ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

КУРСОВОЙ ПРОЕКТ

Дисциплина: Объектно-ориентированное программирование

Тема: Разработка консольного приложения "Магазин по продаже вещей", имитирующего работу онлайн магазина

Выполнил(а): студент(ка) группы 221-373

Белявский Н.К.

(Фамилия И.О.)

Дата, подпись _____ _____

(Дата) (Подпись)

Проверил: _____

(Фамилия И.О., степень, звание)

Дата, подпись _____ _____

(Дата) (Подпись)

Замечания:

Москва

2023

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

УТВЕРЖДАЮ
Зав. каф., к.т.н. Е.В. Булатников
«__»_____2023 г.
___
(Подпись)

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

Студент Белявский Николай Константинович

группа 221-373

(Фамилия, Имя, Отчество)

Тема Разработка консольного приложения "Магазин по продаже вещей", имитирующего работу онлайн магазина

1. **Срок представления работы к защите** «22» декабря 2023 г.
2. Исходные данные для выполнения работы: исходные материалы
3. Содержание курсового проекта: введение, аналитическая часть, практическая часть, заключение, библиографический список
4. Перечень графического материала рисунки, листинги
5. Литература и прочие материалы, рекомендуемые студенту для изучения: указаны в библиографическом списке
6. Дата выдачи задания: « » 2023 г.
7. Руководитель: /Лазарева О.Ю./Москва
8. Задание к исполнению принял: Белявский Н.К. / _ФИО_ /

Оглавление

1. Введение

В современном мире электронной коммерции стремительно развевается сфера онлайн магазинов. В связи с этим, создание программных приложений, способных эмулировать функционал таких магазинов, является актуальной задачей, которая позволяет научиться создавать полноценные проекты. Данная курсовая работа посвящена разработке консольного приложения на языке программирования C#, предназначенного для имитации работы онлайн-магазина. Проектирование и реализация такого приложения позволит глубже понять основные принципы объектно-ориентированного программирования (ООП) и их применение в создании сложных систем.

Цели и задачи

Целью данной курсовой работы является разработка консольного приложения "Магазин по продаже вещей", способного эмулировать базовый функционал онлайн-магазина.

Задачи

Проектирование архитектуры приложения с использованием принципов ООП.

Создание классов, отражающих основные сущности: товары, пользователи.

Реализация механизма взаимодействия с пользователем через консольный интерфейс.

Работа с системой сохранения для хранения и обработки информации о товарах и пользователях.

Реализация взаимодействия с программой разных типов пользователей: админ, работник, покупатель.

2. Анализ требований

Программа должна запускаться на стационарных компьютерах без доступа к интернету для симуляции работы онлайн магазина. Принцип работы аналогичен принципу работы с обычным онлайн магазином: пользователь авторизируется и находит понравившиеся товары, добавляет их в корзину и оформляет заказ.

3. Требования

Интерфейс должен быть читаемым и функциональным.

Программа должна получать имя и пароль пользователя.

Программа должна предоставить возможность покупателю искать товары и покупать их.

Программа должна предоставить возможность работнику добавлять и удалять товары.

Программа должна предоставить возможность админу добавлять и увольнять работников.

В программе должна быть реализована возможность сохранения данных при закрытии и их загрузка при запуске.

Код программы должен учитывать принципы ООП, SOLID.

Код должен быть написан с использованием принятых конвенций и стандартов на языке C#:

- Отступы: 4 пробела

- Именование: CamelCase для методов и свойств, PascalCase для классов

-Комментарии.

4. Проектирование

4.1 Классы и взаимодействие между ними

При проектировании программной части приложения необходимо было учитывать требования. Необходимо помнить, что пользователь имеет атрибуты: ID, имя, электронную почту, пароль, уровень доступа и корзину. Админ, работник и покупатель не должны отличаться ничем, кроме как наличием корзины у покупателя. Для этого нужно создать абстрактный класс пользователь и 3 класса: покупатель, работник и админ, которые будут отличаться уровнем доступа, который мы будем задавать в отдельную переменную << AccessKey>>

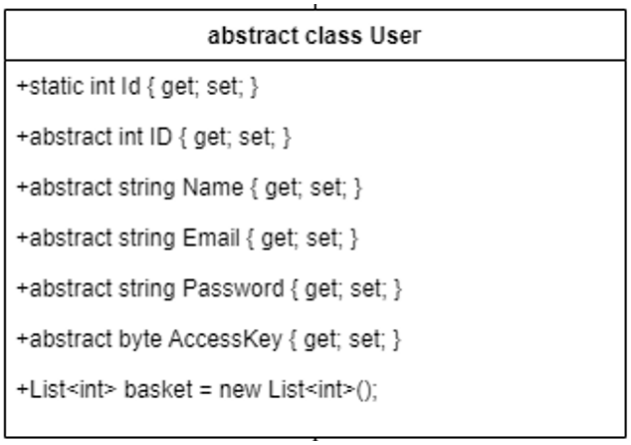

Рисунок 1 - abstract class User

public static int Id { get; set; } – поле для хранения id следующего пользователя

public abstract int ID { get; set; } – поле для хранения id пользователя

public abstract string Name { get; set; } – поле для хранения имени пользователя

public abstract string Email { get; set; } – поле для хранения почты пользователя

public abstract string Password { get; set; } – поле для хранения пароля пользователя

public abstract byte AccessKey { get; set; } – поле для хранения уровня доступа

public List<int> basket = new List<int>(); - список для хранения добавленных в корзину товаров

Теперь нужно создать наследуемые классы , которые будут представлять покупателя, адмика и рабочего:

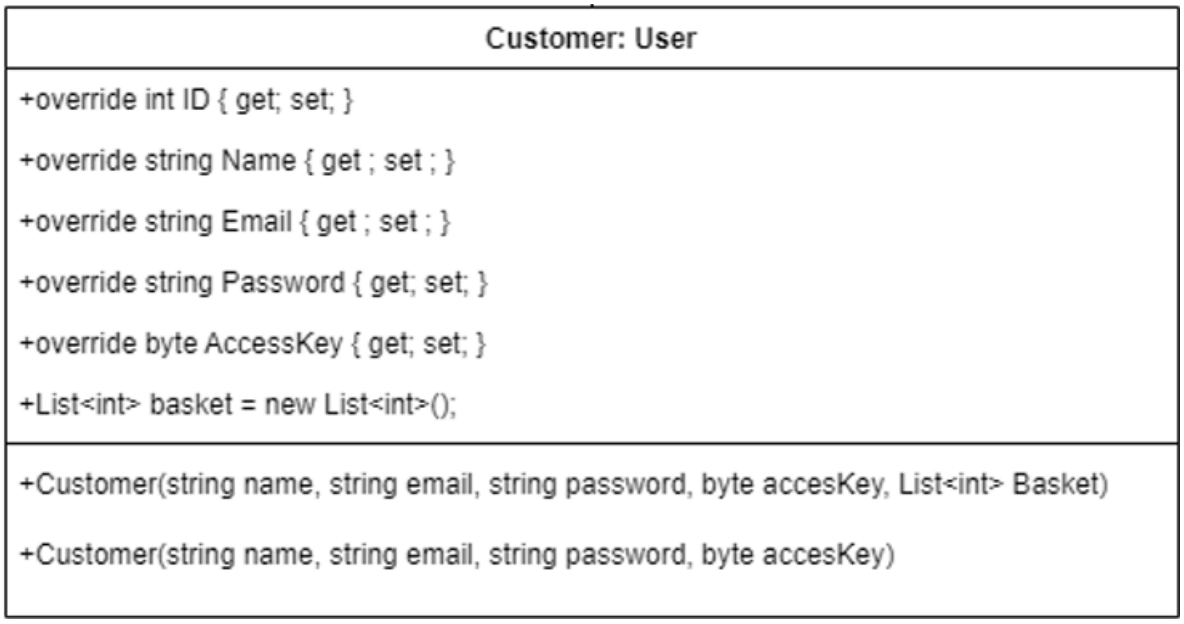

Рисунок 2 - Customer

public Customer(string name, string email, string password, byte accesKey, List<int> Basket) – конструктор для создания пользователя с корзиной, используется для загрузки пользователей.

public Customer(string name, string email, string password, byte accesKey) – конструктор для первичного создания пользователя.

