

1 Program.cs

```
1 using System.IO;
2
3 namespace Eksamensopgave2017
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             // In the start all files will be loaded and "database" lists will be
9             ↪ populated
10            File.WriteAllText(@"../../transactions.csv", string.Empty);
11            ReadFromCsv csvReader = new ReadFromCsv();
12            StregsystemCLI ui = new StregsystemCLI();
13            csvReader.CreateUserList("../../users.csv");
14            csvReader.CreateProductList("../../products.csv");
15            csvReader.CreateTransactionList();
16            //We will now start the interface
17            ui.Start();
18        }
19    }
20 }
21 }
```

2 BuyTransaction.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4
5 namespace Eksamensopgave2017
6 {
7     public class BuyTransaction : Transaction
8     {
9         public BuyTransaction(User user, Product product, DateTime date, decimal
9         ↪ amount, List<User> userlist) : base(user, date, amount)
10        {
11            if (user == null)
12                throw new ArgumentNullException();
13            else
14                this.user = user;
15
16            this.product = product;
```

```

17         this.Date = date;
18         this.Amount = amount;
19         this.UserList = userlist;
20     }
21
22     User user;
23     Product product;
24     DateTime Date;
25     decimal Amount;
26     List<User> UserList;
27
28     public override string ToString()
29     {
30         return $"New purchase ({this.Id}): {this.user.Firstname}
31         ↳ {this.user.Lastname} - {this.Amount} - {this.product.Name} -
32         ↳ [{this.Date}]";
33     }
34
35     public override void Execute()
36     {
37         if (this.user.Balance >= this.product.Price ||
38             ↳ this.product.CanBeBoughtOnCredit)
39         {
40             decimal NewBalance = this.user.Balance - Amount;
41             foreach (var Item in this.UserList.Where(x => x.Id ==
42                 ↳ user.Id))
43             {
44                 Item.Balance = NewBalance;
45             }
46             FileWriters writer = new FileWriters();
47             //The new information which have been written to the list will
48             ↳ over write the old file
49             writer.WriteToUserCsv("../users.csv", UserList);
50             //The transaction is written to the transaction file
51             writer.WriteToTransactionCsv("../transactions.csv",
52                 ↳ this.ToString());
53         }
54         else
55         {
56             return;
57         }
58     }
59 }
60
61 }
62
63 }

```

3 FileWriters.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.IO;
4
5 namespace Eksamensopgave2017
6 {
7     class FileWriters
8     {
9         public void WriteToProductCsv(string filepath, List<Product>
10             ↪ list) //https://www.codeproject.com/Articles/415732/Reading-and-Writing-CSV-Files-in-Csharp
11         {
12             using (CsvFileWriter writer = new CsvFileWriter(filepath))
13             {
14                 CsvRow FirstRow = new CsvRow();
15                 //First row will be the description row
16                 FirstRow.Add(String.Format("id;name;price;active"));
17                 writer.WriteLine(FirstRow);
18                 CsvRow row = new CsvRow();
19                 //For each row in the file we will now fill the file
20                 foreach (Product element in list)
21                 {
22                     //We need to add it all as strings
23                     string active;
24                     if (element.Active == true)
25                         active = "1";
26                     else
27                         active = "0";
28                     row.Add(element.Id.ToString() + ";" + element.Name.ToString() + ";" +
29                         ↪ element.Price.ToString() + ";" + active + "\n");
30                 }
31                 writer.WriteLine(row);
32             }
33         }
34         public void WriteToUserCsv(string filepath, List<User> list)
35         {
36             using (CsvFileWriter writer = new CsvFileWriter(filepath))
37             {
38                 CsvRow FirstRow = new CsvRow();
39                 //First row will be the description row
40
41                 ↪ FirstRow.Add(String.Format("id;firstname;lastname;username;email;balance"));
42                 writer.WriteLine(FirstRow);
43                 CsvRow row = new CsvRow();
44                 //For each row in the file we will now fill the file
```

```

42         foreach (User element in list)
43         {
44             row.Add($"{ element.Id.ToString() };\"{
                ↪ element.Firstname.ToString() }\";\"{
                ↪ element.Lastname.ToString() }\";\"{
                ↪ element.Username.ToString() }\";\"{
                ↪ element.Email.ToString() }\";\"{
                ↪ element.Balance.ToString()} \n");
45         }
46         writer.WriteLine(row);
47     }
48 }
49 public void WriteToTransactionCsv(string filepath, string
    ↪ transactioninfo)//https://social.msdn.microsoft.com/Forums/vstudio/en-US/0271c11d-4cf3-452b-af65-6c06
50 {
51     //We create a list, add a single string to it and them writes it to
    ↪ the file
52     List<string> newLines = new List<string>();
53     newLines.Add(transactioninfo.ToString());
54     File.AppendAllLines(filepath, newLines);
55 }
56 }
57 }
58 }
59 }

