Egholm Sørensen, 9 hours ago | 1 author, 1 change
    public Dictionary<string, Queue<Product>> ProdutPlacements
    {
        get { return _produtPlacements; }
        set { _produtPlacements = value; }
    }
    /// <summary>
    ///  Method for getting all of the contents for the dictionary
    /// </summary>
    /// <returns>returns the dictionary ProdutPlacement </returns>
    3 references | Martin Egholm Sørensen, 9 hours ago | 1 author, 2 changes
    public Dictionary<string, Product> GetAll()
    {
        return Products;
    }
    /// <summary>
    /// method for getting back change from the Buy method
    /// </summary>
    /// <returns> returns change as a double</returns>
    2 references | Martin Egholm Sørensen, 9 hours ago | 1 author, 1 change
    public Double GetChange()
    {
        double change = LeftOverCoin;
        _leftOverCoin = 0;

        return change;
    }
```

```
/// <summary>
/// Method for filling CoinStoage with newcoin
/// </summary>
/// <param name="newcoin">the variable that gets added to coinStoage</param>
2 references | Martin Egholm Sørensen, 9 hours ago | 1 author, 2 changes
public void RestockCoinStoage(double newcoin)
{
    coinStoage += newcoin;
}
/// <summary>
/// empties Coinstoage and returns the values as coinsGained
/// </summary>
/// <returns> returns coinsGained as a double</returns>
2 references | Martin Egholm Sørensen, 9 hours ago | 2 authors, 3 changes
public double EmptyCoinStoage()
{
    double coinsGained = coinStoage;
    coinStoage = 0;
    return coinsGained;
}
```

```
/// <summary>
/// handles buying products by using the param product to location the product as the key value
/// and coin as the payment for the product and then putting the left over double value in _leftOverCoin
/// for the use of the GetChange method and then returning a prodct by dequeueing the dictionary
/// else if the dictionary doesn't conntain the key value it reurns null
/// </summary>
/// <param name="product"> for finding the kay value in dictionary</param>
/// <param name="coin"> for calculating the what value should go into _leftOverCoin by - coin and the price for the product in queue</param>
/// <returns>returns a product unless the kay value dosn't exists</returns>
2 references | crestfall, Less than 5 minutes ago | 2 authors, 3 changes
public Product Buy(string product, double coin)
{
    Product buyPlaceholder = ProdutPlacements[product].Peek();

    if (ProdutPlacements.ContainsKey(product))
    {
        coin -= buyPlaceholder.Pris;
        _leftOverCoin = coin;
        _coinStoage += buyPlaceholder.Pris;

        return ProdutPlacements[product].Dequeue();

    }
    return null;

}
```

```
/// <summary>
/// method for restocking the queue in the dictionary when called enqueues the product
/// at the key value productPlace until their is 10 of them in the queue using the Products dictionary
/// to make sure that the correct product is enqueued in the queue
/// </summary>
/// <param name="productPlace">for finding the kay value in the dictionaries</param>
2 references | crestfall, 5 minutes ago | 2 authors, 3 changes
public void Restock(string productPlace)
{
      while(ProdutPlacements[productPlace].Count() <= 10)
      {
          ProdutPlacements[productPlace].Enqueue(Products[productPlace]);
      }
}
/// <summary>
/// adds a product to the dictionary ProdutPlacements by using newProductName and newProductPrise to create a new ob
/// the adding the object to Products using productPlace as the kay value and then enqueueing it ind
/// ProductPlacements using the same key value as before
/// if ProdutPlacements already contains a key value of productplace then it just enqueues the producr
/// ind that queue
/// </summary>
/// <param name="newProductName">for the name of the new product</param>
/// <param name="newProductPrise">for the pris of the new product</param>
/// <param name="productPlace">for the use of a key value</param>
2 references | Martin Egholm Sørensen, 9 hours ago | 1 author, 2 changes
public void AddProduct(string newProductName, double newProductPrise,string productPlace)
{
    Product productSemiPlacement = new Product(newProductPrise, newProductName);

    if (ProdutPlacements.ContainsKey(productPlace))
    {
        ProdutPlacements[productPlace].Enqueue(productSemiPlacement);
    }
    else
    {
        Products.Add(productPlace, productSemiPlacement);
        ProdutPlacements.Add(productPlace, new Queue<Product>());
        ProdutPlacements[productPlace].Enqueue(productSemiPlacement);
    }
```

