ИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

ДОНЕЦКОЙ НАРОДНОЙ РЕСПУБЛИКИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«Донецкий нацИональнЫй технИчЕСКИЙ унИверситет»

Кафедра «ПРогрАММНАЯ ИНЖЕНЕРИЯ» им. Л. П.Фельдмана

Утверждаю

Зори С.А.

 \_\_\_\_\_\_\_\_\_\_\_

08. 02.2022 г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

к курсовому проекту по дисциплине

"Объектно-ориентированное программирование"

на тему: "Объектно-ориентированная система

“Магазин мобильный телефонов” "

Руководитель: Выполнил:

Старший преподаватель каф. ПИ студент гр.ПИ-21в

Морозова О.В. Рустамов Владислав

Грищенко Д.А.

Ассистент каф. ПИ:

Артеменко О.Г.

 08. 02.2022 г. 08.02.2022 г.

Донецк 2022

РЕФЕРАТ

Цель работы – закрепление практических навыков самостоятельной постановки и решения задачи обработки данных с помощью ЭВМ средствами объектно-ориентированного программирования (ООП).

Для достижения поставленной цели курсового проекта необходимо решить следующие задачи:

* выполнить объектно-ориентрованный анализ предметной области;
* разработать комплекс статических и динамических моделей предметной области;
* осуществить техническое и рабочее проектирование объектно- ориентированной системы;
* выполнить программную реализацию и тестирование спроектированного программного комплекса.

Методы исследования – объектно-ориентированный анализ и проектирование, техническое и рабочее проектирование.

Объект исследования – процесс взаимодействия пользователя с игрой. Результаты работы – объектно-ориентированная система

StorePhones.exe для моделирования базовых игровых операций по улучшению игрового персонажа.

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, ПРОЕКТИРОВАНИЕ, ЯЗЫК UML, С++, КОНСОЛЬНАЯ ПРИЛОЖЕНИЕ.

СОДЕРЖАНИЕ

[Введение 5](#_bookmark0)

1. [Постановка задачи 6](#_bookmark1)
2. [Объектно-ориентированный анализ и проектирование 8](#_bookmark2)
   1. [Объектно-ориентированный анализ 8](#_bookmark3)
   2. [Объектно-ориентированное проектирование 12](#_bookmark4)
      1. [Диаграмма классов 12](#_bookmark5)
      2. [Диаграмма объектов 13](#_bookmark6)
      3. [Диаграммы состояний 14](#_bookmark7)
      4. [Диаграмма последовательности 15](#_bookmark8)
3. [Рабочее проектирование 16](#_bookmark9)
   1. [Модульная структура программы 17](#_bookmark10)
   2. [Структура файлов и каталогов программы 18](#_bookmark11)
   3. [Разработка классов 21](#_bookmark12)
      1. [Реализация инкапсуляции 21](#_bookmark13)
      2. [Реализация наследования 28](#_bookmark14)
      3. [Реализация полиморфизма 38](#_bookmark15)
         1. [Перегрузка операторов 38](#_bookmark16)
         2. [Абстрактные классы и виртуальные методы 39](#_bookmark17)
   4. [Тестирование работоспособности системы классов 40](#_bookmark18)
   5. [Обоснование выбора структур данных 42](#_bookmark19)
   6. [Обоснование выбора алгоритмов 42](#_bookmark20)

[Выводы 44](#_bookmark21)

[Перечень ссылок 45](#_bookmark22)

[Приложение А. Техническое задание 46](#_bookmark23)

[Приложение Б. Описание программы 47](#_bookmark24)

[Приложение В. Руководство программиста 48](#_bookmark25)

[Приложение Д. Руководство оператора 49](#_bookmark26)

[Приложение Е. Текст программы 55](#_bookmark27)

[Приложение Ж. Файлы исходных данных 56](#_bookmark28)

[Приложение К. Файлы выходных данных 62](#_bookmark29)

[Приложение Л. Экранные формы 63](#_bookmark30)

ВВЕДЕНИЕ

Объектно-ориентированное программирование предлагает новый мощный способ решения проблемы сложности программ. Вместо того чтобы рассматривать программу как набор последовательно выполняемых инстр- укций, в объектно-ориентированном программировании программа представляется в виде совокупности объектов, обладающих сходными свойствами и набором действий, которые можно с ними производить [1].

В центре объектно-ориентированного программирования находится понятие объекта. Объект – это сущность, которой можно посылать сообщения и которая может на них реагировать, используя свои данные.

Объект – это экземпляр класса. Данные объекта скрыты от остальной программы. Сокрытие данный называется инкапсуляцией.

Наличие инкапсуляции достаточно для объектности языка программирования, но еще не означает его объектно-ориентированности – для этого требуется наличие наследования.

Но даже наличие инкапсуляции и наследования не делает язык программирования в полной мере объектным с точки зрения объектно- ориентированного программирования. Основные преимущества ООП проявляются только в том случае, когда в языке программирования реализован полиморфизм подтипов – возможность единообразно обрабатывать объекты с различной реализацией при условии наличия общего интерфейса.

Объектно-ориентированное программирование объединило лучшие идеи структурированного с рядом мощных концепций, которые способствуют более эффективной организации программ. Ключевой принципом организации программ является использование “данных, управляющих доступом к коду”.

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Словесное описание предметной области

Эта абстрактная система демонстрирует основные функции сотового телефона, такие как: вызов, отправка сообщений (которые выполняются через класс SIM), подзарядка и т. д. Объекты класса SmartPhone расширяют эту функциональность с помощью таких функций, как: калькулятор, фотографирование, получить конфигурацию и т. д.

Кроме того, для большего разнообразия я добавил класс Shop и Citizen, который может покупать телефон и звонить другим людям.

Согласно заданию , предметная область включает базовый класс MobilePhone и класс SIM

Создать класс MobilePhone, включающий в себя следующие элементы:

- поле «производитель» char \* Vendor;

- поле «модель» char \* Model;

- класс включает объект класса SIM-карта SIM \* card;

- метод получения названия телефона (производитель + модель) char \* GetName();

- метод вызова void Call();

- метод отправки SMS void SendSMS();

- метод «вставить SIM» void InsertSIM(SIM card);

- конструктор с параметрами MobilePhone(char \* Vendor, char \* Model).

Создать класс SIM, содержащий в себе элементы:

- поле «номер» char \* Number;

- поле «оператор» char \* Operator;

- конструктор с параметрами SIM(char \* Operator, char \* Number).

В классе MobilePhone перегрузить оператор ~ (yнарный, оператор класса), новое действие - вынуть SIM-карту.

1. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ
   1. Объектно-ориентированный анализ

Основная идея объектно-ориентированного анализа и проектирования состоит в рассмотрении предметной области и логического решения задачи с точки зрения объектов.[2] Основное внимание уделяется определению и описанию объектов в терминах предметной области.

Производится описание функциональности и поведения, анализ требований и выделяются основные процессы, происходящие в моделируемой системе и их формулировка в виде прецедентов. При моделировании системы с помощью диаграммы прецедентов было определено действующие лицо и его взаимодействие с системой. Задача этого шага в определении видов деятельности участников процесса и составлении концептуальной модели, которая отражает различные категории элементов предметной области и ожидаемый функционал системы.