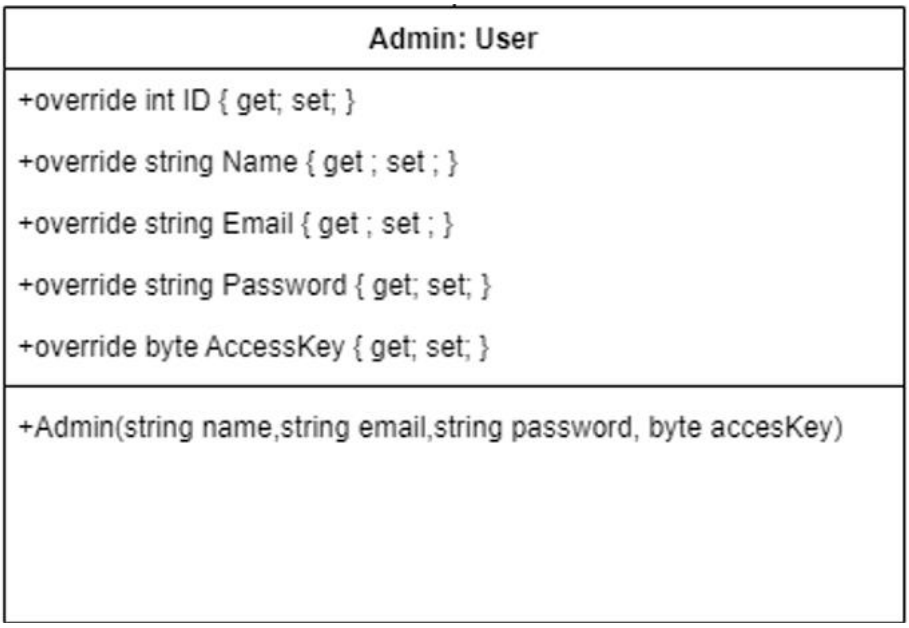

Рисунок 3 - Admin

public Admin(string name,string email,string password, byte accesKey) – конструктор для создания админа с нужными параметрами

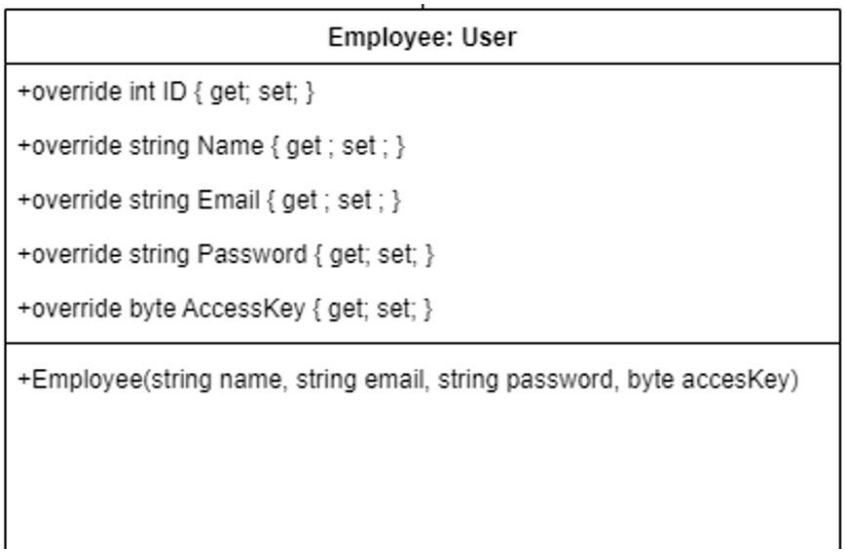

Рисунок 4 - Employee

public Employee(string name, string email, string password, byte accesKey) – конструктор для создание работника с нужными данными

Нужно создать класс «Товар», который имеет имя, цвет, тип, размер, цену и ID а также он должен выводить о себе информацию.

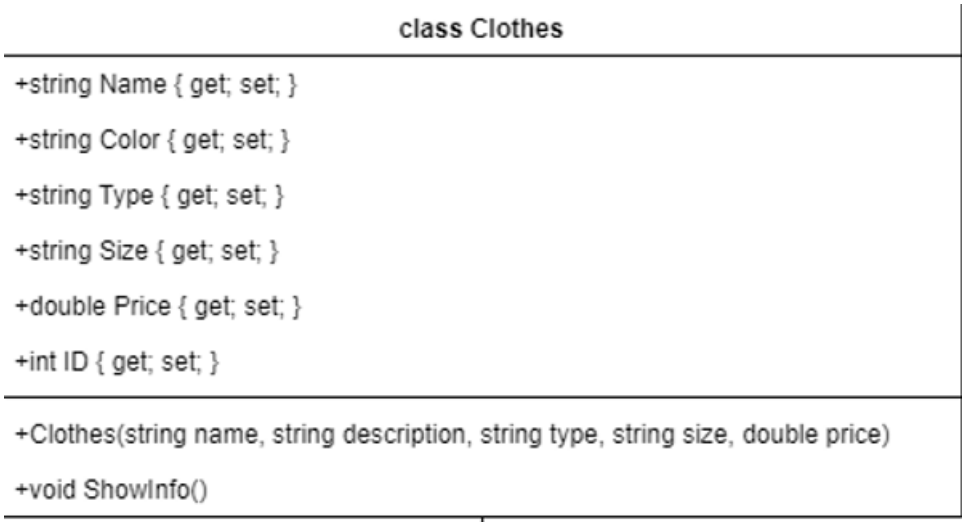

Рисунок 5 - Clothes

public string Name { get; set; } – поле для хранения названия товара

public string Color { get; set; } – поле для хранения цвета товара

public string Type { get; set; } – поле для хранения типа товара

public string Size { get; set; } – поле для хранения размера товара

public double Price { get; set; } – поле для хранения цены товара

public int ID { get; set; } – поле для хранения id товара

public Clothes(string name, string description, string type, string size, double price) – конструктор для создания товара с заданными параметрами

public void ShowInfo() – метод для вывода информации о товаре

Нужно создать класс «Data» для реализации сохранения и загрузки данных о пользователях и товарах. Класс должен иметь поля с названием файла сохранённых пользователей и с названием файла сохранённых товаров. Также класс должен иметь метода сохранения и загрузки данных

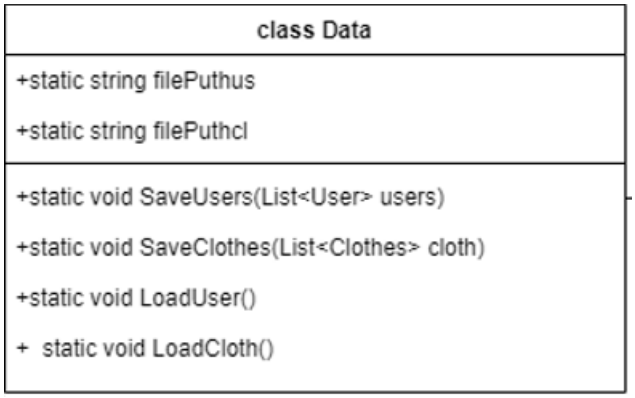

Рисунок 6 - Data

private static string filePathus – переменная для хранения названия файла с сохранёнными пользователями

private static string filePathcl - переменная для хранения названия файла с сохранёнными товарами

public static void SaveUsers(List<User> users) – метод для сохранения списка пользователей

public static void SaveClothes(List<Clothes> cloth) - метод для сохранения списка товаров

public static void LoadUser() – метод для загрузки списка пользователей

public static void LoadCloth() - метод для загрузки списка товаров

Нужно создать класс «AllUser» для хранения всех пользователей и методы для проведения операций, относящихся к ним. Класс должен иметь список со всеми пользователями, а также методы, которые удаляют, добавляют и ищут пользователей

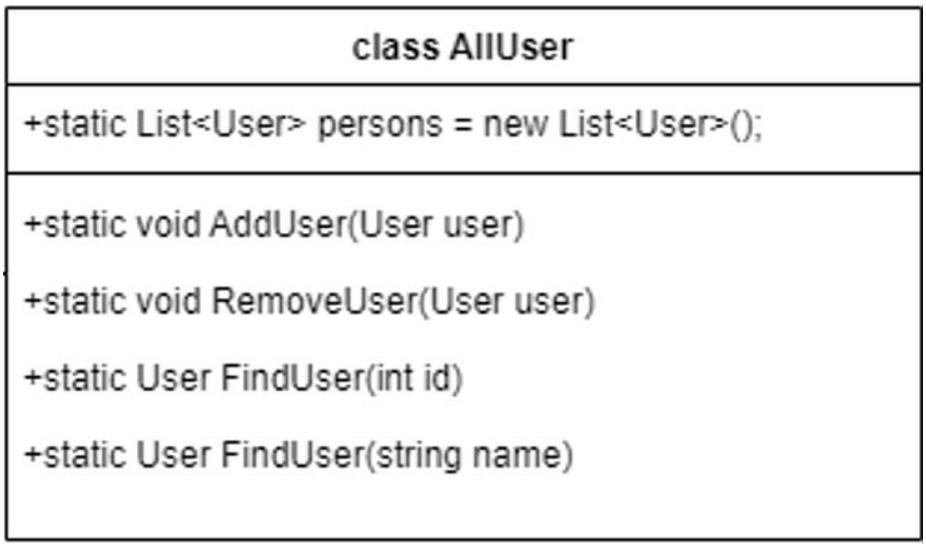

Рисунок 7 - AllUser

public static List<User> persons – список с данными всех пользователей

public static void AddUser(User user) – метод для добавления пользователя в список

public static void RemoveUser(User user) – метод для удаления пользователя из списка

public static User FindUser(int id) – метод для поиска пользователя по id

public static User FindUser(string name) – метод для поиска пользователя по имени

Нужно создать класс «AllClothes» для хранения всех товаров и методы для проведения операций, относящихся к ним. Класс должен иметь список со всеми товарами, а также методы, которые удаляют, добавляют и ищут товары, а также выводят информацию о всех товарах.