```

4 InsertCashTransaction.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Eksamensopgave2017
8  {
9      public class InsertCashTransaction : Transaction
10     {
11         public InsertCashTransaction(User user, DateTime date, decimal amount,
            ↪ List<User> userlist) : base(user, date, amount)
12         {
13             if (user == null)
14                 throw new ArgumentNullException();
15             else
16                 this.user = user;

```

```

17         this.Date = date;
18         this.Amount = amount;
19         this.UserList = userlist;
20     }
21
22     User user;
23     DateTime Date = DateTime.Now;
24     decimal Amount;
25     List<User> UserList;
26
27     public override string ToString()
28     {
29         return $"Cash insertet ({this.Id}): {this.user.Firstname}
30         ↳ {this.user.Lastname} - DKK {this.Amount} - [{this.Date}];
31     }
32
33     public override void Execute()
34     {
35         decimal NewBalance = this.user.Balance + Amount;
36         foreach (var Item in this.UserList.Where(x => x.Id == user.Id))
37         {
38             Item.Balance = NewBalance;
39         }
40         FileWriters writer = new FileWriters();
41         //The new information which have been written to the list will over
42         ↳ write the old file
43         writer.WriteToUserCsv("../users.csv", UserList);
44         //The transaction is written to the transaction file
45         writer.WriteToTransactionCsv("../transactions.csv",
46         ↳ this.ToString());
47     }
48 }
49

```

5 Istregsystem.cs

```

1 using System;
2 using System.Collections.Generic;
3
4 namespace Eksamensopgave2017
5 {
6     public interface Istregsystem

```

```

7      {
8          IEnumerable<Product> ActiveProducts { get; }
9          InsertCashTransaction AddCreditsToAccount(User user, decimal amount,
            ↳ List<User> UserList);
10         BuyTransaction BuyProduct(User user, Product product, List<User>
            ↳ UserList, List<Product> ProductList);
11         Product GetProductByID(int productID, List<Product> ProductList);
12         IEnumerable<Transaction> GetTransactions(User user, int count);
13         User GetUser(Func<User, bool> predicate);
14         User GetUserByUsername(string username, List<User> UserList);
15     }
16 }
17 }

```

6 IstregsystemUI.cs

```

1 namespace Eksamensopgave2017
2 {
3     public interface IstregsystemUI
4     {
5         void DisplayUserNotFound(string username);
6         void DisplayProductNotFound(string product);
7         void DisplayUserInfo(User user);
8         void DisplayTooManyArgumentsError(string command);
9         void DisplayTooFewArgumentsError();
10        void DisplayAdminCommandNotFoundMessage(string adminCommand);
11        void DisplayUserBuysProduct(BuyTransaction transaction);
12        void DisplayUserBuysProduct(int count, BuyTransaction transaction);
13        void Close();
14        void DisplayInsufficientCash(User user, Product product);
15        void DisplayGeneralError(string errorString);
16        void DisplayInactiveProduct(Product product);
17        void Start();
18    }
19 }
20 }

```

7 Product.cs

```
1  using System;
2  using System.Collections.Generic;
3
4  namespace Eksamensopgave2017
5  {
6      public class Product
7      {
8          public List<Product> ProductList = new List<Product>();
9          public Product(string name, int active, int canbeboughtoncredit, decimal
10             ↪ price)
11          {
12              if (name == null || name == "")
13                  throw new ArgumentNullException();
14              else
15                  this.Name = name;
16
17              if (active == 1)
18                  this.Active = true;
19              if (active == 0)
20                  this.Active = false;
21
22              if (canbeboughtoncredit == 1)
23                  this.CanBeBoughtOnCredit = true;
24
25              this.Price = price;
26          }
27
28          public readonly int Id = _nextID++;
29          private static int _nextID = 1;
30          public string Name;
31          public decimal Price;
32          public bool Active;
33          public bool CanBeBoughtOnCredit = false;
34
35          public override string ToString() //Make other return type
36          {
37              return $"{this.Name.ToString()} {this.Price.ToString()}
38                 ↪ ({this.Id.ToString()}) \n";
39          }
40
41          public int CompareTo(Product other)
42          {
43              return this.Name.CompareTo(other.Name);
44          }
45      }
46  }
```