Требования к разрабатываемой объектно-ориентированной системе в целом представлены на диаграмме прецедентов использования (рис. 2.1).

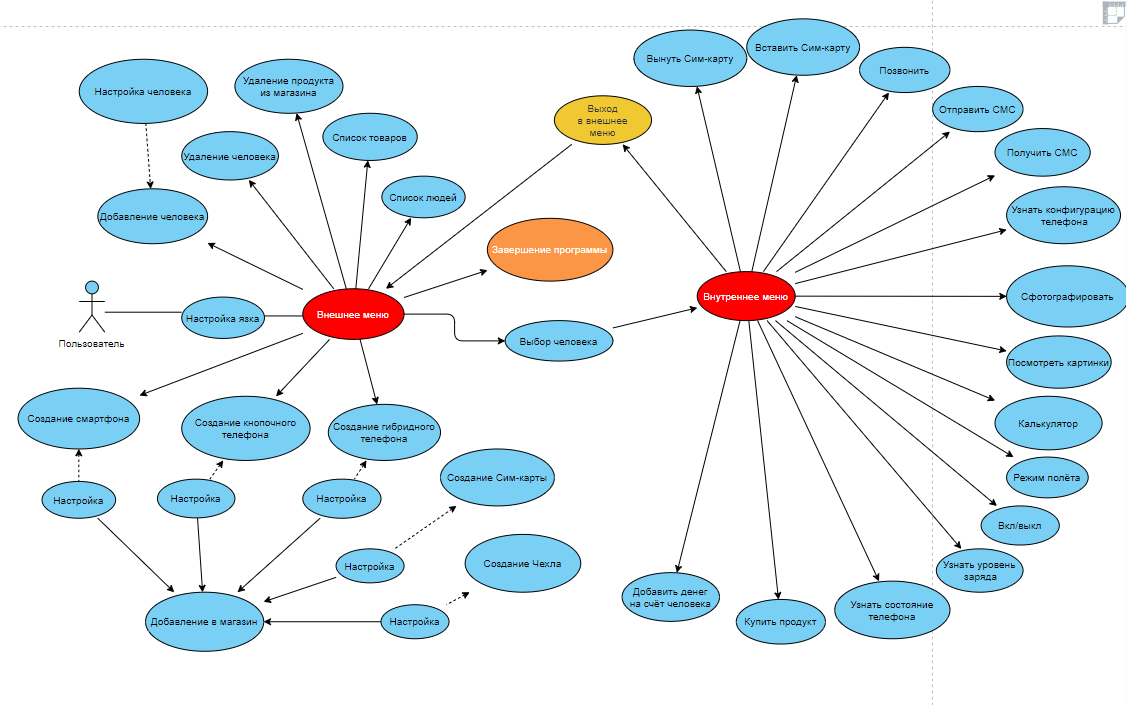


Рисунок 2.1 – Диаграмма прецедентов использования системы и вариантов использования процесса управления объектами

В соответствии с поставленными требованиями, Пользователь может перемещается по меню, из внешнего во внутреннее и наоборот(рис.2.1).

Находясь во внешнем меню, пользователь может создать 3 вида телефона в пункте “Создать Смартфон”, “Создать кнопочный телефона”, “Создать Гибридный телефон” и дать им свои параметры, все телефоны сразу добавляются в магазин. К телефонам можно создать Сим-карты и чехлы(под смартфон и гибридный телефон) в пунктах с соответствующими названиями, эти объекты также попадают в магазин. Также вы можете создать человека в пункте “Создать гражданина”, а также выбрать его в пункте “Выбрать гражданина” для перехода в внутреннее меню[2].Вы можете посмотреть список товаров и людей, а также при необходимости удалить объекты из категорий. [2]Во внутреннем меню пользователь может воспользоваться функциями телефона, продукт в магазине, начислить денег на счёт и выйти в внешнее меню “Выход”.

В ходе объектно-ориентированного анализа требований к системе были выделены дополнительные сущности, расширяющие моделируемую предметную область: PhoneMode, Shop, Person, Civil, Cover, Lang. Чтобы реализовать ромбовидное наследование были добавлены еще сущности расширяющие предметную область: SmartPhone, PushButtonPhone, HybridPhone. Результаты данного этапа декомпозиции представлены на аналитической диаграмме классов (рис. 2.2) в виде взаимосвязанных классов прикладных объектов концептуальной модели.

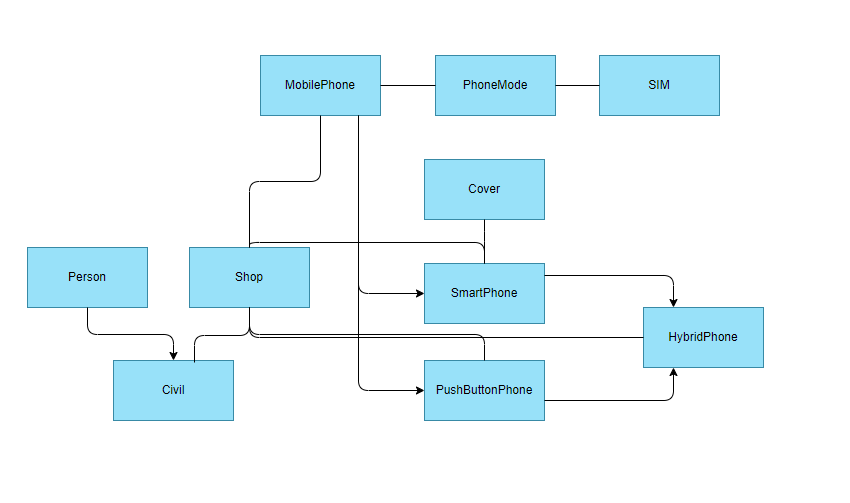


Рисунок 2.2 – Первый этап объектно-ориентированного анализа

(Взаимосвязь классов проектируемой системы)

Для навигации в нашей предметной области введем Lang(“языковой мессенджер”) KursovoyProekt(“Внешнее/внутреннее меню”)(рис. 2.3).

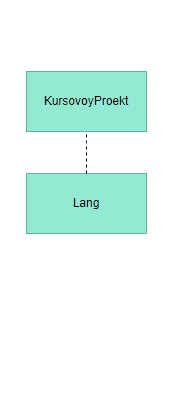


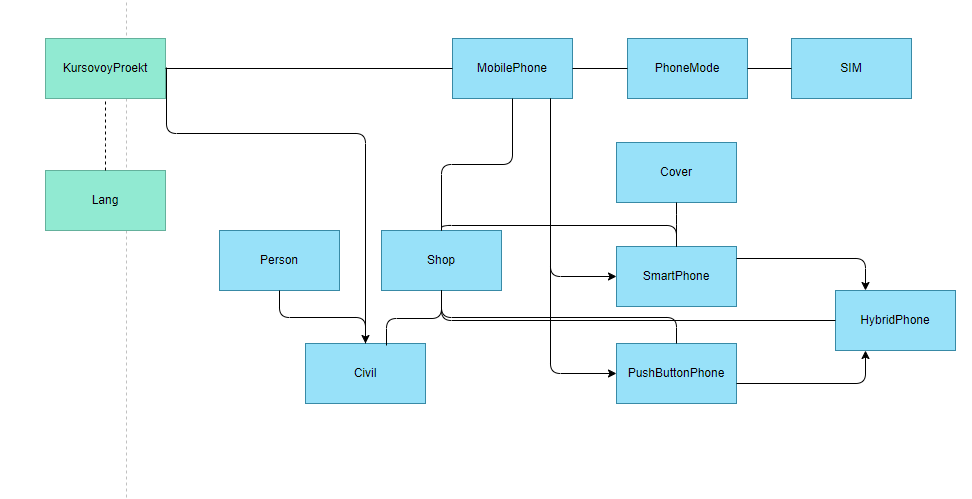
Рисунок 2.3 – Второй этап объектно-ориентированного анализа