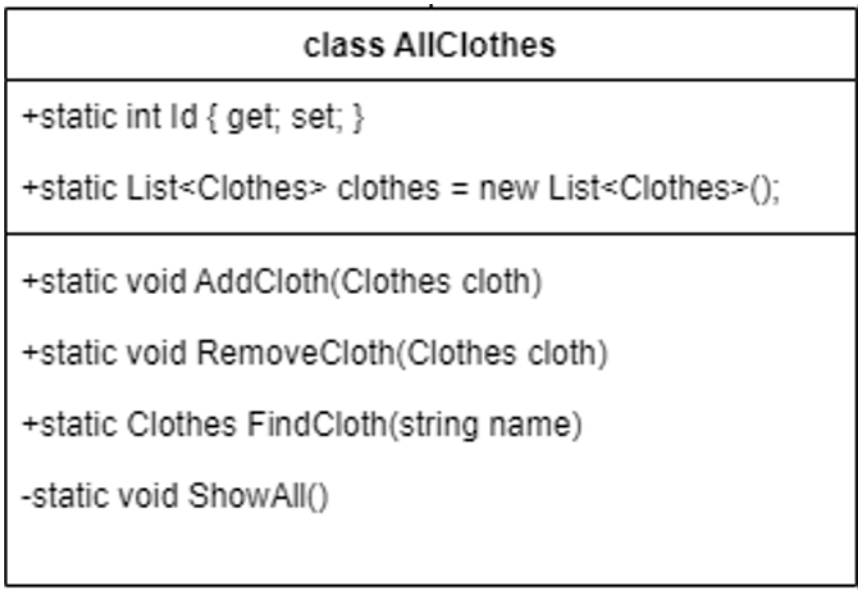

Рисунок 8 - AllClothes

public static int Id { get; set; } – поле для хранения id следующего товара

public static List<Clothes> clothes – список с данными всех товаров

public static void AddCloth(Clothes cloth) – метод для добавления товара в список

public static void RemoveCloth(Clothes cloth) – метод для удаления товара из списка

public static Clothes FindCloth(string name) – метод для поиска товара по имени

public static void ShowAll() – метод для вывода информации о всех товаров на консоль

Класс программы «Program» должен содержать переменную для обозначения текущего пользователя и методы для авторизации и регистрации, а также метод Main, в котором будет производится основное взаимодействие с пользователем.

Рисунок 9 - Cloth Store

public static string selectname – переменная для хранения имени текущего пользователя

User sUser – переменная для хранения текущего пользователя

public static void Enter() – метод для авторизации

public static void Registration() – метод для регистрации

static void Main(string[] args) – основной метод, в котором производится взаимодействие с пользователем

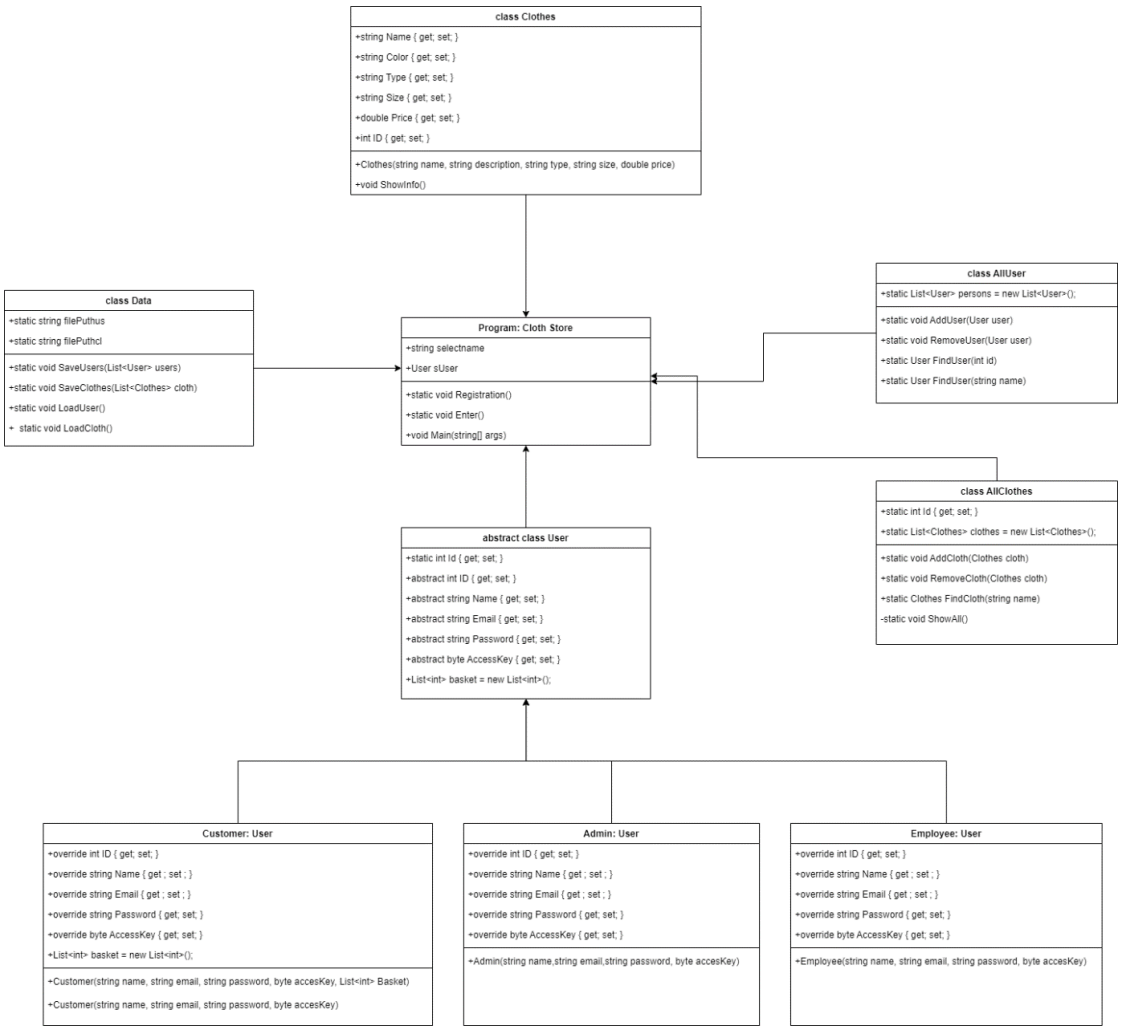

Рисунок 10 - UML диаграмма

5 Реализация

5.1 Функциональность классов и их взаимодействие

Реализация функциональности разрабатывалась на основе выдвинутых требований и результатов проектирования.

Был создан абстрактный класс «User», который стал родительским для классов «Customer», «Admin», «Employee» с такими свойствами

Листинг 5.1.1 поля класса User

```
abstract class User
 {
     public static int Id { get; set; }
     public abstract int ID { get; set;
}
     public abstract string Name { get;
set; }
     public abstract string Email {
get; set; }
     public abstract string Password {
get; set; }
     public abstract byte AccessKey {
get; set; }
     public List<int> basket = new
List<int>();
 }
    }
```

Далее были созданы дочерние классы:

Все классы были созданы с такими свойствами:

Листинг 5.1.2 свойства класса User

```
public override int ID { get; set; }
public override string Name { get ; set
; }
public override string Email { get ;
set ; }
public override string Password { get;
set; }
public override byte AccessKey { get;
set; }
```

«Customer» имел такой конструктор для определения свойств объекта:

Листинг 5.1.3 поля и конструктор класса Customer:

```
public Customer(string name, string
email, string password, byte accesKey,
List<int> Basket)
{
    ID = User.Id;
    User.Id++;
```

```
    basket = Basket;
    AccessKey = accesKey;
    Name = name;
    Email = email;
    Password = password;
    basket.Add(0);
    AllUser.AddUser(this);
}
public Customer(string name, string
email, string password, byte accesKey)
{
    ID = User.Id;
    User.Id++;
    AccessKey = accesKey;
    Name = name;
    Email = email;
    Password = password;
    basket.Add(0);
    AllUser.AddUser(this);}
```

Был использован второй конструктор для того, чтобы создавать пользователя без корзины, а первый используется для загрузки пользователей из файла.

«Admin» также имеет конструктор

Листинг 5.1.4 конструктор класса Admin:

```
public Admin(string name,string
email,string password, byte accesKey)
{
    ID = User.Id;
    User.Id++;
    AccessKey = accesKey;
    Name = name;
    Email = email;
    Password = password;
    basket.Add(0);
    AllUser.AddUser(this);
}
```

«Employee» также имеет конструктор

Листинг 5.1.5 конструктор класса Employee:

```
public Employee(string name, string
email, string password, byte accesKey)
{
    ID = User.Id;
    User.Id++;
    AccessKey = accesKey;
    Name = name;
    Email = email;
    Password = password;
    basket.Add(0);
    AllUser.AddUser(this);
}
```

В каждом конструкторе есть строчки ID = User.Id; User.Id++; Здесь мы присваиваем объекту id и увеличиваем статическое поле User.Id. Таким образом каждый пользователь получает свой уникальный номер, с помощью которого мы потом можем его найти либо загрузить данные о нём.

После этого был создан класс «Clothes», который описывал товар. Класс имеет такие свойства

Листинг 5.1.6 свойства класса Clothes:

```
public string Name { get; set; }
 public string Color { get; set; }
 public string Type { get; set; }
 public string Size { get; set; }
 public double Price { get; set; }
 public int ID { get; set; }
```

Также был создан конструктор

Листинг 5.1.7 конструктор класса Clothes:

```
public Clothes(string name, string
description, string type, string size,
double price)
{
    ID = AllClothes.Id;
    AllClothes.Id++;
    Name = name;
    Color = description;
    Type = type;
    Size = size;
    Price = price;
    AllClothes.AddCloth(this);
}
```

В нём также каждому товару присваивался уникальный номер с помощью статической переменной в классе AllClothes. Это было создано для того, чтобы корректно добавлять товары в корзину и корректно загружать и сохранять данные.

Был создан метод для вывода данных о товаре в консоль public void ShowInfo()

Был создан класс статический «AllClothes», который имел такие поля

Листинг 5.1.8 свойства класса AllClothes:

```
public static int Id { get; set;}
public static List<Clothes> clothes =
new List<Clothes>();
```

С помощью статической переменной Id мы присваиваем каждому товару свой уникальный номер, а в списке хранились объекты класса Clothes.