```
44 }
45 }
```

8 ReadFromCsv.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4
5  namespace Eksamensopgave2017
6  {
7      class ReadFromCsv
8      {
9
10         ↪ //https://www.codeproject.com/Answers/384264/How-to-read-from-csv-file-using-csharp#answer2
11         ↪ //https://msdn.microsoft.com/da-dk/library/98f28cdx.aspx
12         ↪ //Static lists is puplicated so they can be used in all the classes
13         ↪ //They will work as a "database"
14         public static List<Product> ProductList = new List<Product>();
15         public static List<SeasonalProduct> SeasonalProductList = new
16         ↪ List<SeasonalProduct>();
17         public static List<User> UserList = new List<User>();
18         public static List<Transaction> TransactionList = new
19         ↪ List<Transaction>();
20         Stregsystem sSystem = new Stregsystem();
21         public void CreateUserList(string fileLocation)
22         {
23             StreamReader sr = new StreamReader(String.Format(@"{0}",
24             ↪ fileLocation));
25
26             string strline = "";
27             string[] _values = null;
28             int i = 0;
29             while (!sr.EndOfStream)
30             {
31                 i++;
32                 ↪ //Remove all " signes
33                 strline = sr.ReadLine().Replace("\\"", "");
34                 _values = strline.Split(';');
35                 if (_values.Length == 6 && _values[0].Trim().Length > 0)
36                 {
37                     if (_values[0] == "id")
38                     {
39                         ↪ //Remove the desription line
40                     }
41                 }
42             }
43         }
44     }
45 }
```



```

37         else
38         {
39             //Add to list and create an instance of user
40             string firstname = _values[1];
41             string lastname = _values[2];
42             string username = _values[3];
43             string email = _values[4];
44             decimal balance = Convert.ToDecimal(_values[5]);
45
46             UserList.Add(new User(firstname, lastname, username,
47                                     ↪ email, balance));
48         }
49     }
50     sr.Close();
51 }
52
53 public void CreateProductList(string fileLocation)
54 {
55     StreamReader sr = new StreamReader(String.Format@"{0}",
56                                     ↪ fileLocation));
57
58     string strline = "";
59     string[] _values = null;
60     int i = 0;
61     while (!sr.EndOfStream)
62     {
63         i++;
64         strline = sr.ReadLine();
65         //Remove " and HTML tags
66         strline = strline.Replace("\"", "").Replace("<h2>",
67                                     ↪ "").Replace("</h2>", "").Replace("<b>", "").Replace("</b>",
68                                     ↪ "").Replace("<h1>", "").Replace("</h1>", "").Replace("<h3>",
69                                     ↪ "").Replace("<h3>", "").Replace("<blink>",
70                                     ↪ "").Replace("</blink>", "");
71         _values = strline.Split(';');
72         if (_values.Length == 4 && _values[0].Trim().Length > 0)
73         {
74             if (_values[0] == "id")
75             {
76                 //Remove the desription line
77             }
78             else
79             {
80                 string name = _values[1];
81                 decimal price = Convert.ToDecimal(_values[2]);
82                 int active = 0;
83                 if (_values[3] == "1")

```

```

79         active = 1;
80         //We need active to be integer, so we change it
81         ProductList.Add(new Product(name, active, 0, price));
82     }
83 }
84 }
85 sr.Close();
86 //Pupulate Seasonal ProductList and create instances
87 SeasonalProductList.Add(new SeasonalProduct("Coffe (Black/Latte)", 1,
88     ↪ 0, 23, 2016, 01, 01, 2017, 09, 01));
89 SeasonalProductList.Add(new SeasonalProduct("Wine (Red)", 1, 0, 456,
90     ↪ 2016, 01, 01, 2017, 09, 01));
91 SeasonalProductList.Add(new SeasonalProduct("Wine (Wite)", 1, 1, 234,
92     ↪ 2016, 01, 01, 2017, 01, 01));
93 }
94
95 public void CreateTransactionList()
96 {
97     ////Pupulate transaction list and create instances of both types
98     TransactionList.Add(new
99     ↪ BuyTransaction(sSystem.GetUserByUsername("fmikkelsen",
100     ↪ ReadFromCsv.UserList), sSystem.GetProductByID(11,
101     ↪ ReadFromCsv.ProductList), new DateTime(2012,1,18),
102     ↪ sSystem.GetProductByID(11, ReadFromCsv.ProductList).Price,
103     ↪ ReadFromCsv.UserList));
104     TransactionList.Add(new
105     ↪ InsertCashTransaction(sSystem.GetUserByUsername("bsmith",
106     ↪ ReadFromCsv.UserList), new DateTime(2014, 1, 18), 4004,
107     ↪ ReadFromCsv.UserList));
108     TransactionList.Add(new
109     ↪ BuyTransaction(sSystem.GetUserByUsername("bsmith",
110     ↪ ReadFromCsv.UserList), sSystem.GetProductByID(13,
111     ↪ ReadFromCsv.ProductList), new DateTime(2016, 6, 18),
112     ↪ sSystem.GetProductByID(11, ReadFromCsv.ProductList).Price,
113     ↪ ReadFromCsv.UserList));
114     TransactionList.Add(new
115     ↪ InsertCashTransaction(sSystem.GetUserByUsername("inielsen",
116     ↪ ReadFromCsv.UserList), new DateTime(2016, 8, 18), 4004,
117     ↪ ReadFromCsv.UserList));
118     TransactionList.Add(new
119     ↪ BuyTransaction(sSystem.GetUserByUsername("inielsen",
120     ↪ ReadFromCsv.UserList), sSystem.GetProductByID(15,
121     ↪ ReadFromCsv.ProductList), DateTime.Now,
122     ↪ sSystem.GetProductByID(11, ReadFromCsv.ProductList).Price,
123     ↪ ReadFromCsv.UserList));
124
125     // Execute them to create a transaction file
126     foreach (var element in ReadFromCsv.TransactionList)