(Взаимосвязь классов навигации)

Для реализации взаимодействия между телефонами нужно ввести человека , поэтому

расширим предметную область введя Person (“Человек”) который будет наследовать Civil.Совместим первый и второй этап.(рис 2.4)

Рисунок 2.4 –Третий этап объектно-ориентированного анализа



(Соединение взаимосвязей классов навигации и классов предметной области)

* 1. Объектно-ориентированное проектирование

Это метод создания таких моделей, которые фокусируют внимание на объектах, найденных в самой предметной области, и образуют объектно- ориентированную декомпозицию. То есть это разбиение системы на сущности, являющиеся какими-либо объектами действующими в той ситуации, которую как раз и моделирует система.[3] Эти объекты включают в себя атрибуты, свойства и методы.

* + 1. Диаграмма классов

Одним из первый этапов объектно-ориентированного проектирование является просмотрение статической диаграммы классов, потому что он рассматривает логические взаимосвязи классов между собой. Такая диаграмма демонстрирует классы системы, их атрибуты, методы и взаимосвязи между ними.

На рис. 2.5 изображена диаграмма классов для проектируемой объектно-ориентированной системы. На диаграмме показано взаимодействие классов (одиночное и множественное наследование (ромбовидное), агрегация, композиция, ассоциация). Классы имеют свои get-методы для получения доступа к полям, set-методы установки значений полям и аction- методы, которые реализуют поведение класса.

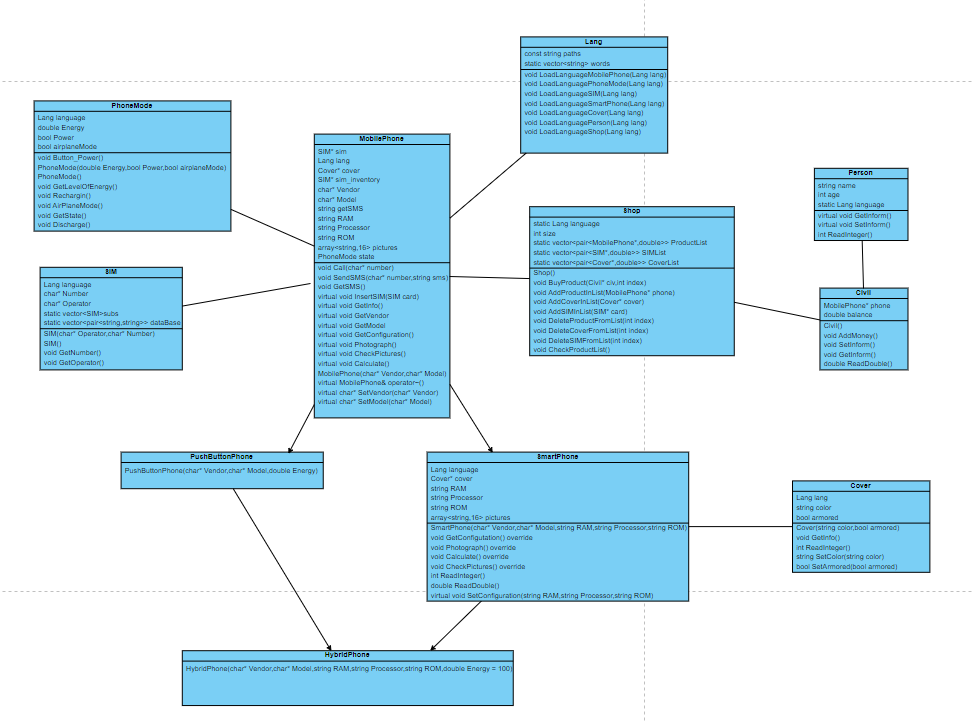


Рисунок 2.4 Диаграмма классов проектируемой системы.

* + 1. Диаграмма объектов

Диаграмма объектов выступает в качестве снимка объектов системы в определенный момент времени и показывает значения, которые могут принимать поля объектов.

Диаграмма объектов, которая отображает характерные значения атрибутов объектов заданной объектно-ориентированной системы, приведена на рис. 2.5.