С помощью метода public static void AddCloth(Clothes cloth) реализовано пополнение списка новыми товарами. С помощью метода RemoveCloth(Clothes cloth) реализовано удаление товаров из списка

Я добавил метод для поиска объекта класса Clothes в списке clothes по имени

Листинг 5.1.9 метод поиска объекта класса Clothes:

```
public static Clothes FindCloth(string
name)
 {
     return clothes.Find(cloth =>
cloth.Name == name);
 }
```

Я добавил метод, который выводит информацию о всех товарах, которые есть в списке

Листинг 5.1.10 метод для вывода информации о всех товарах:

```
public static void ShowAll()
 {
     foreach(Clothes cloth in clothes)
     {
         Console.WriteLine($"Товар -
{cloth.Name}: {cloth.Type},
{cloth.Color} цвет, {cloth.Size},
{cloth.Price}\n");
     }
 }
```

Далее я создал статичный класс «AllUser», который хранит всех пользователей и производить над ними разные операции. Список пользователей:

Листинг 5.1.11 Список пользователей в классе AllUser:

```
public static List<User> persons = new
List<User>();
```

Также добавил методы для взаимодействия со списком пользователей:

Листинг 5.1.12 Методы для взаимодействия со списком пользователей:

```
public static void AddUser(User user)
////Добавление пользователя
 {
     persons.Add(user);
 }
 public static void RemoveUser(User
user)     ////Удаление пользователя
 {
     persons.Remove(user);
     Console.WriteLine($"Пользователь
{user.Name} удалён\n");
 }
 public static User FindUser(int id)
////Поиск по id
 {
     return persons.Find(user =>
User.Id == id);
 }
 public static User FindUser(string
name)     ////Поиск по имени
 {
     return persons.Find(user =>
user.Name == name);
 }
```

Первый метод получал в качестве аргумента объект класса User и добавлял его в список пользователей.

Второй метод получал в качестве аргументы объект класса User и удалял его из списка пользователя и сообщал об этом пользователю в консоли

Третий метод принимает данные об id типа int и ищет пользователя в списке

Четвёртый метод принимает данные об имени типа string и ищет пользователя в списке

Далее надо было создать статический класс, который отвечал бы за сохранение и за загрузку данных о пользователях и товарах. Я создал класс «Data» с такими полями:

Листинг 5.1.13 Переменные, которые хранят названия файлов с данными, в классе Data:

```
private static string filePathus =
"users.txt";
private static string filePathcl =
"clothes.txt";
```

Первое поле хранило название файла, в который загружаются данные о пользователях

Второе поле хранило название файла, в который загружаются данные о товарах

Рассмотрим метод, который сохранял информацию о пользователях:

Листинг 5.1.14 Метод сохранения данных пользователей:

```
public static void SaveUsers(List<User>
users)
{
    using (StreamWriter writer = new
StreamWriter(filePathus))
    {
        foreach (var user in users)
        {
            if (user.GetType().Name ==
"Customer")
            {
                string basketString =
string.Join(",", user.basket);

writer.WriteLine($"{user.Name}*{user.E
mail}*{user.Password}*{user.AccessKey}
*{basketString}");
            }
            else
            {

writer.WriteLine($"{user.Name}*{user.E
mail}*{user.Password}*{user.AccessKey}
");
            }

        }
    }
}
```

Данный метод принимал в качестве аргумента список типа User. Далее с помощью StreamWriter мы создаём наш файл. Далее с помощью цикла foreach перебираем список users. Метод сохранял информацию о пользователи в таком формате {user.Name}*{user.Email}*{user.Password}*{user.AccessKey}*{basketString}

Геннадий*gena444@mail.ru*gena444*2

Геннадий – имя

gena444@mail.ru – почта

gena444 – пароль

2 - уровень доступа, где 0 - покупатель, 1 – работник, 2 – админ

Все поля разделены знаком «*», что указано в коде

Если пользователь является покупателем, то к его данным прибавлялись ещё и данные об id товаров в корзине:

Вот пример: Nikolay*nikolas@mail.ru*nik234*0*0,2,5

Nikolay – имя

nikolas@mail.ru – почта

nik234 – пароль

0 – уровень доступа, где 0 - покупатель, 1 – работник, 2 – админ

0,2,5 – id товаров в корзине

Для сохранения данных о товарах был создан метод «SaveClothes»

Листинг 5.1.15 Метод для сохранения данных о всех товарах:

```
public static void
SaveClothes(List<Clothes> cloth)
{
    using (StreamWriter writer = new
StreamWriter(filePathcl))
    {
        foreach (var cl in cloth)
        {

writer.WriteLine($"{cl.Name}
{cl.Color} {cl.Type} {cl.Size}
{cl.Price}");
        }
    }
}
```

Метод получал в качестве аргумента список типа Cloth, в котором хранятся данные о товарах. Далее с помощью StreamWriter мы создаём наш

файл. Далее с помощью цикла foreach перебираем список cloth. Метод сохранял каждый объект таким образом, разделяя информацию пробелом:

{cl.Name} {cl.Color} {cl.Type} {cl.Size} {cl.Price}

Dripper Зёленый Кепка 52 1470

Dripper – Название

Зёленый – Цвет

Кепка – Тип товара

52 – Размер

1470 – Цена

Рассмотрим метод для загрузки пользователей в систему. Я создал метод void LoadUser():

Листинг 5.1.16 Метод для загрузки пользователей в систему:

```
public static void LoadUser()
 {
     if (File.Exists(filePathus))
     {
         using (StreamReader reader =
new StreamReader(filePathus))
         {
             string line;
             while ((line =
reader.ReadLine()) != null)
             {
                 string[] parts =
line.Split('*');
                 if (parts.Length > 3)
                 {
                     string name =
parts[0];
                     string email =
parts[1];
                     string password =
parts[2];
                     byte AccessKey =
byte.Parse(parts[3]);

                     if (AccessKey ==
0)
                     {
                         List<int>
basket =
parts[4].Split(',').Select(int.Parse).
ToList();
                         new
Customer(name, email, password,
AccessKey, basket);
```

```
                    }
                    if (AccessKey ==
1)
                    {
                        new
Employee(name, email, password,
AccessKey);
                    }
                    if (AccessKey ==
2)
                    {
                        new
Admin(name, email, password,
AccessKey);
                    }

                }
            }
        }
    }
 }
```

С помощью StreamReader мы считываем информацию с нашего файла, который находится в файле, название которого хранит поле filePathus. Вводим символ, по которому мы будем разделять части друг от друга, в нашем случае это «*»:

Листинг 5.1.17 Локальная переменная, которая хранит символ-разделитель:

```
 string[] parts = line.Split('*');
```

Далее создаём локальные переменные и обозначаем, какой переменной приравнять какую часть нашей строки:

Листинг 5.1.18 Локальные переменные, обозначающие данные пользователя, которые были импортированы из файла:

```
string name = parts[0];
string email = parts[1];
string password = parts[2];
byte AccessKey = byte.Parse(parts[3]);
```

Далее мы создаём с помощью этих переменных объекты классов, которые мы определяем с помощью переменной AccessKey: если = «0», то это покупатель, для которого мы также загружаем корзину и создаём объект класса Custoner:

Листинг 5.1.19 Создание объектов дочерних от класса User

```
if (AccessKey == 0)
                    {
```

```
                            List<int>
basket =
parts[4].Split(',').Select(int.Parse).
ToList();
                            new
Customer(name, email, password,
AccessKey, basket);
                        }
```

Если = «1», то это рабочий, и мы просто создаёт объект класса Employee:

Листинг 5.1.20 Создание объекта класса Employee:

```
if (AccessKey == 1)
{
new Employee(name, email, password,
AccessKey);
}
```

Если = «2», то это админ, и мы создаём объект класса Admin:

Листинг 5.1.21 Создание объекта класса Admin:

```
if (AccessKey == 2)
{
new Admin(name, email, password,
AccessKey);
}
```

Рассмотрим метод загрузки товаров:

Листинг 5.1.22 Метод для загрузки данных о товарах:

```
public static void LoadCloth()
{
    if (File.Exists(filePathcl))
    {
        using (StreamReader reader =
new StreamReader(filePathcl))
        {
            string line;
            while ((line =
reader.ReadLine()) != null)
            {
                string[] parts =
line.Split(' ');
                if (parts.Length == 5)
                {
                    string name =
parts[0];
                    string color =
parts[1];
                    string type =
parts[2];
                    string size =
parts[3];
                    double price =
double.Parse(parts[4]);
```

```
                    new
Clothes(name,color,type,size,price);


                }
            }
        }
    }
}
```

С помощью StreamReader мы считываем информацию с нашего файла, который находится в файле, название которого хранит поле filePathcl. Вводим символ, по которому мы будем разделять части друг от друга, в нашем случае это « »:

Листинг 5.1.23 Локальная переменная, которая хранит символ-разделитель:

```
string[] parts = line.Split(' ');
```

Далее создаём локальные переменные и обозначаем, какой переменной приравнять какую часть нашей строки:

Листинг 5.1.24 Локальные переменные, которые хранят данные о товаре:

```
string name = parts[0];
string color = parts[1];
string type = parts[2];
string size = parts[3];
double price = double.Parse(parts[4]);
```

Далее создаём с помощью этих переменных объект класса Clothes:

Листинг 5.1.25 Создаем объект класса Clothes:

```
New
Clothes(name,color,type,size,price);
```

Перейдём к основной части программы - internal class Program. Класс, в котором проходит всё взаимодействие с пользователем моей программы.

Я создал методы для авторизации и регистрации пользователей:

Листинг 5.1.26 Методы для авторизации и регистрации:

```
public static void Enter()
public static void Registration()
```

Метод Enter проводит корректность вводимых пользователем данных и проверяет наличие пользователя в базе данных. Если пользователя нет, то он предлагает зарегистрироваться и вызывает метод для регистрации. Если

пароль был введён неверно, то он об этом сообщает и процесс авторизации проходит заново.