```

```

103         element.Execute();
104     }
105 }
106 }
107
108 }

```

9 SeasonalProduct.cs

```

1  using System;
2
3  namespace Eksamensopgave2017
4  {
5      public class SeasonalProduct : Product
6      {
7          public DateTime SeasonStartDate;
8          public DateTime SeasonEndDate;
9
10         public SeasonalProduct(string name, int active, int canbeboughtoncredit,
11             ↪ decimal price, int startYear, int startMonth, int startDay, int
12             ↪ endYear, int endMonth, int endDay) : base(name, active,
13             ↪ canbeboughtoncredit, price)
14         {
15             if (name == null || name == "")
16                 throw new ArgumentNullException();
17             else
18                 this.Name = name;
19
20             this.SeasonStartDate = new DateTime(startYear, startMonth, startDay);
21             this.SeasonEndDate = new DateTime(endYear, endMonth, endDay);
22             //Check if the product should be active or not
23             if (this.SeasonStartDate < DateTime.Now && this.SeasonEndDate >
24                 ↪ DateTime.Now)
25                 this.Active = true;
26             else
27                 this.Active = false;
28
29             if (canbeboughtoncredit == 1)
30                 this.CanBeBoughtOnCredit = true;
31
32             this.Price = price;
33         }
34     }
35 }

```

10 Stregsystem.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4
5  namespace Eksamensopgave2017
6  {
7      public class Stregsystem : IStregsystem
8      {
9          StregsystemCLI CLI = new StregsystemCLI();
10
11          public IEnumerable<Product> ActiveProducts
12          {
13              get
14              {
15                  //Combine the to product lists after having checked if the season
16                  ↪ products should be activated or deactivated
17                  //It runs after every command, so this will keep them up to date
18                  List<Product> productList = new List<Product>();
19                  foreach (SeasonalProduct element in
20                      ↪ ReadFromCsv.SeasonalProductList)
21                  {
22                      if(element.SeasonStartDate < DateTime.Now &&
23                      ↪ element.SeasonEndDate > DateTime.Now)
24                      {
25                          element.Active = true;
26                      }
27                      else
28                      {
29                          element.Active = false;
30                      }
31                  }
32
33                  foreach(var element in ReadFromCsv.SeasonalProductList.FindAll(x
34                      ↪ => x.Active == true)) productList.Add(element);
35                  foreach(var element in ReadFromCsv.ProductList.FindAll(x =>
36                      ↪ x.Active == true)) productList.Add(element);
37                  //http://stackoverflow.com/questions/3386767/linq-orderby-query
38                  IEnumerable<Product> AktiveProductList = productList.OrderBy(x =>
39                      ↪ x.ToString()).ToList();
40                  return AktiveProductList;
41              }
42          }
43
44          public InsertCashTransaction AddCreditsToAccount(User user, decimal
45              ↪ amount, List<User> UserList)
```

```

39     {
40         InsertCashTransaction Transaction = new InsertCashTransaction(user,
41             ↪ DateTime.Now, amount, UserList);
42         return Transaction;
43     }
44     public BuyTransaction BuyProduct(User user, Product product, List<User>
45         ↪ UserList, List<Product> ProductList)
46     {
47         BuyTransaction Transaction = new BuyTransaction(user, product,
48             ↪ DateTime.Now, product.Price, UserList);
49         return Transaction;
50     }
51     public Product GetProductByID(int productID, List<Product> ProductList)
52     {
53         //Check if the product exists. If it does check if it is active
54         if(ProductList.Exists(x => x.Id == productID))
55         {
56             Product product = ProductList.Find(x => x.Id == productID);
57             if (product.Active)
58             {
59                 return product;
60             }
61             else
62             {
63                 throw new InactiveProductException();
64             }
65         }
66         else
67         {
68             throw new ProductDoesNotExistException();
69         }
70     }
71 }
72
73 public IEnumerable<Transaction> GetTransactions(User user, int count)
74 {
75     IEnumerable<Transaction> transactions =
76         ↪ ReadFromCsv.TransactionList.Where(x => x.user ==
77         ↪ user).OrderByDescending(x => x.Date).Take(count);
78     return transactions;
79 }
80
81 public User GetUserByUsername(string username, List<User> UserList)
82 {

```

```

82         if (UserList.Exists(x => x.Username == username))
83         {
84             User user = UserList.Find(x => x.Username == username);
85             return user;
86         }
87         else
88         {
89             throw new InvalidUsernameException();
90         }
91     }
92
93     public User GetUser(Func<User, bool> predicate)
94     {
95         throw new NotImplementedException();
96     } //Not Done Yet
97 }
98
99
100 }

```

11 StregsystemCLI.cs