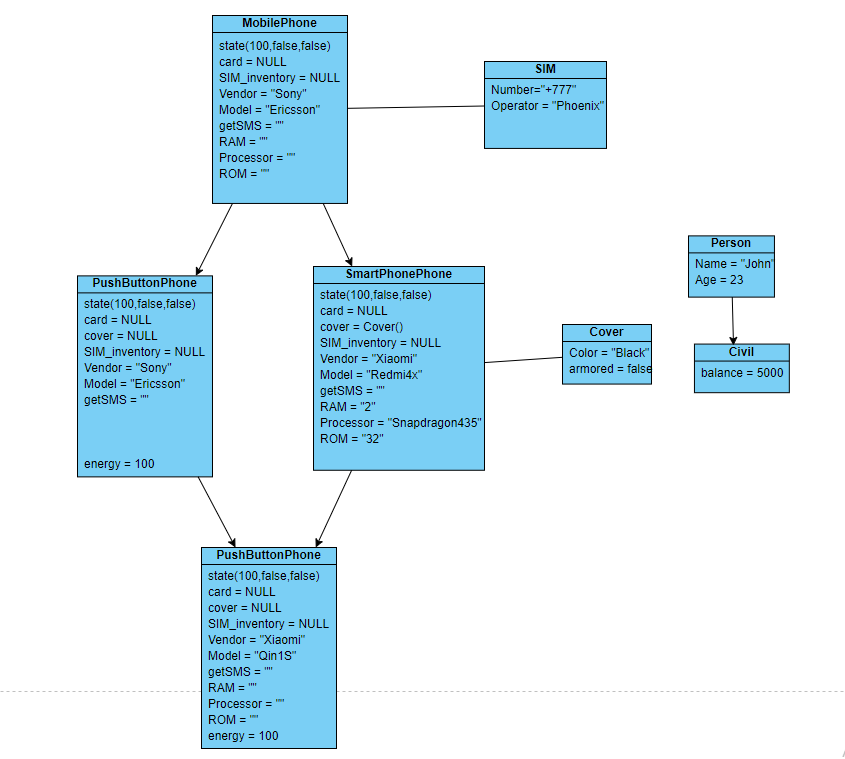


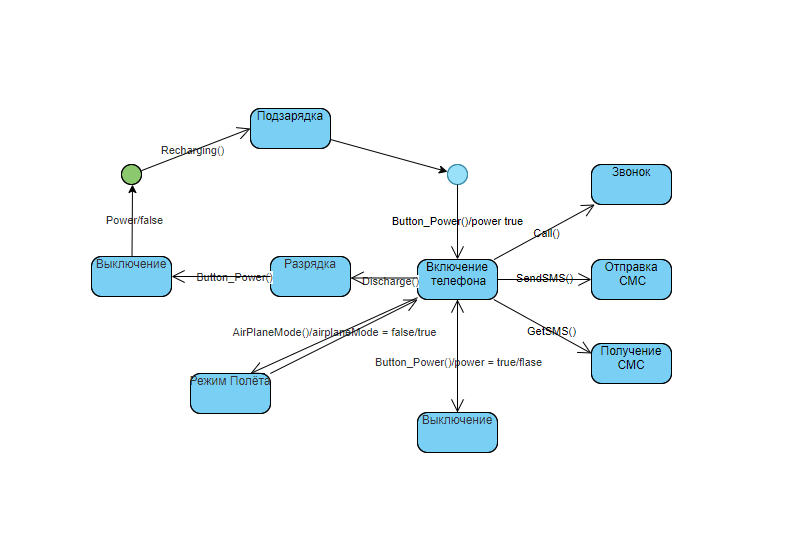
Рисунок 2.5 – Диаграмма объектов системы

* + 1. Диаграммы состояний

Диаграмма состояний показывает, как объект переходит из одного состояния в другое и служит для моделирования динамических аспектов системы.

Диаграмма состояний моделирует жизненный цикл и то в каких состояниях могут находиться объекты класса в процессе их жизненного цикла.

На рис. 2.6 показана диаграмма состояний для объектов класса MobilePhone



* + 1. Диаграмма последовательности

Диаграмма последовательности описывает взаимодействие системы в различный условиях их поведения и отображает порядок сообщений, которые передаются между объектами, упорядоченный по времени, с отражением продолжительности обработки и последовательности их проявления сообщений.[4] Это диаграмма для моделирования динамических аспектов объектно-ориентированного проектирования системы, которая описывает взаимодействие объектов, как обмен информацией между ними.

На рис. 2.6 показана диаграмма последовательности, опиcывающая основной сценарий создания гражданина, телефона и начала взаимодействия с ним.

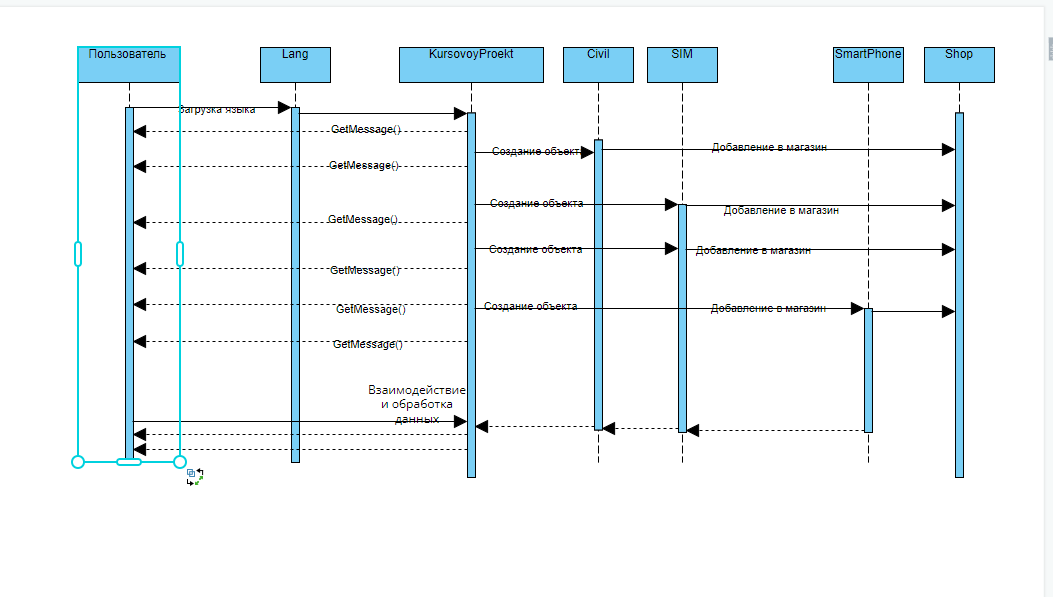


Рисунок 2.6 – Диаграмма последовательности для сценария гражданин, сим-карта и смартфон

1. РАБОЧЕЕ ПРОЕКТИРОВАНИЕ

Рабочее проектирование осуществляется на заключительной стадии процесса проектирования. На данном этапе осуществляется адаптация базовых средств программного обеспечения, обработка объектов (создание кода) и разработка программных модулей. Выполняется отладка, тестирование работоспособности программного кода, определение структуры программы, взаимодействие её модулей и разработку выбранного алгоритма. Основной задачей данного этапа является создание документации на программный продукт, включающий: описание программы, руководство программиста и руководство оператора.

* 1. Модульная структура программы

Разработанная система основывается на принципах раздельной компиляции, в соответствии с которым текст программы разделяется на модули. Каждому модулю соответствует свой класс системы, который предусматривает заголовочный файл, содержащий в себе интерфейс класса и файл реализации, содержащий в себе реализацию методов описанных в заголовочном файле (рис. 3.1).

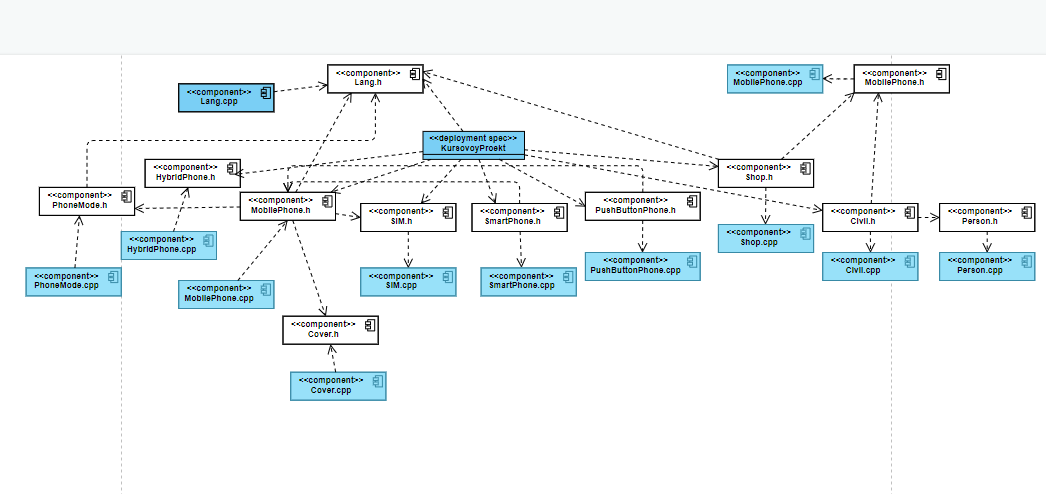


Рисунок 3.1 – Диаграмма модулей и её компонентов системы.

