МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ижевский государственный технический университет имени М. Т. Калашникова»

Институт «Информатика и вычислительная техника»

Кафедра «Программное обеспечение»

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_»

Дата \_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Объектно-ориентированное программирование»

на тему: Информационная система «Банкомат»

Выполнил

студент гр. Б04-191-3 Р. А. Гумметов

Руководитель

к.т.н., доцент В. Г. Власов

Рецензия:

степень достижения поставленной цели работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

полнота разработки темы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

уровень самостоятельности работы обучающегося\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

недостатки работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc44616610)

[1. ОПИСАНИЕ ЗАДАНИЯ 4](#_Toc44616611)

[1.1 Первоначальные требования 4](#_Toc44616612)

[1.2 Описание требований 4](#_Toc44616613)

[1.3 Уточненные требования 5](#_Toc44616614)

[2. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc44616615)

[4. ДИАГРАММА КЛАССОВ 10](#_Toc44616616)

[5. ВЫБОР И ОБОСНОВАНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ 17](#_Toc44616617)

[ЗАКЛЮЧЕНИЕ 18](#_Toc44616618)

[ПРИЛОЖЕНИЕ 19](#_Toc44616619)

# ВВЕДЕНИЕ

В последние время активно развивается информатизация банковского дела, связанная с использованием современных информационных систем. Это, в первую очередь, связано с востребованностью информационных технологий на динамично развивающемся рынке банковских услуг. Во-вторых, высокий спрос населения на автоматизацию. Банкомат - программно-технический комплекс, предназначенный для автоматизированных выдачи и приёма наличных денежных средств с использованием платёжных карт.

Целью курсовой работы является разработать информационную систему банкомата.

# 1. ОПИСАНИЕ ЗАДАНИЯ

## 1.1 Первоначальные требования

В курсовой работе должна быть разработана информационная система банкомата. Необходимо реализовать выполнение следующих пользовательских запросов:

1. прием карты

2. Проверка ПИН через сервер банка — формат JSON

3. выдача наличных (с механизмом проверки )

4. прием наличных (с механизмом проверки внесенных банкнот)

5. показ информации о балансе

Формат внутреннего сохранения текущих данных на ЖД банкомата - XML. Содержимое JSON и XML разрабатывается студентом.

## 1.2 Описание требований

Информационная система банкомата должна содержать файлы с текущими данными на ЖД банкомата. Файл xml должен содержать данные о банковских картах, такие как номер карты, баланс на карте, пользователь, который привязан к данной карте. При запуске программы, после того как пользователь вставил карту в банкомат, программа должна запрашивать номер и ПИН его банковской карты. После этого программа должна проверять соответствие номера и ПИН карты через сервер банка и если пользователь ввел правильные данные, то программа должна предоставить меню. В меню должны быть такие пункты, как выдача наличных, прием наличных и показ информации о балансе. При изменении текущих данных после работы программы, например, изменение баланса какой-либо карты, программа должна сериализовать новые данные и сохранить их в файл формата xml.

## 1.3 Уточненные требования

При запуске программа должна спрашивать пользователя, хочет ли он вставить карту. Если пользователь вставляет карту, то программа запрашивает номер карты и ПИН. Если данные введены пользователем неправильно, то программа сообщает об этом и заканчивает работу. В случае, если данные введены правильно, программа предоставляет меню:

1) Внести наличные

2) Снять наличные

3) Узнать баланс

4) Выйти

При выборе первого пункта меню программа должна попросить пользователя написать сумму, которую он хочет внести, а после этого добавить на счет карты данную сумму.

При выборе второго пункта меню программа должна попросить пользователя написать сумму, которую он хочет снять, а после этого списать с счета карты данную сумму.

При выборе третьего пункта меню программа должна вывести на экран баланс на карте.

При выборе четвертого пункта меню программа должна закончить работу.