```

1  using System;
2
3  namespace Eksamensopgave2017
4  {
5
6      public class StregsystemCLI : IStregsystemUI
7      {
8          public void Close()
9          {
10             _running = false;
11         }
12
13         public void DisplayAdminCommandNotFoundMessage(string adminCommand)
14         {
15             Console.WriteLine($"The admin command '{adminCommand}' could not be
16                 ↳ found");
17         }
18
19         public void DisplayGeneralError(string errorString)
20         {
21             throw new NotImplementedException();
22         }
23     }
24 }

```

```

23     public void DisplayInactiveProduct(Product product)
24     {
25         Console.WriteLine($"'{product.Name}' is inactive (ID:
        ↳ {product.Id}");
26     }
27
28     public void DisplayInsufficientCash(User user, Product product)
29     {
30         Console.WriteLine($"{user.Firstname} {user.Lastname} does not have
        ↳ sufficient cash to buy {product.Name} \n{user.Firstname} have a
        ↳ balance of {user.Balance}");
31     }
32
33     public void DisplayProductNotFound(string product)
34     {
35         Console.WriteLine($"Product not found");
36     }
37
38     public void DisplayTooManyArgumentsError(string command)
39     {
40         Console.WriteLine($"The following command contans to many arguments:
        ↳ {command}");
41     }
42
43     public void DisplayTooFewArgumentsError()
44     {
45         Console.WriteLine($"To few arguments");
46     }
47
48     public void DisplayUserBuysProduct(BuyTransaction transaction)
49     {
50         Console.WriteLine($"{transaction.user.Firstname} bought a
        ↳ product\nUser balance: {transaction.user.Balance}");
51     }
52
53     public void DisplayUserInfo(User user)
54     {
55         Stregsystem sSystem = new Stregsystem();
56         //http://stackoverflow.com/a/5344836
57         //http://stackoverflow.com/a/10883477
58         Console.WriteLine($"{user.Username}, {user.Firstname}
        ↳ {user.Firstname}, balance: {user.Balance}");
59         foreach (var element in sSystem.GetTransactions(user, 10))
60         {
61             Console.WriteLine(element);
62         }
63     }

```

```

64
65     public void DisplayUserNotFound(string username)
66     {
67         Console.WriteLine($"User {username} does not exist");
68     }
69
70     public void DisplayAktiveProducts()
71     {
72         Console.Clear();
73         Stregsystem ssystem = new Stregsystem();
74         foreach (var Item in ssystem.ActiveProducts)
75         {
76             Console.Write(Item.ToString());
77         }
78         Console.WriteLine();
79     }
80     private bool _running;
81     public void Start()
82     {
83
84         StregsystemController ssc = new StregsystemController();
85         _running = true;
86         DisplayAktiveProducts();
87         Console.Write("\nQuickbuy: ");
88         do
89         {
90             string command = Console.ReadLine();
91             DisplayAktiveProducts();
92             if (command == ":q" || command == ":quit")
93                 Close();
94             ssc.RunCommand(command);
95             Console.Write("\n\nQuickbuy: ");
96
97         } while (_running);
98     }
99
100     public void DisplayUserBuysProduct(int count, BuyTransaction transaction)
101     {
102         Console.WriteLine($"{transaction.user.Firstname} bought {count}
103         ↪ products \nUser balance: {transaction.user.Balance}");
104     }
105
106     public void LowBalanceWarning(User user, decimal amount)
107     {
108         if (user.Balance < amount) Console.WriteLine("!!---> YOUR BALANCE IS
109         ↪ UNDER 50 <---!!");
110     }
111 }

```