Преимущество раздельной компиляции состоит в том, что при изменение одного файла нет необходимости перекомпилировать заново всю программу. При работе со сложными проектами это экономит много времени.[5] Раздельная компиляция позволяет не знать как работает функция изнутри и позволяет распределить работу между несколькими программистами разделив модули на конкретные задачи.

* 1. Структура файлов и каталогов программы

На рис.3.2 изображена структура файлов программы. Файлы программы содержат в себе программные файлы и файлы данных. К программным файлам относятся заголовочные файлы и файлы реализации , а к файлам данных – файл конфигураций , файлы локализации , файл рейтинга и файл логов (ошибочных ситуаций).

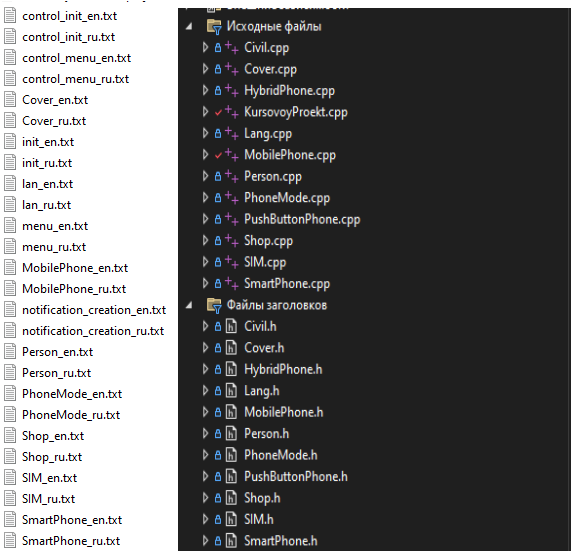


Рисунок 3.2 – Структура файлов программы

Файл конфигурации File.cfg, расположенный в корневом каталоге программы содержит в себе относительные либо абсолютные пути к файлам локализации. На рис. 3.3 изображен синтаксис для добавления файлов локализации в файл конфигурации. Это создано для того чтобы минимизировать вводимые пользователем данные.

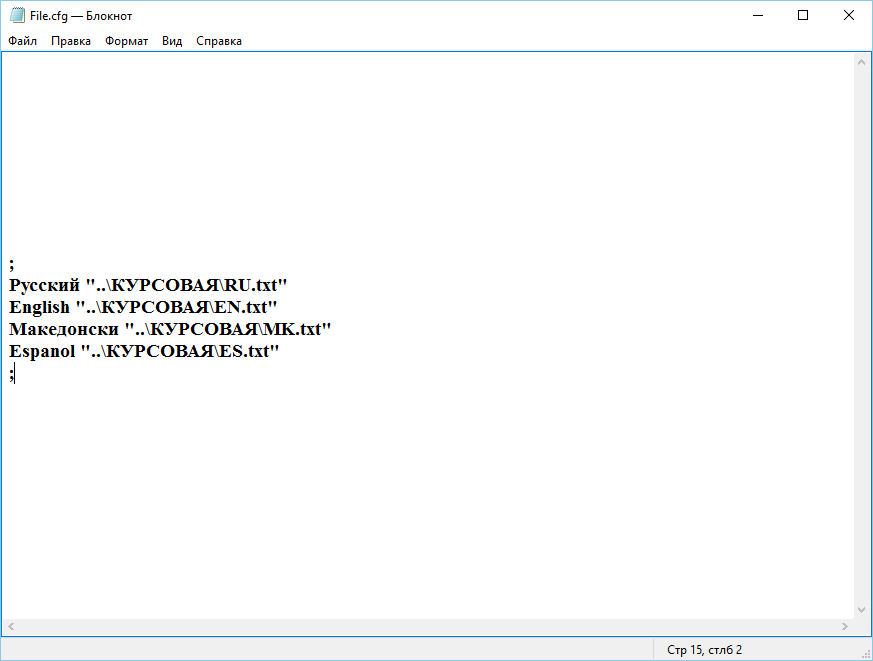


Рисунок 3.3 – Синтаксис файла конфигурации

Все место до и после открывающих кавычек “;” используется для заметок. Внутри кавычек для добавления своего файла локализации используется формат: <Название языка> <”Путь к файлу локализации”>.

Файлы локализации предназначены для реализации мультиязычности интерфейса программы и содержат тексты сообщений и декоративные элементы в виде текста, выдаваемых программой. Программа содержит 2 файла локализации: русский (…RU.txt), английский (…En.txt),. При запуске системы происходит загрузка данных из первого файла локализации в память представляющую собой вектора типа string

static vector<string> language;

static vector<string> init\_words;

static vector<string> control\_init\_words;

static vector<string> control;

static vector<string> MobilePhone\_lang;

static vector<string> PhoneMode\_lang;

static vector<string> SIM\_lang;

static vector<string> SmartPhone\_lang;

static vector<string> Cover\_lang;

static vector<string> Shop\_lang;

static vector<string> Person\_lang;

Вектор (массив строк) хранит в себе данные из файла до тех пор , пока пользователь не поменяет язык. После смены языка вектор очищается и в него загружаются файлы из файла который был выбран.

Такой подход позволит сократить количество ненужных данных в векторе и уменьшить память занимаемую вектором.

Файлы локализации содержат декоративные элементы и перевод для:

* Программных сообщений
* Информации “Описание предметной области”;
* Информации “Титульный лист”.

Для хранения перечисленной информации используются соответствующие файлы (рис. 3.3):

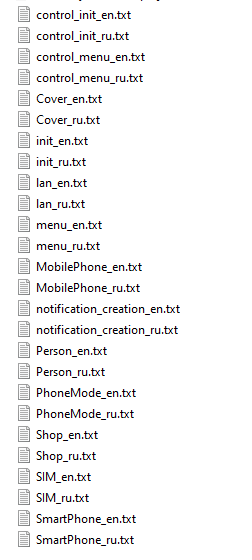


Рисунок 3.3 – файлы локализации.

В момент запуска программы текст из первого успешно открытого файла указанного в файле конфигурации загружается в вектор. После этого пользователь сможет поменять язык в главном меню.

При запуске система проверяет наличие файлов в каталогах указанных в файле конфигураций. Если все каталоги указанные в файле конфигурации введут к несуществующим файлам или файл конфигурации пуст , программа выдаст сообщение пользователю и завершит программу. Сообщение выводится на английском языке.

* 1. Разработка классов
     1. Реализация инкапсуляции

В объектно-ориентированном программировании инкапсуляцией называется механизм, который объединяет данные и код, манипулирует этими данными , а также защищает и то , и другое от внешнего вмешательства или неправильного использования.

Целью инкапсуляции является сокрытие реализаций каких-либо частей от внешнего мир. В таких языках как C++, С# и др. для инкапсуляции используют get- и set-методы.Get-методы для получения значения , set- методы для установки значения в защищенные или приватные поля класса. Поля должны использовать модификатор private или protected, а методы public.

На рис. 3.4 приведен пример реализации принципа инкапсуляции на примере класса SmartPhone.