# 2. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Банкомат - автоматизированное устройство, предназначенное для осуществления операций с денежными средствами. Операции могут осуществляться в рамках использования пластиковых карт (дебетовых и кредитных) или без них. Банкоматы позволяют осуществлять прием и выдачу наличных средств, оплачивать товары/услуги, подтверждать операции, получать выписку о состоянии счета, подключать/отключать опции на пластиковых картах и др. Стоит заметить, что банкомат конкретного банка может обслуживать не только карты, выпущенные этим кредитным учреждением, но и карты других банков. Однако, как правило, за операции с картами других банков (за исключением случаев, когда эти банки являются партнерами) взимается комиссия.

Принцип действия банкомата следующий. После введения карты банкомат запрашивает ПИН-код (секретный код), далее пользователь осуществляет запрос на проведение определенной операции. Запрошенная операция, ПИН-код, содержимое карты (магнитная полоса и др.) шифруются и направляются в систему банка, который обслуживает данное устройство (этот банк называется банк-эквайер). Далее запрос поступает в платежную систему, которая, в свою очередь, маршрутизирует запрос в банк, который выпустил карту. После рассмотрения запроса ответ поступает в банкомат. Весь этот длинный путь запроса фиксируется в банкомате, чтобы в случае возникновения спорных ситуаций можно было провести расследование. Держателям карт важно помнить, что свой ПИН-код ни в коем случае нельзя передавать третьим лицам, так как операции, которые были подтверждены ПИН-кодом, считаются операциями, совершенными владельцем карты.

Составим словарь предметной области. Для банкомата существительными могут быть: "карта", "владелец", "номер", "ПИН-код", "фамилия", "имя", "отчество". В качестве глаголов могут быть: "вставить карту", "снять наличные", "внести наличные", "узнать баланс", "приветствие".

3. РАЗРАБОТКА СХЕМЫ ФУНКЦИОНАЛЬНЫХ БЛОКОВ

Для понимания того, что нам необходимо реализовать, нужно выдвинуть конкретные требования для разрабатываемой программы. Для этого требуется разработать схему функциональных блоков, которая будет отображать суть работы системы. На рис.1 представлена схема работы системы:

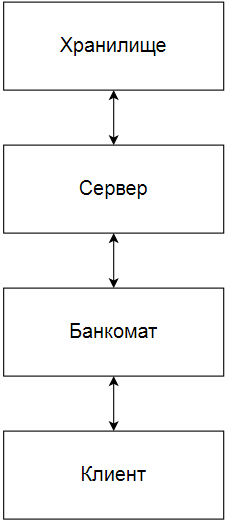


Рис. 1. Схема функциональных блоков

Для понимания того, как необходимо организовывать связи между этими блоками, необходимо разработать схему того, как будут обмениваться данными

блоки. На рис.2 представлена данная схема:

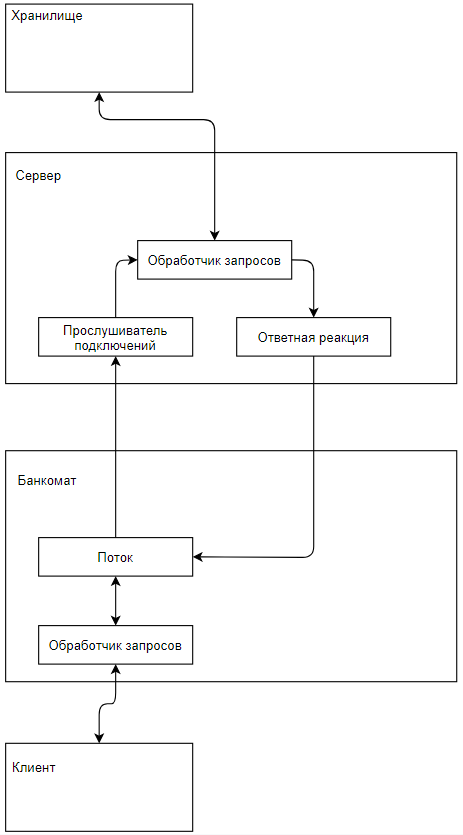


Рис. 2. Схема обмена данными между блоками

После запуска программа должна запрашивать номер и ПИН-код банковской карты пользователя. После ввода, банкомат отправляет запрос к серверу с целью проверки соответствующего ПИН-кода у карты с данным номером. Данные всех банковских карт находятся в хранилище, поэтому сервер обращается к хранилищу. После этого в банкомат поступает ответная реакция от сервера. В случае, если ПИН-код введен правильно, программа должна предоставить пользователю меню, в котором он может выбрать необходимую ему операцию. Далее банкомат снова отправляет запрос к серверу с целью выполнения операции, сервер, в свою очередь обращается к хранилищу, после чего в банкомат отправляется ответная реакция сервера на исходный запрос.