Метод Registration позволяет пользователю зарегистрироваться в системе, метод просит пользователя ввести имя, если имя занято, то метод просит пользователя изменить имя. После этого человек вводит почту и пароль для своего аккаунта. Далее метод создаёт объект класса Customer

Каждый метод после его завершения сохраняет имя текущего пользователя, чтобы знать какие функции выдать для пользования человеку.

Далее у нас запускается static void Main(string[] args). Сразу загружаются данные с помощью ранее написанных методов в классе Data:

Листинг 5.1.27 Вызов методов для загрузки данных при запуске программы:

```
Data.LoadCloth();
Data.LoadUser();
```

Далее с помощью ввода с консоли мы просим пользователя либо зайти в свой аккаунт либо создать новый.

Далее с помощью ранее сохранённого имени пользователя в методах входа или регистрации мы запоминаем текущего пользователя:

Листинг 5.1.28 Создаём переменную, которая будет хранить в себе данные о текущем пользователе:

```
User sUser =
AllUser.FindUser(selectname);
```

Мы создаём новую переменную типа User, в которую сразу помещаем объект из списка всех пользователей с тем именем, которое нам нужно.

В зависимости от типа пользователя его ждёт разный функционал:

Если пользователь = Customer, то:

Он может начать просмотр товаров

Я решил, что пользователь будет видеть случайный товар и по описанию должен решить, добавлять в корзину или нет. Для этого я использовал переменную типа Random:

Листинг кода 5.1.29 Выбор товара, который увидит пользователь:

```
Random random = new Random();
 int randindex =
random.Next(AllClothes.clothes.Count);
randindex =
random.Next(AllClothes.clothes.Count);
```

Далее создаём локальную переменную типа Clothes:

Листинг 5.1.30 Создание локальной переменной типа Clothes:

```
Clothes cloth =
AllClothes.clothes[randindex];
```

И показываем информацию о товаре с помощью метода класса Clothes ShowInfo(). Далее даём пользователю выбор: добавить вещь, пропустить вещь или выйти. Если пользователь добавляет вещь, то мы в корзину этого пользователя добавляем id товара:

Листинг 5.1.31 Добавление товара в корзину пользователя:

```
sUser.basket.Add(cloth.ID);
```

И сохраняем с помощью Data.SaveUsers

Если пользователь выбирает следующую вещь, то мы просто начинаем следующую итерацию цикла

Если пользователь выходит, то мы выходим из цикла и снова предлагаем выбор между просмотром товара и корзиной.

Он может зайти в корзину

Когда пользователь заходит в корзину, у него есть выбор между двумя действиями: оформить заказ и выйти. Если пользователь выбирает оформить заказ, то мы вызываем такой блок кода:

Листинг 5.1.32 Оформление заказа:

```
double sum = 0;
foreach (int id in sUser.basket)
{
    if (id == 0)
    {
        continue;
    }
    sum +=
AllClothes.clothes[id].Price;
}
Console.WriteLine($"Заказ вышел на
{sum} рублей");
sUser.basket = new List<int>();
sUser.basket.Add(0);
Data.SaveUsers(AllUser.persons);
```

Мы создаём локальную переменную для подсчёта суммы заказа и перебираем все товары в корзине пользователя с помощью цикла foreach. Далее мы просто складываем цены всех товаров с помощью id товаров в корзине и списка в классе AllClothes

Далее мы обнуляем корзину и сохраняем данные.

Если пользователь выбирает выйти, то мы просто завершаем цикл

Если пользователь = «Employee», то он может:

Добавить товар

При добавлении товара программа вызывает этот блок кода:

Листинг 5.1.33 Добавление товара в базу товаров:

```
Console.Clear();
Console.WriteLine();
string name;
string color;
string type;
string size;
double price;
Console.WriteLine("|Добавление
товара|\n");
Console.WriteLine("Введите название:
");
name = Console.ReadLine();
Console.WriteLine("Введите цвет: ");
color = Console.ReadLine();
Console.WriteLine("Введите тип товара:
");
type = Console.ReadLine();
Console.WriteLine("Введите размер: ");
size = Console.ReadLine();
Console.WriteLine("Введите цену: ");
price =
double.Parse(Console.ReadLine());
Clothes newClothes = new Clothes(name,
color, type, size, price);
Data.SaveClothes(AllClothes.clothes);
Console.Clear();
Console.WriteLine($"Товар
{newClothes.Name} добавлен");
Console.WriteLine();
```

Мы создаём локальные переменные, которые попросим заполнить пользователя. Просим пользователя ввести название, цвет, тип товара, размер, цену.

Далее создаём объект класса Clothes, в конструктор которого передаём локальные переменные. Потом сохраняем список товаров.

Удалить товар

Мы просим пользователя ввести название товара и запоминаем его. С помощью функции поиска в классе AllClothes мы находим этот товар и удаляем его. Далее сохраняем список товаров.

Посмотреть все товары

Выводим список всех товаров с помощью метода класса AllClothes

Листинг 5.1.34 Вывод данных о всех товарах:

```
AllClothes.ShowAll();
```

Если пользователь = «Admin», то он может:

Добавить сотрудника

Для добавления сотрудника программа запускает такой блок кода:

Листинг 5.1.35 Добавление сотрудника:

```
Console.Clear();
 string name;
 string email;
 string pass;
 Console.WriteLine("|Добавление
сотрудника|\n");
 Console.WriteLine("Введите имя
сотрудника: ");
 name = Console.ReadLine();
 Console.WriteLine("Введите почту
сотрудника: ");
 email = Console.ReadLine();
 Console.WriteLine("Введите пароль
сотрудника: ");
 pass = Console.ReadLine();
 Employee newemployee = new
Employee(name, email, pass,1);
 Data.SaveUsers(AllUser.persons);
```

Мы создаём локальные переменные и просим админа ввести в них имя, почту и пароль сотрудника. После этого создаём объект класса Employee и в конструктор передаём локальные переменные. Сохраняем список пользователей

Удалить сотрудника

Просим админа ввести имя сотрудника для удаления. С помощью метода в классе AllUser мы находим и удаляем сотрудника:

Листинг 5.1.36 Увольнение сотрудника:

```
AllUser.RemoveUser(AllUser.FindUser(nam
e));
```

Далее мы сохраняем список пользователей.

Вывести список всех сотрудников

Выводим такой блок кода:

Листинг 5.1.37 Вывод данных о всех сотрудниках

```
Console.WriteLine("|Список
сотрудников|\n");
 foreach (User emp in AllUser.persons)
 {
     if(emp.GetType().Name ==
"Employee")
     {

Console.WriteLine($"{emp.Name},
{emp.Email}, {emp.Password}\n");
     }

 }
```

С помощью цикла foreach перебираем все объекты из списка persons и проверяем, является ли текущий объект рабочим, если да, то выводим информацию об этом сотруднике.

6. Тестирование

Тестирование проведём на основе требований:

Заходим в программу и регистрируем нового пользователя

```
Здравствуйте! Вы находитесь в магазины одежды streetwear <<Ruskiy>>
Войдите в систему или зарегистрируйтесь

1- Войти 2 - Зарегистрироваться

2
```

Рисунок 11 - Регистрация нового пользователя

Вводим данные

```
Введите имя
Игорь
Введите почту
igordemidov233345@yandex.ru
Введите пароль
igor23333
```

Рисунок 12 - Ввод данных

Начнём просмотр товаров и добавим несколько:

```
+---------------------------------------+
|Название              |NeonGlow        |
|Цвет                  |Неоновые        |
|Тип                   | Носки          |
|Размер                | 42             |
|Цена                  | 300            |
+---------------------------------------+


1 - добавить в корзину
2 - следующая вещь
3 - выйти
1
```

Рисунок 13 - Просмотр товаров

Зайдём в корзину и посмотрим добавленные товары:

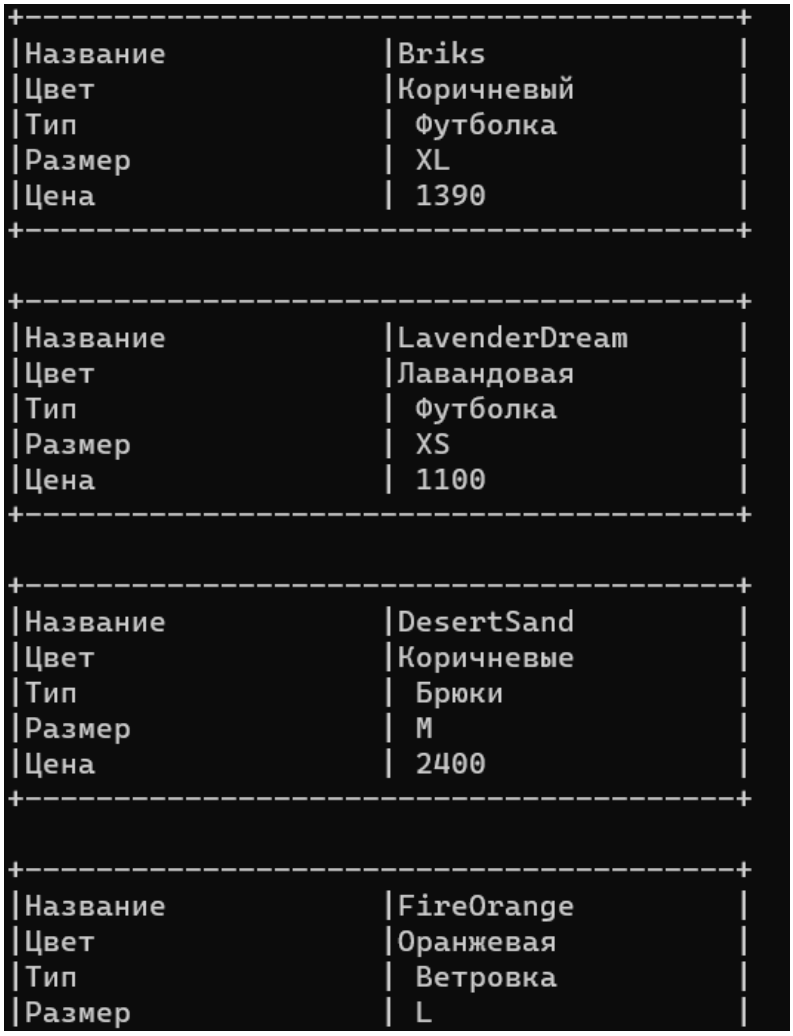

Рисунок 14 - Корзина товаров

Оформим заказ:

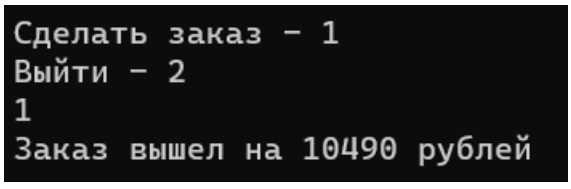

Рисунок 15 - Оформление заказа

Зайдём в корзину и проверим наличие товаров, корзина пуста:

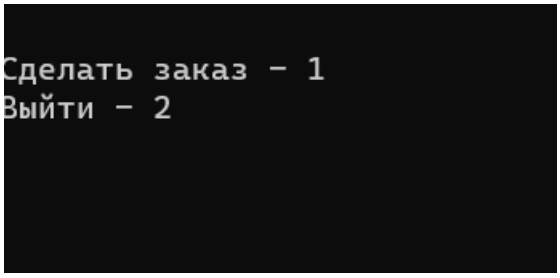

Рисунок 16 - Проверка наличия товара в корзине после заказа

Зайдём в аккаунт рабочего:

```
Здравствуйте! Вы находитесь в магазины одежды streetwear <<Ruskiy>>
Войдите в систему или зарегистрируйтесь

1- Войти 2 - Зарегистрироваться

1
Имя:
Hasber
Пароль:
hasber1
```

Рисунок 17 - Вход в аккаунт рабочего

```
Приветствуем вас Работник Hasber


Выберите действие:
1 - Добавить товар
2 - Удалить товар
3 - Посмотреть все товары
```

Рисунок 18 - Выбираем действие

Добавим товар:

```
|Добавление товара|

Введите название:
BrandMenShirt
Введите цвет:
Белый
Введите тип товара:
Футболка
Введите размер:
XL
Введите цену:
3200
```

Рисунок 19 - Добавление товара

Посмотрим товары:

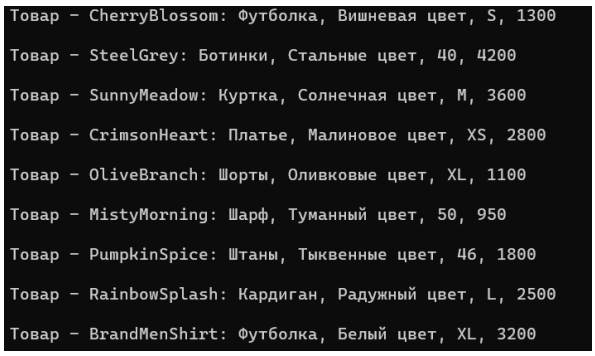

Рисунок 20 - Просмотр товара

Мы видим, что товар добавился

Удалим товар, который только что добавили:

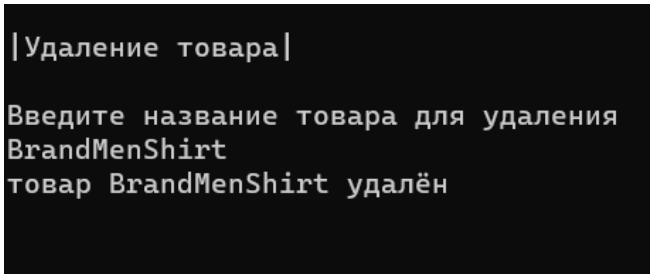

Рисунок 21 - Удаление товара

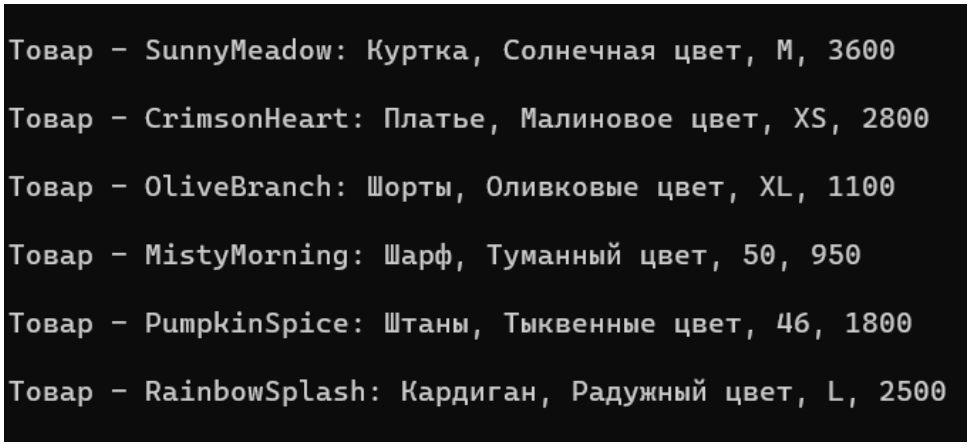

Рисунок 22 - Проверка товаров

Зайдём в аккаунт админа:

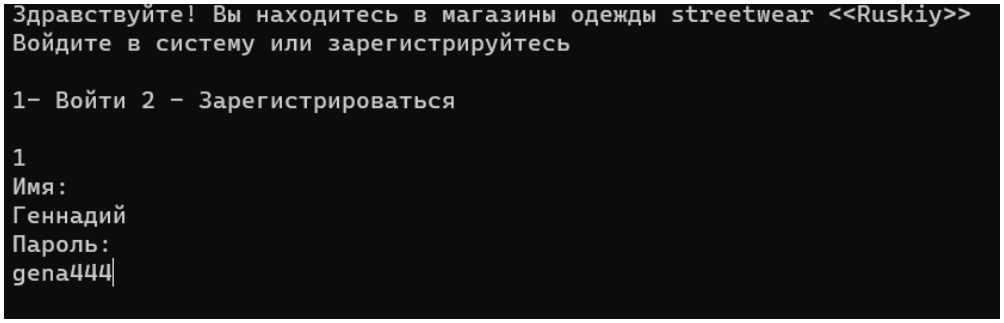

Рисунок 23 - Вход в аккаунт админа

Добавим нового сотрудника:

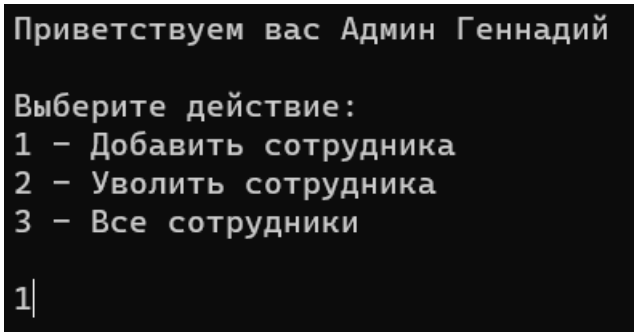

Рисунок 24 - Выбираем действие админа

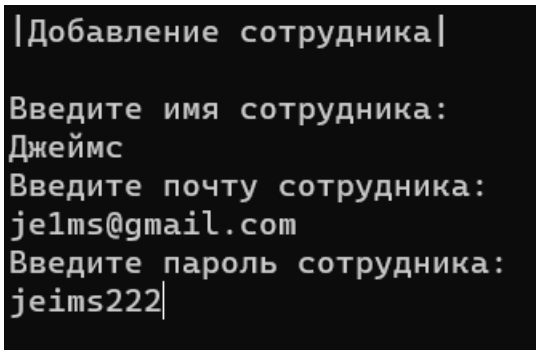

Рисунок 25 - Добавление сотрудника

Посмотрим список всех сотрудников:

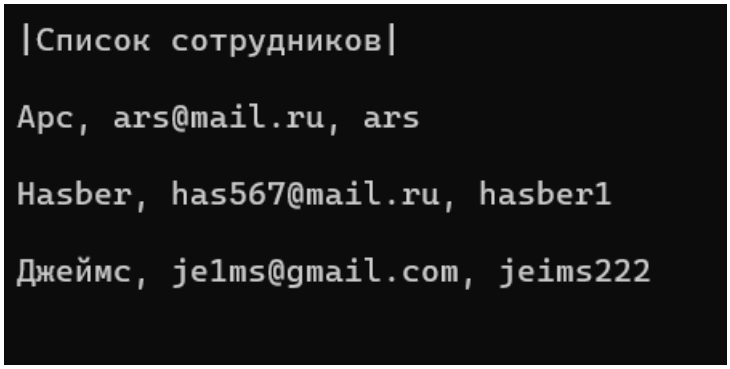

Рисунок 26 - Просмотр списка сотрудников

Уволим сотрудника:

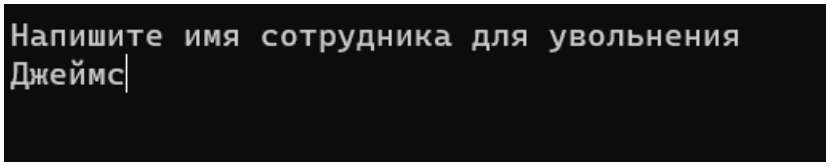

Рисунок 27 - Увольнение сотрудника

Проверим удаление:

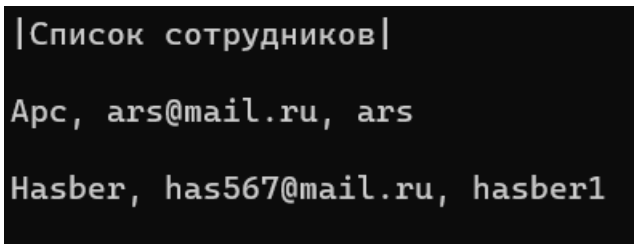

Рисунок 28 - Проверка удаления

7. Заключение

В заключение можно сказать, что программа выполнена в соответствии с поставленными задачами и выполняет всем требованиям. Программа онлайн магазин выполняет все задачи, поставленные в начале работы. Покупатель может сделать заказ, сотрудник может заниматься менеджментом товаров, а админ может заниматься кадровой деятельностью. Программа выполнена по принципам ООП: абстракции, инкапсуляция, наследование, полиморфизм. Программа по результатам тестов работает корректно. Система сохранения также работает корректно и хорошо показала себя с большим объёмом данных.