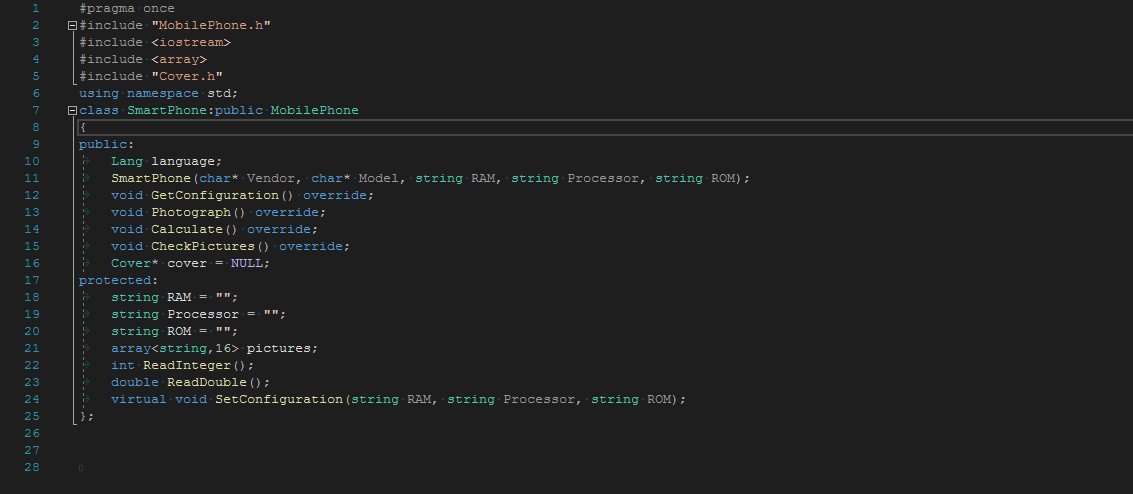


Рисунок 3.4 – Реализация принципа инкапсуляции для класса SmartPhone

Здесь доступ к полям класса возможен только через публичные get-и set-методы. Set-методы контролируют корректность введения и присваивания защищенным полям значений. Это позволяет обеспечивать безопасность хранимых в классе данных.

Подробная спецификация полей методов разработанных классов приведена в табл. 3.1-3.8.

Таблица 3.1 – Спецификация класса MobilePhone

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| PhoneMode state | Поле состояния телефона |
| Lang language | Поле локализации класса |
| SIM\* card | Поле хранения объекта Сим-карты для возможности использования методов Call,SendSMS,GetSMS |
| Cover\* cover | Поле хранения чехла телефона |
| SIM\* SIM\_inventory | Поле хранения неактивной Сим-карты |
| char\* Vendor | Поле названия производителя |
| char\* Model | Поле название модели телефона |
| string getSMS | Поле хранения получаемых сообщений |
| string RAM | Поле, к-ое хранит кол-во ОЗУ |
| string Processor | Поле, к-ое хранит название процессора |
| string ROM | Поле, к-ое хранит кол-во ПЗУ |
| array<string, 16> pictures | Массив, к-ый хранит название фотографий |
| void Call(char\* number) | Метод звонка |
| void SendSMS(char\* number, string sms) | Метод отправки сообщения |
| void GetSMS() | Метод получения сообщения |
| virtual void InsertSIM(SIM card) | Метод вставки Сим-карты |
| virtual void GetInfo() | Метод получения информации о телефоне |
| virtual void GetVendor() | Метод получения информации о производителе |
| virtual void GetModel() | Метод получения информации о модели |
| virtual void GetConfiguration() | Метод получения конфигурации |
| virtual void Photograph() | Метод фотографирования |
| virtual void CheckPictures() | Метод просмотра списка фотографий |
| virtual void Calculate() | Метод калькулятор |
| virtual MobilePhone& operator~() | Перегрузка метода ~(Вытащить Сим-карту) |
| virtual char\* SetVendor(char\* Vendor) | Метод установки производителя |
| virtual char\* SetModel(char\* Model) | Метод установки модели |

Таблица 3.2 – Спецификация класса PhoneMode

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| Lang language | Поле локализации класса |
| void Button\_Power() | Метод включения/отключения телефона |
| void GetLevelOfEnergy() | Метод получения инф. об уровне энергии |
| void Rechargin() | Метод подзарядки телефона |
| void AirPlaneMode() | Метод включения/отключения режима полёта |
| void GetState() | Метод получения информации о состоянии телефона |
| void Discharge() | Асинхронный метод разрядки телефона |
| double Energy | Поле кол-во энергии |
| bool Power | Поле состояния вкл/выкл |
| bool airplaneMode | Поле состояния режима полёта вкл/выкл |

Таблица 3.3 – Спецификация класса SIM

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| Lang language | Поле локализации класса |
| char\* Number | Поле хранения номера |
| char\* Operator | Поле хранения оператора |
| void GetNumber() | Метод получения номера |
| void GetOperator() | Метод получения оператора |
| static void GetToData(SIM sub) | Метод добавления абонента в вектор |
| static void GetToDataSMS(char\* number, string sms) | Метод добавления сообщения для абонента |
| static vector<SIM> subs | Вектор с хранением абонентов |
| static vector<pair<string, string>> dataBase | Вектор хранения сообщений для абонентов |

Таблица 3.4 – Спецификация абстрактного класса SmartPhone

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| Lang language | Поле локализации класса |
| void GetConfiguration() override | Метод получения конфигурации |
| void Photograph() override | Метод фотографирования |
| void Calculate() override | Метод производящий вычисления на калькуляторе |
| void CheckPictures() override | Метод проверки списка фотографий |
| Cover\* cover | Поле хранения чехла |
| string RAM | Поле кол-ва ОЗУ |
| string Processor | Поле названия процессора |
| string ROM | Поле кол-ва ПЗУ |
| array<string,16> pictures | Массив хранения названия фотографий |
| int ReadInteger() | Проверка валидации введённых чисел |
| double ReadDouble() | Проверка валидации введённых чисел |
| virtual void SetConfiguration(string RAM, string Processor, string ROM) | Метод установки конфигурации |

Таблица 3.5 – Спецификация класса Cover

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| Lang lang | Поле локализации класса |
| void GetInfo() | Метод получения информации о чехле |
| int ReadInteger() | Метод проверки валидации введённых чисел |
| string color | Поле цвета чехла |
| bool armored | Поле категории чехла(Прочный true/false) |
| string SetColor(string color) | Метод установки цвета |
| bool SetArmored(bool armored) | Метод установки прочности |

Таблица 3.6 – Спецификация класса Shop

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| static Lang language | Поле локализации класса |
| void BuyProduct(Civil\* civ,int index) | Метод покупки товара |
| void AddProductInList(MobilePhone\* phone) | Метод добавления телефона в вектор |
| void AddCoverInList(Cover\* cover) | Метод добавления чехла в вектор |
| void AddSIMInList(SIM\* card) | Метод добавления сим-карты в вектор |
| void DeleteProductFromList(int index) | Метод удаления телефона в вектор |
| void DeleteCoverFromList(int index) | Метод удаления чехла в вектор |
| void DeleteSIMFromList(int index) | Метод удаления сим-карты в вектор |
| void CheckProductList() | Метод проверки списка товаров |
| int size | Поле хранения размера списка товаров |
| static vector<pair<MobilePhone\*,double>> ProductList | Вектор хранения телефона и цены |
| static vector<pair<SIM\*, double>> SIMList | Вектор хранения сим-карты и цены |
| static vector<pair<Cover\*, double>> CoverList | Вектор хранения чехла и цены |