# 4. ДИАГРАММА КЛАССОВ

Начнем с обработки словаря предметной области. Эта обработка состоит в выявлении того, какие слова соответствуют объектам, классам, свойствам и методам. Список существительных служит основой для выделения классов и их свойств, с список глаголов - для определения методов. Для информационной системы банкомата укажем следующее:

- классы: карта, владелец;

- свойства: номер, ПИН-код, фамилия, имя, отчество;

- методы: вставить карту, снять наличные, внести наличные, узнать баланс, приветствие.

Следующий шаг состоит в том, чтобы определить, какой из классов какие свойства и методы содержит. Для класса "карта" справедливо наличие таких свойств, как "номер" и "ПИН-код", а также методов "вставить карту", "снять наличные", "внести наличные" и "узнать баланс". Для класса "владелец" справедливо наличие таких свойств, как "фамилия", "имя", "отчество" и метода "приветствие".

Класс "владелец" может являться наследником класса "карта". Таким образом, у класса "владелец" будут поля класса "карта", которые ему доступны и к владельцу может быть привязана определенная карта.

Метод "приветствие" может быть виртуальным в классе "карта" и переопределяться в классе "владелец".

Класс Card:

открытые поля:

* Список владельцев карт
* Список карт
* Номер карты, тип string
* ПИН карты, тип string
* Баланс на карте, тип int
* Метод, предоставляющий пользователю меню
* Метод для ввода номера карты
* Метод для ввода ПИН карты
* Метод, чтобы снять наличные с карты
* Метод, чтобы положить наличные на карту
* Метод, чтобы узнать баланс
* Метод, предоставляющий доступные купюры для ввода и снятия наличных
* Метод, который ищет карту с данными, которые ввел пользователь
* Виртуальный метод, который приветствует пользователя

Класс Person (наследуется от класса Card)

открытые поля:

* Имя пользователя, тип string
* Фамилия пользователя, тип string
* Отчество пользователя, тип string
* Переопределенный метод, который приветствует пользователя

Схема классов представлена на рис. 3.

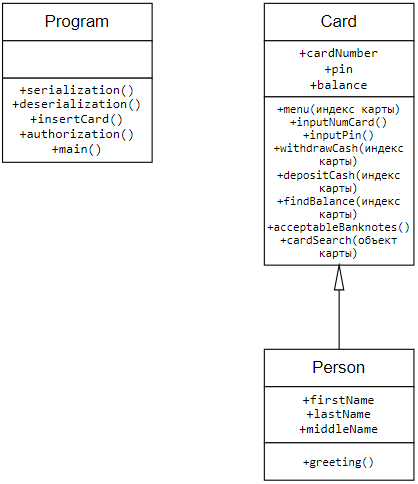


Рис. 3. Схема классов

Описание схемы классов:

Класс Program

На схеме присутствует класс Program, не имеющий свойств. В данном классе содержатся методы сериализации и десериализации дынных xml и json форматов, приема карты, авторизации. Также класс Program содержит метод main(), с которого начинается запуск программы.

Объявление класса Program:

class Program {}

В классе Program присутствуют следующие методы:

* static public void Serialization(). Модификатор static необходим для того, чтобы вызывать данный метод не используя объект класса. Метод должен сериализовать данные в файлы xml и json форматов и не должен ничего возвращать, поэтому используется ключевое слово void. В качестве параметров метод ничего не принимает.
* static public void Deserialization(). Модификатор static необходим для того, чтобы вызывать данный метод не используя объект класса. Метод должен десериализовать данные из файлов xml и json форматов и не должен ничего возвращать, поэтому используется ключевое слово void. В качестве параметров метод ничего не принимает.
* static public int InsertCard(). Модификатор static необходим для того, чтобы вызывать данный метод не используя объект класса. Метод должен предлагать пользователю вставить карту в банкомат и должен вернуть номер команды, введенной пользователем, поэтому используется ключевое слово int. В качестве параметров метод ничего не принимает.
* static public void Authorization(). Модификатор static необходим для того, чтобы вызывать данный метод не используя объект класса. Метод должен произвести авторизацию пользователя и не должен ничего возвращать, поэтому используется ключевое слово void. В качестве параметров метод ничего не принимает.
* static void Main(string[] args). Данный метод является точкой входа в программу, static означает, что можно вызвать функцию без создания объекта класса. Метод ничего не должен возвращать, поэтому используется ключевое слово void. **Параметр метода - это массив string, который представляет аргументы командной строки.**

Класс Card

Класс Card содержит такие свойства, как номер карты, ПИН-код и баланс, а также методы меню, ввод номера карты, ввод ПИН, снять наличные, внести наличные, узнать баланс, показать доступные купюры и поиск карты.

Класс Card является открытым, поэтому объявим его с модификатором доступа public:

public class Card {}

В классе Card должны присутствовать следующие методы:

* static public void Menu(int cardIndex). Модификатор static необходим для того, чтобы вызывать данный метод не используя объект класса. Метод предоставляет пользователю меню и ничего не должен возвращать, поэтому используется ключевое слово void. В качестве параметра метод принимает индекс карты в списке, с которой будет работать.
* static public string InputNumCard(). Модификатор static необходим для того, чтобы вызывать данный метод не используя объект класса. Метод предлагает пользователю ввести номер банковской карты. Он должен вернуть номер карты пользователя в качестве строки, поэтому используется ключевое слово string. В качестве параметров метод ничего не принимает.
* static public string InputPin(). Модификатор static необходим для того, чтобы вызывать данный метод не используя объект класса. Метод предлагает пользователю ввести ПИН банковской карты. Он должен вернуть ПИН карты пользователя в качестве строки, поэтому используется ключевое слово string. В качестве параметров метод ничего не принимает.
* static public void WithdrawCash(int cardIndex). Модификатор static необходим для того, чтобы вызывать данный метод не используя объект класса. Метод должен реализовать снятие наличных с карты пользователя и ничего не должен возвращать, поэтому используется ключевое слово void. В качестве параметра метод принимает индекс карты в списке, с которой будет работать.
* static public void DepositCash(int cardIndex). Модификатор static необходим для того, чтобы вызывать данный метод не используя объект класса. Метод должен реализовать внесение наличных на карту пользователя и ничего не должен возвращать, поэтому используется ключевое слово void. В качестве параметра метод принимает индекс карты в списке, с которой будет работать.
* static public void FindBalance(int cardIndex). Модификатор static необходим для того, чтобы вызывать данный метод не используя объект класса. Метод должен предоставлять информацию о балансе на карте пользователя и ничего не должен возвращать, поэтому используется ключевое слово void. В качестве параметра метод принимает индекс карты в списке, с которой будет работать.
* static public void AcceptableBanknotesOnScreen(). Модификатор static необходим для того, чтобы вызывать данный метод не используя объект класса. Метод должен предоставлять пользователю доступные купюры и ничего не должен возвращать, поэтому используется ключевое слово void. В качестве параметров метод ничего не принимает.
* static public int CardSearch(Card card). Модификатор static необходим для того, чтобы вызывать данный метод не используя объект класса. Метод должен искать в списке карт карту с данными, соответствующими тем, которые ввел пользователь и возвращать индекс этой карты в списке карт, поэтому используется ключевое слово int. В качестве параметров метод принимает объект карты с данными, которые ввел пользователь.
* public virtual void Greeting(). Метод должен приветствовать пользователя после успешной авторизации. Он не должен ничего возвращать, поэтому используется ключевое слово void. В качестве параметров ничего не принимает. Метод является виртуальным и в классе Card его тело пустое.

Класс Person

Класс Person является классом-наследником от класса Card. В нем содержаться такие свойства, как имя, фамилия, отчество, а также метод, приветствующий пользователя после авторизации.