```
110 }
111 }
```

12 StregsystemController.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text.RegularExpressions;
5
6 namespace Eksamensopgave2017
7 {
8     class StregsystemController
9     {
10
11         ↪ //http://stackoverflow.com/questions/2829873/how-can-i-detect-if-this-dictionary-key-exists-in-c
12         ↪ //https://www.tutorialspoint.com/csharp/csharp_delegates.htm
13         ↪ //https://msdn.microsoft.com/en-us/library/bb882516.aspx
14
15         ↪ //http://stackoverflow.com/questions/4233536/c-sharp-store-functions-in-a-dictionary
16
17         ↪ //http://stackoverflow.com/questions/2896715/dictionary-with-delegate-or-switch
18         ↪ //http://stackoverflow.com/a/21099511
19
20         delegate List<Product> StatusCommands(int productId);
21         delegate void QuitCommand();
22         delegate List<Product> CreditCommands(int productId);
23         delegate InsertCashTransaction AddCreditsCommands(User user, decimal
24         ↪ amount, List<User> UserList);
25
26         StregsystemCLI CLI = new StregsystemCLI();
27         Stregsystem sSystem = new Stregsystem();
28
29         public Dictionary<string, Delegate> _adminCommands = new
30         ↪ Dictionary<string, Delegate>();
31
32         public static List<Product> Aktivat(int productId)
33         {
34             StregsystemCLI CLI = new StregsystemCLI();
35             FileWriters writer = new FileWriters();
36             ReadFromCsv.ProductList.Where(x => x.Id ==
37             ↪ productId).ToList().ForEach(x => x.Active = true);
38             writer.WriteToProductCsv("../products.csv",
39             ↪ ReadFromCsv.ProductList);
40             CLI.DisplayAktiveProducts();
41         }
42     }
43 }
```

```

34         return null;
35     }
36
37     public static List<Product> Deactivate(int productId)
38     {
39         StregsystemCLI CLI = new StregsystemCLI();
40         FileWriters writer = new FileWriters();
41         ReadFromCsv.ProductList.Where(x => x.Id ==
42             ↳ productId).ToList().ForEach(x => x.Active = false);
43         writer.WriteToProductCsv("../products.csv",
44             ↳ ReadFromCsv.ProductList);
45         CLI.DisplayAktiveProducts();
46         return null;
47     }
48
49     public static List<Product> CreaditOn(int productId)
50     {
51         FileWriters writer = new FileWriters();
52         ReadFromCsv.ProductList.Where(x => x.Id ==
53             ↳ productId).ToList().ForEach(x => x.CanBeBoughtOnCredit = true);
54         writer.WriteToProductCsv("../products.csv",
55             ↳ ReadFromCsv.ProductList);
56         return null;
57     }
58
59     public static List<Product> CreditOff(int productId)
60     {
61         FileWriters writer = new FileWriters();
62         ReadFromCsv.ProductList.Where(x => x.Id ==
63             ↳ productId).ToList().ForEach(x => x.CanBeBoughtOnCredit = false);
64         writer.WriteToProductCsv("../products.csv",
65             ↳ ReadFromCsv.ProductList);
66         return null;
67     }
68
69     public StregsystemController()
70     {
71         StatusCommands MakeAktive = new StatusCommands(Aktivate);
72         StatusCommands MakeDeactivated = new StatusCommands(Deactivate);
73         QuitCommand Quit = new QuitCommand(CLI.Close);
74         CreditCommands MakeCreditOn = new CreditCommands(CreaditOn);
75         CreditCommands MakeCreditOff = new CreditCommands(CreditOff);
76         AddCreditsCommands AddCredits = new
77             ↳ AddCreditsCommands(sSystem.AddCreditsToAccount);
78
79         _adminCommands.Add(":activate", MakeAktive);
80         _adminCommands.Add(":deactivate", MakeDeactivated);
81         _adminCommands.Add(":quit", Quit);

```

```

75         _adminCommands.Add(":q", Quit);
76         _adminCommands.Add(":crediton", MakeCreditOn);
77         _adminCommands.Add(":creditoff", MakeCreditOff);
78         _adminCommands.Add(":addcredits", AddCredits);
79     }
80
81     public void RunCommand(string command)
82     {
83         if (command == "")
84             CLI.DisplayTooFewArgumentsError();
85         else if (Convert.ToString(command[0]) == ":") //If it starts with :
86             ↪ we know it is a admin command
87         {
88             string[] SplittedString = command.Split(' '); //Split it so we
89             ↪ can see which command and use the info
90             string AdminCommand = Convert.ToString(SplittedString[0]);
91             if (_adminCommands.ContainsKey(AdminCommand));
92             {
93                 if (AdminCommand == ":q" || AdminCommand == ":quit")
94                 {
95                     _adminCommands["q"].DynamicInvoke();
96                 }
97                 else if (AdminCommand == ":activate" || AdminCommand ==
98                     ↪ ":deactivate" || AdminCommand == ":crediton" ||
99                     ↪ AdminCommand == ":creditoff")
100                 {
101                     int productId = Convert.ToInt32(SplittedString[1]);
102                     _adminCommands[AdminCommand].DynamicInvoke(productId);
103                 }
104                 else if (AdminCommand == ":addcredits")
105                 {
106                     string username = Convert.ToString(SplittedString[1]);
107                     User user = sSystem.GetUserByUsername(username,
108                         ↪ ReadFromCsv.UserList);
109                     decimal amount = Convert.ToDecimal(SplittedString[2]);
110                     sSystem.AddCreditsToAccount(user, amount,
111                         ↪ ReadFromCsv.UserList).Execute();
112                 }
113                 else
114                 {
115                     CLI.DisplayAdminCommandNotFoundMessage(command);
116                 }
117             }
118         }
119     }
120
121     //Setup Regex to notice the tree different kinds of commands