Таблица 3.7 – Спецификация класса Person

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| static Lang language | Поле локализации класса |
| virtual void GetInform() | Метод получения информации |
| string name | Поле хранения имени |
| int age | Поле хранения возраста |
| virtual void SetInfom() | Метод установки информации |
| int ReadInteger() | Метод проверки валидации введённых чисел |

Таблица 3.8 – Спецификация класса Civil

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| MobilePhone\* phone | Поле хранения телефона |
| void AddMoney(double add) | Метод добавление денег на баланс |
| void SetInfom() override | Метод установки информации |
| void GetInform() override | Метод получения информаци |
| double ReadDouble() | Метод проверки валидации ведённых чисел |
| double balance | Поле хранения кол-ва средств |

* + 1. Реализация наследования

Наследование – это один из основных принцип объектно- ориентированного программирования. Это свойство объектов, которое позволяет порождать своих потомков. Объект-потомок автоматически наследует от родителя все поля и методы, может дополнять объекты новыми полями и заменять методы родителя или дополнять их. Принцип наследования решает проблему модификации свойств объекта, придает объектно-ориентированному программированию гибкость и позволяет поддерживать концепцию иерархии классов. Применение иерархии классов делает управляемыми большие потоки информации.

В программе было реализовано одиночное и ромбовидное наследование (множественное наследование).

Класс SmartPhone наследует и модифицирует абстрактный класс MobilePhone(рис. 3.5)

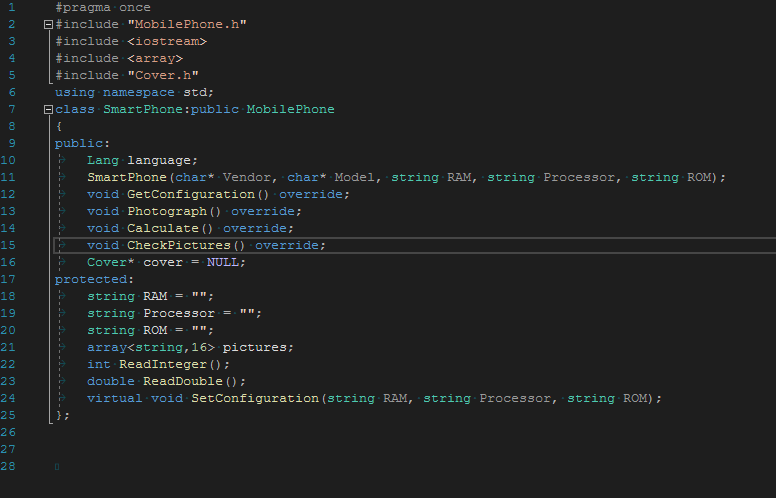


Рисунок 3.5 – Реализация одиночного наследования

Множественное наследование или ромбовидное наследование позволяет классу HybridPhone хранить и модифицировать в себе методы и свойства классов MobilePhone , SmartPhone, PushButtonPhone (рис. 3.6).

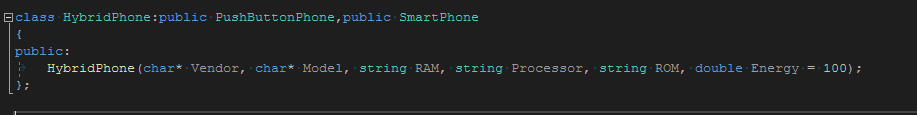


Рисунок 3.6 –Реализация ромбовидного наследования

* + 1. Реализация полиморфизма

Полиморфизм – это одна из парадигм объектно-ориентированного программирования. Это способность объекта использовать методы производного класса, который не существует на момент создания базового.

* + - 1. Перегрузка операторов

Перегрузка операторов – это описание в одной области видимости нескольких функций с одним и тем же именем [6] и возможность назначать новый смысл операторам при использовании их с определенным классом.

В курсовом проекте была выполнена дружественна перегрузка побитового оператора инверсия для класса MobilePhone.

Унарный оператор ~ перегружен и определяется соответственно в виде метода без явных параметров (рис. 3.8). Новым действие, выполняемое этим оператором – извлечение сим-карты.

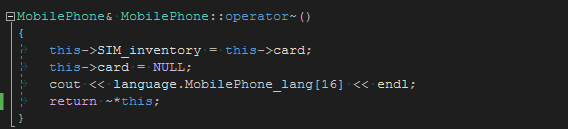


Рисунок 3.8 – Перегрузка оператора ~ внутри класса MobilePhone

* + - 1. Абстрактные классы и виртуальные методы

Абстрактный класс – это класс, который не предполагает создания экземпляров и имеет хотя бы один виртуальный метод. Абстрактные классы могут использоваться только в качестве базового для других классов.

В курсовом проекте абстрактным классом является класс MobilePhone.Он является абстрактным, поскольку содержит виртуальные методы (см. спецификацию). Эти методы определяются в классе SmartPhone и HybridPhone.



Рисунок 3.9 – Реализация абстрактного полиморфизма

* 1. Тестирование работоспособности системы классов

Система PhonesStore.exe обрабатывает такие ошибочные ситуации :

* Ошибки оператора
* Ошибки программиста
* Отсутствие файлов данных

При отсутствии файла конфигураций или файла локализации, пользователь получит соответствующие сообщение на английском языке, и выполнение программы завершится в аварийном режиме (рис. 3.10).

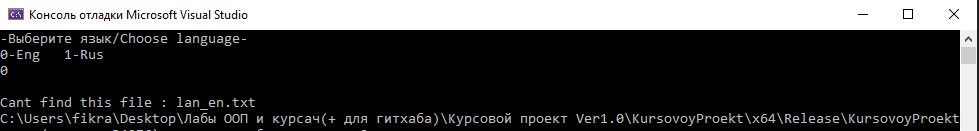


Рисунок 3.10 – Сообщение об отсутствии файла какого-либо файла с названием пути отсутствующего файла

Все вводимые пользователем данных проверяются некорректность. В случае неверно введенных данных пользователю будет выведено сообщение с ошибкой(рис. 3.11).

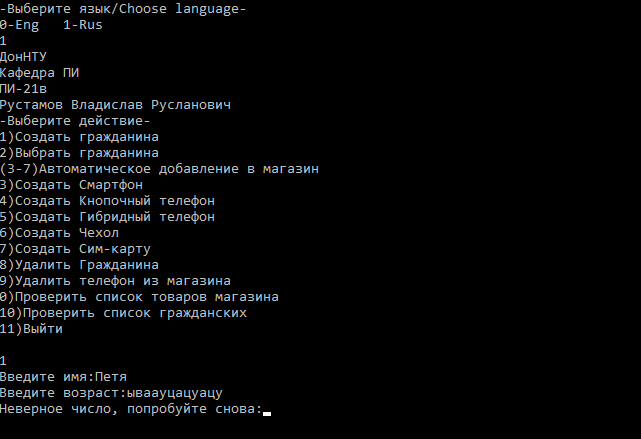


Рисунок 3.11 – Сообщение с ошибкой неверного ввода

* 1. Обоснование выбора структур данных

Для хранения данных из файла локализации используется vector<string> (массив строк ), который после каждой смены языка очищается и заполняется новыми файлами. Использование вектора в программе обусловлено тем, что вектор это динамическая структура данных, которая сужается или увеличивается самостоятельно , и у неё есть итераторы, что позволяет быстрее делать обход массива и при обходе сложнее выйти за границы.