```
    1 reference | Martin Egholm Sørensen, 9 hours ago | 2 authors, 3 changes
    public VendingMachineRepo()
    {
        coinStoage = 0;
        ProdutPlacements = new Dictionary<string, Queue<Product>>();
        ProdutPlacements.Add("a1", new Queue<Product>());
        ProdutPlacements.Add("a2", new Queue<Product>());
        ProdutPlacements.Add("a3", new Queue<Product>());
        ProdutPlacements.Add("a4", new Queue<Product>());
        Products = new Dictionary<string, Product>();
        Products.Add("a1", new Product(5, "faxe"));
        Products.Add("a2", new Product(20, "ice tea"));
        Products.Add("a3", new Product(8, "cola"));
        Products.Add("a4", new Product(2, "monster"));
        ProdutPlacements["a1"].Enqueue(Products["a1"]);
        ProdutPlacements["a1"].Enqueue(Products["a1"]);
        ProdutPlacements["a2"].Enqueue(Products["a2"]);
        ProdutPlacements["a3"].Enqueue(Products["a3"]);
        ProdutPlacements["a4"].Enqueue(Products["a4"]);
    }
}
```

Vendginmachineservice

```
IVendingMachine _vendingMahince;
1 reference | crestfall, 1 day ago | 1 author, 1 change
public VendingmachineService(IVendingMachine repo)
{
    _vendingMahince = repo;
}

0 references | Martin Egholm Sørensen, 1 day ago | 1 author, 1 change
public void RestockCoinStoage(double newcoin)
{
    _vendingMahince.RestockCoinStoage(newcoin);
}

0 references | Martin Egholm Sørensen, 1 day ago | 2 authors, 2 changes
public double EmptyCoinStoage()
{
    return _vendingMahince.EmptyCoinStoage();
}
```

```
/// <summary>
/// uses the Getall and Peek method to check if their is a high enough value in coin to beable to make the
/// purchase before calling the buy method in VendingMachineRepo and then returning a product else it returns nul
/// </summary>
/// <param name="product">for use as a key value to locate the product</param>
/// <param name="coin">for use to be able to pay if the value is high enough</param>
/// <returns>returns a product</returns>
1 reference | crestfall, 5 minutes ago | 2 authors, 3 changes
public Product Buy(string product, double coin)
{
    Dictionary<string, Product> buyDictionary = _vendingMahince.GetAll();

    Product itembuying = buyDictionary[product];
    if (coin >= itembuying.Pris)
    {
        return _vendingMahince.Buy(product, coin);

    }
    else
    {
        return null;
    }
}

1 reference | Martin Egholm Sørensen, 1 day ago | 1 author, 1 change
public void Restock(string productPlace)
{
    _vendingMahince.Restock(productPlace);
}

0 references | Martin Egholm Sørensen, 1 day ago | 1 author, 1 change
public void addProduct(string newProductName, double newProductPrise, string productPlace)
{
    _vendingMahince.AddProduct( newProductName, newProductPrise,productPlace);
}
```

```
1 reference | Martin Egholm Sørensen, 9 hours ago | 1 author, 1 change
public Double GetChange()
{
    return _vendingMahince.GetChange();
}

2 references | Martin Egholm Sørensen, 9 hours ago | 1 author, 1 change
public Dictionary<string, Product> GetAll()
{
    return _vendingMahince.GetAll();
}
```

Main

```
VendingMachineRepo vendingRepo = new VendingMachineRepo();
VendingmachineService vendingservice = new VendingmachineService(vendingRepo);

Dictionary<string, Product> vendingmachine = vendingservice.GetAll();
Console.WriteLine("here is a list of the current items we have");

foreach (KeyValuePair<string, Product> kv in vendingmachine)
{
    vendingservice.Restock(kv.Key);
    string place = kv.Key;

    Console.WriteLine("place " + kv.Key + " "+ vendingmachine[place]+ " price "+ vendingmachine[place].Pris+" coins

}
vendingmachine = vendingservice.GetAll();

bool unpaid = false;

while (unpaid == false)
{
    Console.Write("enter witch product you want: ");
    string choois = Console.ReadLine();
    Console.Write("enter coins: ");
    int coin = int.Parse(Console.ReadLine());

        Console.WriteLine(vendingservice.Buy(choois, coin)+" obtained");
        Console.WriteLine("you get "+vendingservice.GetChange()+" coins back");
```

```
    Console.Write("type 1 if you wish to buy more from the vending machine: ");
    try
    {
        string continu = Console.ReadLine();
        if(continu != "1")
        {
            unpaid = true;
        }
    }
    catch
    {
        Console.WriteLine("thank you for using the vending machine");
    }

}
```