Данный класс также является открытым, поэтому объявим его с модификатором доступа public:

public class Person : Card {}

В классе Person присутствует только один метод:

public override void Greeting(). Метод должен приветствовать пользователя после успешной авторизации. Он не должен ничего возвращать, поэтому используется ключевое слово void. В качестве параметров ничего не принимает. Метод является виртуальным и в классе Card и переопределяется в классе Person.

# 5. ВЫБОР И ОБОСНОВАНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ

Рекомендованными в рамках дисциплины языками являются С++ и С#, поэтому при написании программы для курсовой работы я выбирал между ними. C# является более новым языком программирования по отношению к C++, более высокоуровневым. Например, тут присутствует автоматическое управление памятью. C# использует объектно-ориентированный подход к программированию во всем. Приложения, написанные на C# отличаются удобочитаемостью кода и высокой скоростью разработки. Одним из самых важных плюсов C# является наличие большого количества библиотек и шаблонов. Поэтому я считаю, что для написания программы для курсовой работы отлично подходит C#. В качестве среды разработки выбрана система Microsoft Visual Studio.

# ЗАКЛЮЧЕНИЕ

Программа успешно реализована и работает в соответствии с поставленными задачами. Реализованы такие функции, как снятие наличных с карты, внесение наличных на карту и получение информации о балансе. Создан файл persons.xml, хранящий текущие данные на ЖД банкомата, а также файл cards.json, хранящий номера карт и соответствующие им ПИН. В программе используются такие принципы ОПП, как инкапсуляция (созданы классы на основе сущностей), наследование (class Person наследуется от class Card) и полиморфизм (в классе Card определен виртуальный метод Greeting(), который переопределяется в классе Person).

# ПРИЛОЖЕНИЕ

Листинг программы

using System;

using System.Collections.Generic;

using System.IO;

using System.Runtime.Serialization;

using System.Runtime.Serialization.Json;

using System.Xml.Serialization;

using System.Threading;

namespace ConsoleApp12

{

[DataContract]

public class Card

{

public static List<Person> persons = new List<Person>();

public static List<Card> checkPin = new List<Card>();

[DataMember]

public string CardNumber { get; set; }

[DataMember]

public string Pin { get; set; }

public int Balance { get; set; }

static public void Menu(int cardIndex)

{

Console.Clear();

Console.WriteLine("1. Снять наличные");

Console.WriteLine("2. Внести наличные");

Console.WriteLine("3. Узнать баланс");

Console.WriteLine("4. Выйти");

int command = Convert.ToInt32(Console.ReadLine());

switch (command)

{

case 1:

{

WithdrawCash(cardIndex);

Menu(cardIndex);

break;

}

case 2:

{

DepositCash(cardIndex);

Menu(cardIndex);

break;

}

case 3:

{

FindBalance(cardIndex);

Menu(cardIndex);

break;

}

case 4:

{

break;

}

}

}

static public string InputNumCard()

{

var numCard = "0";

do

{

Console.Clear();

Console.WriteLine("Введите номер карты:");

numCard = Console.ReadLine();

} while (numCard.Length != 16);

return numCard;

}

static public string InputPin()

{

var pin = "0";

do

{

Console.Clear();

Console.WriteLine("Введите PIN:");

pin = Console.ReadLine();

} while (pin.Length != 4);

return pin;

}

static public void WithdrawCash(int cardIndex)

{

Console.Clear();

AcceptableBanknotesOnScreen();

Console.WriteLine("Введите сумму, которую хотите снять:");

bool correctSum = false;

int sum;

do

{

sum = Convert.ToInt32(Console.ReadLine());

if (sum % 10 == 0 && sum >= 10)

correctSum = true;

else

{

Console.Clear();

Console.WriteLine("Сумма некорректна\n");

AcceptableBanknotesOnScreen();

Console.WriteLine("Введите сумму, которую хотите снять:");

}

} while (!correctSum);

if (persons[cardIndex].Balance >= sum)

{

persons[cardIndex].Balance -= sum;

Console.Clear();

Console.WriteLine("Сумма " + sum + " рублей успешно списана\n");

Console.WriteLine("Для возврата в меню нажмите любую клавишу");

Console.ReadKey();