8. Список литературы

- 1) Троелсен, Э. and Ф. Джепикс, 2018. Язык программирования C# 7 и платформы.NET и .NET Core. Вильямс.
- 2) Албахари, Дж. and Б. Албахари, 2018. C# 7.0. Справочник. Полное описание языка. Вильямс.
- 3) Филлипс Б., Стюарт К., Марсикано К. Android. Программирование для профессионалов. 3-е изд. — СПб.: Питер, 2017. — 688 с.
- 4) Освой программирование играючи. Сайт Александра Климова / URL: http://developer.alexanderklimov.ru/android/
- 5) Мейер Б. Объектно-ориентированное конструирование программных систем. М.: Русская Редакция, 2005
- 6) Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования. С.-Петербург: Питер, 2006. — 156 с.
- 7) Троелсен Э. C# и платформа .NET. С.-Петербург: Питер, 2006.
- 8) Синтес А. Освой самостоятельно объектно-ориентированное программирование за 21 день. Москва; С.-Петербург; Киев: Вильямс, 2002. — 321 с.
- 9) Дж. Кьоу, М. Джеанини. Объектно-ориентированное программирование. С.-Петербург: Питер, 2005.
- 10) Уоткинз Д., Хаммонд М, Эйбрамз Б. Программирование на платформе .NET.: М.: Вильямс, 2003.

9. Приложение

9.1 Program.cs

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

namespace ClothStore
{


    internal class Program
    {
        public static int selectid = 0;
        public static string selectname;
        public static void Enter()   //////////////Функция для
авторизации пользователя
        {
            while (true)
            {
                Console.WriteLine("Имя: ");
                string name = Console.ReadLine();
                Console.WriteLine("Пароль: ");
                string pass = Console.ReadLine();
                if (AllUser.FindUser(name) != null)
                {
                    User usr = AllUser.FindUser(name);
                    if (usr.Password == pass)
                    {
                        selectid = usr.ID;
                        Console.WriteLine("Вы успешно
вошли\n");
                        selectname = name;
                        break;
                    }
                    else
                        Console.WriteLine("Неправильный
пароль\n");
                }
                else
                {
                    Console.WriteLine("Такого пользователя не
существует");
                    Console.WriteLine("Хотите
зарегистрироваться или попробовать снова? \n 1- регистрация 2-
поробовать снова\n");
                    string d = Console.ReadLine();
                    if (d == "1")
                    {
                        Registration();
                    }
                }
            }
        }

        public static void Registration()
///////////////Функция для авторизации
        {
            Console.WriteLine("Введите имя");
```

```
            string name;
            while (true)
            {
                name = Console.ReadLine();
                User usr = AllUser.FindUser(name);
                if(usr != null)
                {
                    Console.WriteLine("Введите новое имя,
текущее занято");
                }
                else
                {
                    break;
                }
            }

            Console.WriteLine("Введите почту");
            string email = Console.ReadLine();
            Console.WriteLine("Введите пароль");
            string pass = Console.ReadLine();
            Customer newcustomer = new
Customer(name,email,pass,0);
            Data.SaveUsers(AllUser.persons);
            selectname = name;
            Console.WriteLine();

        }

        static void Main(string[] args)
        {
            Data.LoadCloth();      ////Загрузка списка с
товарами
            Data.LoadUser();       ////Загрузка списка с
пользователями
            Console.WriteLine("Здравствуйте! Вы находитесь в
магазины одежды streetwear <<Ruskiy>>");
            Console.WriteLine("Войдите в систему или
зарегистрируйтесь\n");

            Console.WriteLine("1- Войти 2 -
Зарегистрироваться\n");
            int variant = int.Parse(Console.ReadLine());
            if (variant == 1)
            {
                Enter();
            }
            else if (variant == 2)
            {
                Registration();
            }
            Console.Clear();
            User sUser = AllUser.FindUser(selectname);
/////Программа запоминает текущего пользователя
            if (sUser.GetType().Name == "Admin")
/////Функционал Админа
            {
                while(true)
                {

                    Console.WriteLine();
                    Console.WriteLine($"Приветствуем вас Админ
{sUser.Name}\n");
                    Console.WriteLine("Выберите действие:");
```

```
                    Console.WriteLine("1 - Добавить
сотрудника");
                    Console.WriteLine("2 - Уволить
сотрудника");
                    Console.WriteLine("3 - Все сотрудники\n");
                    string num = Console.ReadLine();
                    if (num == "1")
                    {
                        Console.Clear();
                        string name;
                        string email;
                        string pass;
                        Console.WriteLine("|Добавление
сотрудника|\n");
                        Console.WriteLine("Введите имя
сотрудника: ");
                        name = Console.ReadLine();
                        Console.WriteLine("Введите почту
сотрудника: ");
                        email = Console.ReadLine();
                        Console.WriteLine("Введите пароль
сотрудника: ");
                        pass = Console.ReadLine();
                        Employee newemployee = new
Employee(name, email, pass,1);
                        Data.SaveUsers(AllUser.persons);
                    }
                    else if (num == "2")
                    {
                        Console.Clear();
                        Console.WriteLine("Напишите имя
сотрудника для увольнения");
                        string name = Console.ReadLine();

AllUser.RemoveUser(AllUser.FindUser(name));
                        Data.SaveUsers(AllUser.persons);
                    }
                    else if (num == "3")
                    {
                        Console.Clear();
                        Console.WriteLine("|Список
сотрудников|\n");
                        foreach (User emp in AllUser.persons)
                        {
                            if(emp.GetType().Name ==
"Employee")
                            {
                                Console.WriteLine($"{emp.Name},
{emp.Email}, {emp.Password}\n");
                            }

                        }
                    }
                }
            }
            else if (sUser.GetType().Name == "Employee")
////Функционал Работника
            {
                Console.WriteLine($"Приветствуем вас Работник
{sUser.Name}\n");
                while(true)
                {
                    Console.WriteLine();
```

```
                    Console.WriteLine("Выберите действие:");
                    Console.WriteLine("1 - Добавить товар");
                    Console.WriteLine("2 - Удалить товар");
                    Console.WriteLine("3 - Посмотреть все
товары\n");
                    string num = Console.ReadLine();
                    if (num == "1")
                    {
                        Console.Clear();
                        Console.WriteLine();
                        string name;
                        string color;
                        string type;
                        string size;
                        double price;
                        Console.WriteLine("|Добавление
товара|\n");
                        Console.WriteLine("Введите название:
");
                        name = Console.ReadLine();
                        Console.WriteLine("Введите цвет: ");
                        color = Console.ReadLine();
                        Console.WriteLine("Введите тип товара:
");
                        type = Console.ReadLine();
                        Console.WriteLine("Введите размер: ");
                        size = Console.ReadLine();
                        Console.WriteLine("Введите цену: ");
                        price =
double.Parse(Console.ReadLine());
                        Clothes newClothes = new Clothes(name,
color, type, size, price);
                        Data.SaveClothes(AllClothes.clothes);
                        Console.Clear();
                        Console.WriteLine($"Товар
{newClothes.Name} добавлен");
                        Console.WriteLine();
                    }
                    else if (num == "2")
                    {
                        Console.Clear();
                        Console.WriteLine();
                        Console.WriteLine("|Удаление
товара|\n");
                        Console.WriteLine("Введите название
товара для удаления");
                        string name = Console.ReadLine();

AllClothes.RemoveCloth(AllClothes.FindCloth(name));
                        Data.SaveClothes(AllClothes.clothes);
                        Console.WriteLine();
                    }
                    else if (num == "3")
                    {
                        Console.Clear();
                        Console.WriteLine("|Просмотр всех
товаров|\n");
                        AllClothes.ShowAll();
                    }
                }
            }
            else                    ////////Функционал
Пользователя
```