```

```

116 Match username = Regex.Match(command, @"^[a-z\d_-]+$"); //Only
    ↳ username
117 Match buy = Regex.Match(command, @"^[a-z\d_-]+ \d+$"); //username
    ↳ numbers
118 Match multibuy = Regex.Match(command, @"^[a-z\d_-]+ \d+ \d+$");
    ↳ //username numbers numbers
119
120 if (username.Success)
121 {
122     try
123     {
124         User user = sSystem.GetUserByUsername(command,
            ↳ ReadFromCsv.UserList);
125         CLI.DisplayUserInfo(user);
126     }
127     catch (InvalidUsernameException)
128     {
129         CLI.DisplayUserNotFound(command);
130     }
131 }
132 else if (buy.Success)
133 {
134     string[] buyArray = command.Split(' '); //Split the command
        ↳ to get the information
135     try
136     {
137         User user =
            ↳ sSystem.GetUserByUsername(Convert.ToString(buyArray[0]),
            ↳ ReadFromCsv.UserList);
138         Product product =
            ↳ sSystem.GetProductByID(Convert.ToInt32(buyArray[1]),
            ↳ ReadFromCsv.ProductList);
139         if (user.Balance >= product.Price ||
            ↳ product.CanBeBoughtOnCredit)
140         {
141             sSystem.BuyProduct(user, product,
                ↳ ReadFromCsv.UserList,
                ↳ ReadFromCsv.ProductList).Execute();
142             CLI.DisplayUserBuysProduct(sSystem.BuyProduct(user,
                ↳ product, ReadFromCsv.UserList,
                ↳ ReadFromCsv.ProductList));
143             CLI.LowBalanceWarning(user, 50); // Check for low
                ↳ balance
144         }
145     else
146     {
147         CLI.DisplayInsufficientCash(user, product);
148     }

```

```

149     }
150     catch (InvalidUsernameException) {
151         ↪ CLI.DisplayUserNotFound(Convert.ToString(buyArray[0]));
152         ↪ }
153     catch (InactiveProductException)
154     {
155         ↪ //Create a list of all inactive products to finde the
156         ↪ specifc product by id
157         List<Product> productList = new List<Product>();
158         foreach (var element in
159             ↪ ReadFromCsv.SeasonalProductList.FindAll(x =>
160             ↪ x.Active == false)) productList.Add(element);
161         foreach (var element in ReadFromCsv.ProductList.FindAll(x
162             ↪ => x.Active == false)) productList.Add(element);
163         Product product = productList.Find(x => x.Id ==
164             ↪ Convert.ToInt32(buyArray[1]));
165         CLI.DisplayInactiveProduct(product);
166     }
167     catch (ProductDoesNotExistException) {
168         ↪ CLI.DisplayProductNotFound(null); }
169 }
170 else if (multibuy.Success)
171 {
172     string[] buyArray = command.Split(' ');
173     try
174     {
175         User user =
176             ↪ sSystem.GetUserByUsername(Convert.ToString(buyArray[0]),
177             ↪ ReadFromCsv.UserList);
178         int productCount = Convert.ToInt32(buyArray[1]);
179         Product product =
180             ↪ sSystem.GetProductByID(Convert.ToInt32(buyArray[2]),
181             ↪ ReadFromCsv.ProductList);
182         decimal totalPrice = productCount * product.Price;
183         if ((user.Balance >= totalPrice ||
184             ↪ product.CanBeBoughtOnCredit))
185         {
186             for (int i = 0; i < productCount; i++) // Run the
187                 ↪ command once for each bought product
188                 sSystem.BuyProduct(user, product,
189                     ↪ ReadFromCsv.UserList,
190                     ↪ ReadFromCsv.ProductList).Execute();
191             CLI.DisplayUserBuysProduct(productCount,
192                 ↪ sSystem.BuyProduct(user, product,
193                 ↪ ReadFromCsv.UserList, ReadFromCsv.ProductList));
194             CLI.LowBalanceWarning(user, 50);
195         }
196     }
197     else

```

```

179         {
180             CLI.DisplayInsufficientCash(user, product);
181         }
182     }
183
184     ↪ //http://stackoverflow.com/questions/136035/catch-multiple-exceptions-at-once
185     catch (InvalidUsernameException) {
186         ↪ CLI.DisplayUserNotFound(Convert.ToString(buyArray[0]));
187         ↪ }
188     catch (InactiveProductException)
189     {
190         List<Product> productList = new List<Product>();
191         foreach (var element in
192             ↪ ReadFromCsv.SeasonalProductList.FindAll(x =>
193             ↪ x.Active == false)) productList.Add(element);
194         foreach (var element in ReadFromCsv.ProductList.FindAll(x
195             ↪ => x.Active == false)) productList.Add(element);
196         Product product = productList.Find(x => x.Id ==
197             ↪ Convert.ToInt32(buyArray[2]));
198         CLI.DisplayInactiveProduct(product);
199     }
200     catch (ProductDoesNotExistException) {
201         ↪ CLI.DisplayProductNotFound(null); }
202 }
203 else
204 {
205     CLI.DisplayTooManyArgumentsError(command);
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

13 Transaction.cs