}

else

{

Console.Clear();

Console.WriteLine("На балансе не достаточно средств\n");

Console.WriteLine("Для возврата в меню нажмите любую клавишу");

Console.ReadKey();

}

}

static public void DepositCash(int cardIndex)

{

Console.Clear();

AcceptableBanknotesOnScreen();

Console.WriteLine("Введите сумму, которую хотите внести:");

bool correctSum = false;

int sum;

do

{

sum = Convert.ToInt32(Console.ReadLine());

if (sum % 10 == 0 && sum >= 10)

correctSum = true;

else

{

Console.Clear();

Console.WriteLine("Сумма некорректна\n");

AcceptableBanknotesOnScreen();

Console.WriteLine("Введите сумму, которую хотите внести:");

}

} while (!correctSum);

persons[cardIndex].Balance += sum;

Console.Clear();

Console.WriteLine("Сумма " + sum + " рублей успешно внесена\n");

Console.WriteLine("Для возврата в меню нажмите любую клавишу");

Console.ReadKey();

}

static public void FindBalance(int cardIndex)

{

Console.Clear();

Console.WriteLine("На Вашем балансе " + persons[cardIndex].Balance + " рублей" + "\n");

Console.WriteLine("Для возврата в меню нажмите любую клавишу");

Console.ReadKey();

}

static public void AcceptableBanknotesOnScreen()

{

Console.WriteLine("Допустимые банкноты:\n");

Console.WriteLine("10 рублей");

Console.WriteLine("50 рублей");

Console.WriteLine("100 рублей");

Console.WriteLine("200 рублей");

Console.WriteLine("500 рублей");

Console.WriteLine("1000 рублей");

Console.WriteLine("2000 рублей");

Console.WriteLine("5000 рублей\n");

}

static public int CardSearch(Card card)

{

int cardIndex = -1;

for (int i = 0; i < checkPin.Count; i++)

{

if (checkPin[i].CardNumber == card.CardNumber && checkPin[i].Pin == card.Pin)

{

cardIndex = i;

break;

}

}

return cardIndex;

}

public virtual void Greeting() { }

}

public class Person : Card

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string MiddleName { get; set; }

public override void Greeting()

{

Console.Clear();

Console.WriteLine("Здравствуйте, " + LastName + " "+ FirstName + " " + MiddleName + "!\n");

Thread.Sleep(2000);

Console.Clear();

}

}

class Program

{

static XmlSerializer xmlFormatter = new XmlSerializer(typeof(List<Person>));

static DataContractJsonSerializer jsonFormatter = new DataContractJsonSerializer(typeof(List<Card>));

static public void Serialization()

{

using (var file = new FileStream("persons.xml", FileMode.OpenOrCreate))

{

xmlFormatter.Serialize(file, Card.persons);

}

}

static public void Deserialization()

{

using (var file = new FileStream("persons.xml", FileMode.OpenOrCreate))

{

Card.persons = xmlFormatter.Deserialize(file) as List<Person>;

}

using (var file = new FileStream("cards.json", FileMode.OpenOrCreate))

{

Card.checkPin = jsonFormatter.ReadObject(file) as List<Card>;

}

}

static public int InsertCard()

{

Console.WriteLine("Нажмите 1, чтобы вставить карту:");

int command = 0;

try

{

command = Convert.ToInt32(Console.ReadLine());

}

catch

{

Console.Clear();

Console.WriteLine("Некорректный ввод");

}

return command;

}

static public void Authorization()

{

if (InsertCard() == 1)

{

var card = new Card() { CardNumber = Card.InputNumCard(), Pin = Card.InputPin() };

bool correctPin = false;

foreach (var tempCard in Card.checkPin)

{

if (tempCard.CardNumber == card.CardNumber && tempCard.Pin == card.Pin)

{

correctPin = true;

break;

}

}

if (correctPin)

{

int cardIndex = Card.CardSearch(card);

Card.persons[cardIndex].Greeting();

Card.Menu(cardIndex);

}

else

{

Console.WriteLine("Введен неправильный PIN");

}

}

}

static void Main(string[] args)

{

Deserialization();

Authorization();

Serialization();

}

}

}