```
            {
                Console.WriteLine($"Приветствуем вас
{sUser.Name}");
                Console.WriteLine("Выберите действие:");
                while (true)
                {
                    Console.WriteLine();
                    Console.WriteLine("1 - Начать просмотр
товаров");
                    Console.WriteLine("2 - Зайти в корзину");
                    string c = Console.ReadLine();
                    if (c == "1")
                    {
                        Console.Clear();
                        while (true)
                        {
                            Random random = new Random();
                            int randindex =
random.Next(AllClothes.clothes.Count);
                            while(true)
                            {
                                if (randindex == 0)
                                    randindex =
random.Next(AllClothes.clothes.Count);
                                else
                                    break;
                            }
                            Clothes cloth =
AllClothes.clothes[randindex];
                            cloth.ShowInfo();
                            Console.WriteLine();
                            Console.WriteLine("1 - добавить в
корзину");
                            Console.WriteLine("2 - следующая
вещь");
                            Console.WriteLine("3 - выйти");
                            string n = Console.ReadLine();
                            if (n == "1")
                            {
                                Console.Clear();
                                sUser.basket.Add(cloth.ID);
                                Console.WriteLine("Добавлено");

Data.SaveUsers(AllUser.persons);
                            }
                            else if (n == "2")
                            {
                                Console.Clear();
                            }
                            else if (n == "3")
                            {
                                Console.Clear();
                                break;
                            }
                        }

                    }
                    if (c == "2")
                    {
                        Console.Clear();
                        foreach (int id in sUser.basket)
                        {
                            if (id == 0)
```

```
                                {
                                    continue;
                                }
                                AllClothes.clothes[id].ShowInfo();
                            }
                            Console.WriteLine();
                            Console.WriteLine("Сделать заказ - 1");
                            Console.WriteLine("Выйти - 2");
                            string d = Console.ReadLine();
                            if (d == "1")
                            {
                                double sum = 0;
                                foreach (int id in sUser.basket)
                                {
                                    if (id == 0)
                                    {
                                        continue;
                                    }
                                    sum +=
AllClothes.clothes[id].Price;
                                }
                                Console.WriteLine($"Заказ вышел на
{sum} рублей");
                                sUser.basket = new List<int>();
                                sUser.basket.Add(0);
                                Data.SaveUsers(AllUser.persons);
                            }
                        }
                    }
                }
            }

    }
}
```

9.2 User.cs

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

namespace ClothStore.Classes
{
    abstract class User
    {
        public static int Id { get; set; }
        public abstract int ID { get; set; }
        public abstract string Name { get; set; }
        public abstract string Email { get; set; }
        public abstract string Password { get; set; }
        public abstract byte AccessKey { get; set; }
        public List<int> basket = new List<int>();


    }
}
```

9.3 Customer.cs

```
using System;
using System.Collections.Generic;
```

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

namespace ClothStore.Classes
{
    class Customer : User
    {
        public override int ID { get; set; }
        public override string Name { get ; set ; }
        public override string Email { get ; set ; }
        public override string Password { get; set; }
        public override byte AccessKey { get; set; }

        public Customer(string name, string email, string
password, byte accesKey, List<int> Basket)
        {
            ID = User.Id;
            User.Id++;
            basket = Basket;
            AccessKey = accesKey;
            Name = name;
            Email = email;
            Password = password;
            basket.Add(0);
            AllUser.AddUser(this);
        }
        public Customer(string name, string email, string
password, byte accesKey)
        {
            ID = User.Id;
            User.Id++;
            AccessKey = accesKey;
            Name = name;
            Email = email;
            Password = password;
            basket.Add(0);
            AllUser.AddUser(this);
        }
    }
}
```

9.4 Admin.cs

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

namespace ClothStore.Classes
{
    class Admin : User
    {
        public override int ID { get; set; }
        public override string Name { get; set; }
        public override string Email { get; set; }
        public override string Password { get; set; }
        public override byte AccessKey { get; set; }
        public Admin(string name,string email,string password,
byte accesKey)
        {
            ID = User.Id;
            User.Id++;
```

```
            AccessKey = accesKey;
            Name = name;
            Email = email;
            Password = password;
            basket.Add(0);
            AllUser.AddUser(this);
        }

    }
}
```

9.5 Employee.cs

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

namespace ClothStore.Classes
{
    class Employee : User
    {
        public override int ID { get; set; }
        public override string Name { get; set; }
        public override string Email { get; set; }
        public override string Password { get; set; }
        public override byte AccessKey { get; set; }
        public Employee(string name, string email, string
password, byte accesKey)
        {
            ID = User.Id;
            User.Id++;
            AccessKey = accesKey;
            Name = name;
            Email = email;
            Password = password;
            basket.Add(0);
            AllUser.AddUser(this);
        }
    }
}
```

9.6 Clothes.cs

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

namespace ClothStore.Classes
{
     class Clothes
    {
        public string Name { get; set; }
        public string Color { get; set; }
        public string Type { get; set; }
        public string Size { get; set; }
        public double Price { get; set; }
        public int ID { get; set; }

        public Clothes(string name, string description, string
type, string size, double price)
        {
```

```
            ID = AllClothes.Id;
            AllClothes.Id++;
            Name = name;
            Color = description;
            Type = type;
            Size = size;
            Price = price;
            AllClothes.AddCloth(this);
        }


        public void ShowInfo()     ////Вывод информации о
товаре
        {
            Console.WriteLine("+" + new string('-', 40) + "+");
            Console.WriteLine($"|{"Название",-20}|{Name,-
19}|");
            Console.WriteLine($"|{"Цвет",-20}|{Color,-19}|");
            Console.WriteLine($"|{"Тип",-19} | {Type,-18}|");
            Console.WriteLine($"|{"Размер",-19} | {Size,-
18}|");
            Console.WriteLine($"|{"Цена",-19} | {Price,-18}|");
            Console.WriteLine("+" + new string('-', 40) + "+");
            Console.WriteLine();
        }

    }
}
```

9.7 AllClothes.cs

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

namespace ClothStore.Classes
{
    static class AllClothes
    {
        public static int Id { get; set; }         ////Учёт
следующего id для товара
        public static List<Clothes> clothes = new
List<Clothes>();        ////List с товарами
        public static void AddCloth(Clothes cloth)
////Добавление товара
        {
            clothes.Add(cloth);
        }
        public static void RemoveCloth(Clothes cloth)
////Удаление товара
        {
            clothes.Remove(cloth);
            Console.WriteLine($"товар {cloth.Name} удалён\n");
        }
        public static Clothes FindCloth(string name)
////Поиск товара по имени
        {
            return clothes.Find(cloth => cloth.Name == name);
        }
        public static void ShowAll()    ////Вывод всех товаров
в консоль
        {
```

```
            foreach(Clothes cloth in clothes)
            {
                Console.WriteLine($"Товар - {cloth.Name}:
{cloth.Type}, {cloth.Color} цвет, {cloth.Size},
{cloth.Price}\n");
            }
        }

    }
}
```

9.8 AllUsers.cs

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

namespace ClothStore.Classes
{
    static class AllUser
    {
        public static List<User> persons = new List<User>();
////List всех пользователей
        public static void AddUser(User user)
////Добавление пользователя
        {
            persons.Add(user);
        }
        public static void RemoveUser(User user)
////Удаление пользователя
        {
            persons.Remove(user);
            Console.WriteLine($"Пользователь {user.Name}
удалён\n");
        }
        public static User FindUser(int id)         ////Поиск
по id
        {
            return persons.Find(user => User.Id == id);
        }
        public static User FindUser(string name)     ////Поиск
по имени
        {
            return persons.Find(user => user.Name == name);
        }

    }
}
```

9.9 Data.cs

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

namespace ClothStore.Classes
{
    static class Data
    {
        private static string filePathus = "users.txt";
```

```
        private static string filePathcl = "clothes.txt";
        public static void SaveUsers(List<User> users)
        {
            using (StreamWriter writer = new
StreamWriter(filePathus))
            {
                foreach (var user in users)
                {
                    if (user.GetType().Name == "Customer")
                    {
                        string basketString = string.Join(",",
user.basket);

writer.WriteLine($"{user.Name}*{user.Email}*{user.Password}*{us
er.AccessKey}*{basketString}");
                    }
                    else
                    {

writer.WriteLine($"{user.Name}*{user.Email}*{user.Password}*{us
er.AccessKey}");
                    }

                }
            }
        }
        public static void SaveClothes(List<Clothes> cloth)
        {
            using (StreamWriter writer = new
StreamWriter(filePathcl))
            {
                foreach (var cl in cloth)
                {
                    writer.WriteLine($"{cl.Name} {cl.Color}
{cl.Type} {cl.Size} {cl.Price}");
                }
            }
        }

        public static void LoadUser()
        {
            if (File.Exists(filePathus))
            {
                using (StreamReader reader = new
StreamReader(filePathus))
                {
                    string line;
                    while ((line = reader.ReadLine()) != null)
                    {
                        string[] parts = line.Split('*');
                        if (parts.Length > 3)
                        {
                            string name = parts[0];
                            string email = parts[1];
                            string password = parts[2];
                            byte AccessKey =
byte.Parse(parts[3]);

                            if (AccessKey == 0)
                            {
                                List<int> basket =
parts[4].Split(',').Select(int.Parse).ToList();
```

```
                                new Customer(name, email,
password, AccessKey, basket);
                            }
                            if (AccessKey == 1)
                            {
                                new Employee(name, email,
password, AccessKey);
                            }
                            if (AccessKey == 2)
                            {
                                new Admin(name, email,
password, AccessKey);
                            }

                        }
                    }
                }
            }
        }

        public static void LoadCloth()
        {
            if (File.Exists(filePathcl))
            {
                using (StreamReader reader = new
StreamReader(filePathcl))
                {
                    string line;
                    while ((line = reader.ReadLine()) != null)
                    {
                        string[] parts = line.Split(' ');
                        if (parts.Length == 5)
                        {
                            string name = parts[0];
                            string color = parts[1];
                            string type = parts[2];
                            string size = parts[3];
                            double price =
double.Parse(parts[4]);

                            new
Clothes(name,color,type,size,price);


                        }
                    }
                }
            }


    }
}
```