```
1 using System;
2
3 namespace Eksamensopgave2017
4 {
5     public class Transaction
6     {
7         public Transaction(User user, DateTime date, decimal amount)
8         {
9             if (user == null)
10                 throw new ArgumentNullException();
11             else
12                 this.user = user;
13             this.Date = date;
14             this.Amount = amount;
15         }
16
17         public readonly int Id = _nextID++;
18         private static int _nextID = 1;
19         public User user;
20         public DateTime Date;
21         public decimal Amount;
22
23         public override string ToString()
24         {
25             return $"New transaction ({this.Id}): {this.user.Firstname}
26                 ↳ {this.user.Lastname} - DKK {this.Amount} - [{this.Date}];
27         }
28
29         public virtual void Execute()
30         {
31         }
32     }
33 }
34 }
```

14 User.cs

```
1 using System;
2 using System.Text.RegularExpressions;
3
4 namespace Eksamensopgave2017
5 {
6     public class User : IComparable<User>
7     {
8         //https://msdn.microsoft.com/en-us/library/twcw2f1c(v=vs.110).aspx
9         Regex UsernameValidator = new Regex(@"^[a-z\d_]+$");
10        Regex EmailLocalValidator = new Regex(@"^[a-zA-Z\d\-\.]+$");
11        Regex EmailDomainValidator = new Regex(@"^[a-zA-Z\d\-\.]+$");
12        public User(string firstname, string lastname, string username, string
13        ↪ email, decimal balance)
14        {
15            string ConvertedUsername = username.Replace("\", "");
16            ↪ //http://stackoverflow.com/a/1177897
17            Match mUsername = UsernameValidator.Match(ConvertedUsername);
18            if (mUsername.Success)
19                this.Username = ConvertedUsername;
20            else
21                throw new InvalidUsernameException();
22
23            string[] SplittedEmail = email.Split('@'); // Split it to find local
24            ↪ and domain
25            Match mLocal = EmailLocalValidator.Match(SplittedEmail[0]);
26            Match mDomain = EmailDomainValidator.Match(SplittedEmail[1]);
27            int DomainLength = SplittedEmail[1].Length - 1;
28            //If both the Regex validators is okay and the char check does not
29            ↪ match we assign email
30            if (mLocal.Success && mDomain.Success && SplittedEmail[1][0] !=
31            ↪ Convert.ToChar(".") && SplittedEmail[1][0] !=
32            ↪ Convert.ToChar("-") && SplittedEmail[1][DomainLength] !=
33            ↪ Convert.ToChar(".") && SplittedEmail[1][DomainLength] !=
34            ↪ Convert.ToChar("-"))
35                this.Email = email;
36            else
37                throw new InvalidEmailException();
38
39            this.Balance = balance;
40
41            if (firstname == null || firstname == "" || lastname == null ||
42            ↪ lastname == "")
43                throw new ArgumentNullException();
44            else
45            {
46
```



```

37         this.Firstname = firstname;
38         this.Lastname = lastname;
39     }
40
41 }
42
43 public readonly int Id = _nextID++;
44 private static int _nextID = 1;
45 public string Firstname;
46 public string Lastname;
47 public string Username;
48 public string Email;
49 public decimal Balance;
50
51
52 ↪ //https://msdn.microsoft.com/en-us/library/system.object.tostring(v=vs.110).aspx
53 public override string ToString()
54 {
55     return $"{this.Firstname.ToString()} {this.Lastname.ToString()}\n";
56 }
57
58 public override bool Equals(object obj)
59 {
60     User other = obj as User;
61     if (other == null)
62         return false;
63     return Id.Equals(other.Id);
64 }
65
66 public override int GetHashCode()
67 {
68     return base.GetHashCode();
69 }
70
71 public int CompareTo(User other)
72 {
73     return this.Id.CompareTo(other.Id);
74 }
75 }
76
77 }

```

15 WriteToCsv.cs

```
1 using System.Collections.Generic;
2 using System.IO;
3 using System.Text;
4
5 namespace Eksamensopgave2017 //
6     ↳ https://www.codeproject.com/Articles/415732/Reading-and-Writing-CSV-Files-in-Csharp
7 {
8     public class CsvRow : List<string>
9     {
10         public string LineText { get; set; }
11     }
12
13     public class CsvFileWriter : StreamWriter
14     {
15         public CsvFileWriter(Stream stream) : base(stream)
16         {
17         }
18
19         public CsvFileWriter(string filename) : base(filename)
20         {
21         }
22
23         public void WriteRow(CsvRow row)
24         {
25             StringBuilder builder = new StringBuilder();
26             foreach (string value in row)
27             {
28                 builder.Append(value);
29             }
30             row.LineText = builder.ToString();
31             WriteLine(row.LineText);
32         }
33     }
34 }
35 }
```