Для удобства пользователя все пути к файлам локализации хранятся в файле конфигурации .

Векторы являются наиболее подходящим типом контейнера для последовательности, когда на первом месте стоит производительность.

* 1. Обоснование выбора алгоритмов

Одним из основных алгоритмов программы является алгоритм взаимодействие пользователя с меню. Переход по пунктам меню осуществляется при нажатии определенных клавиш.

После запуска программы в память подгружаются данные из файла локализации для её корректной работы. В случае если файлы небыли найдены или их не удалось открыть, будет выведено сообщение об ошибке и программа завершит свою работу. Если никаких ошибок не возникло в ходе загрузки данных из файла локализации, происходит отображение всей нужной информации. Использование такого подхода сократит обращение к данным из файла, что в свою очередь увеличит производительность программы.

Содержимое файлов локализации загружается и хранится в памяти до тех пор, пока пользователь не завершит программу или не решит поменять язык. Информация хранящаяся в памяти выводится по мере надобности на экран. После смены языка происходит задержка связанная с перезаписью данных в память , но она не существенная так, что она не влияет на производительность.

ВЫВОДЫ

Разработанная программа PhonesStore.exe является объектно- ориентированной которая моделирует предметную область «Консольный магазин мобильных телефонов». Она предназначена для использования в личных целях для отдыха.

В ходе создания системы были пройдены следующие этапы объектно- ориентированной разработки программного продукта:

* объектно-ориентированное проектирование и анализ предметной области;
* моделирование системы средствами UML;
* рабочее и техническое проектирование системы;
* объектно-ориентированное программирование на языке С++;
* системное тестирование и оценка качества ПО;
* создание проектной, технической и пользовательской документации.

Разработанная объектно-ориентированная система соответствует поставленным требованиям, реализует поставленную задачу.

ПЕРЕЧЕНЬ ССЫЛОК

1. Лафоре Р., Объектно-ориентированное программирование С++. – СПб:Питер, 2004. – 920с: ил.
2. Анализ и проектирование.Конспект лекций [Электронный ресурс]. – Режим доступа: ooad.asf.ru/students/lecture\_risp/lec012.aspx
3. Объектно-ориентированная декомпозиция.[Электронный ресурс].- Режим доступа: fkn.ktu10.com/?q=node/2301
4. Фаулер М., UML. Основы. – СПб:Символ плюс, 2004. – 192с: ил.
5. Полный справочник по С [Электронный ресурс]. – Режим доступа: lord- n.narod.ru/download/books/walla/programming/spr\_po\_C/01/0108.htm
6. Волкова И.А., Иванов Л.Е., Карпов Л.Е., Основы объектно- ориентированного программирования. Язык программирования С++. – СПб:Москва, 2011 – 112с: ил.

ПРИЛОЖЕНИЕ А ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Приложение Б

Описание программы

Б.1 Общие сведения

Программа PhonesStore.exe предназначена для моделирования процесса взаимодействия человека с телефоном.

Б.2 Функциональное назначение

Программа предназначена для использования в личных целях. Б.3 Используемые технические средства

Для запуска программы требуется персональный компьютер с частотой

процессора не менее 1.2 ГГц и объемом оперативной памяти не менее 128 Мб.

Б.4 Вызов и загрузка

Вызов и загрузка программы осуществляется путем запуска исполняемого файла PhonesStore.exe.

Б.5 Входные данные

В качестве исходных данных программа использует данные, считываемые из внешних файлов, а также вводимые пользователем с клавиатуры.

Приложение В Руководство программиста

Файл конфигурации программы File.cfg состоит из кавычек,точки с запятой и строки с названием языка и в кавычках пути к файлу локализации. Название языка и путь к файлу локализации и надо указывать на одной строке и внутри кавычек.

Содержимое файлов локализации разбито на два раздела:

* Название языка;
* Путь к файлу локализации;

Приложение Д Руководство оператора

Д.1 Назначение программы

Программа предназначена для личного использования. Д.2 Условия выполнения программы

Для выполнения программы необходим ПК функционирующий под управлением операционной системы семейства Windows (не ниже Windows XP).

Д.3 Выполнение программы

При запуске приложения появляется заставка, которая выводит титульный лист пояснительной записки к курсовому проекту. Для переключения языка пользователь должен быть в Главное меню, выбрать пункт “Язык” и выбрать требуемый язык из списка. Для выхода программы следует нажать клавишу ESC в главном меню или нажать требуемую клавишу чтобы выйти в главное меню.

Выбор необходимого пункта меню осуществляется путем нажатия клавиш с числами (1-12). Для возврата в предыдущее меню используется клавиша ESC или 0. При нажатии 11 в главном меню программа завершит свою работу.

Приложение Е

Текст программы

#include <iostream>

#include "MobilePhone.h"

#include "SIM.h"

#include <stdio.h>

#include <thread>

#include <chrono>

#include "SmartPhone.h"

#include "Shop.h"

#include "Civil.h"

#include "PushButtonPhone.h"

#include "Lang.h"

#include <fstream>

#include "HybridPhone.h"

#include <conio.h>

#include <filesystem>

using namespace std;

void Menu();

void Control(int index);

bool IsPhoneExsist(int index, Lang lang);

vector<Civil> civils;

Shop shop;

Lang lang;

int ReadInteger() {

int number;

char c;

while (1) {

if (!scanf\_s("%d", &number) || getchar() != '\n') {

while ((c = getchar()) != '\n' && c != EOF);

}

else {

return number;

}

printf\_s("Incorrect number.\nTry again: ");

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Menu();

}

void Menu() {

string temp\_ru;

string temp\_en;

ifstream fin;

ifstream init\_fin;

ifstream finTemp;

cout << "-Выберите язык/Choose language-\n0-Eng\t1-Rus" << endl;

int i = -1;

while (i < 0 || i > 1) {

i = ReadInteger();

}

switch (i) {

case 1:

fin.open(lang.pathRu);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathRu;

return;

}

while (!fin.eof()) {

temp\_ru = "";

getline(fin, temp\_ru);

lang.language.push\_back(temp\_ru);

lang.lang = 1;

}

if (!init\_fin) {

cout << "\nCant find this file : " << lang.pathRu\_init;

break;

}

init\_fin.open(lang.pathRu\_init);

while (!init\_fin.eof()) {

temp\_ru = "";

getline(init\_fin, temp\_ru);

lang.init\_words.push\_back(temp\_ru);

}

lang.LoadLanguageMobilePhone(lang);

lang.LoadLanguagePhoneMode(lang);

lang.LoadLanguageSIM(lang);

lang.LoadLanguageSmartPhone(lang);

lang.LoadLanguageCover(lang);

lang.LoadLanguageShop(lang);

lang.LoadLanguagePerson(lang);

break;

case 0:

fin.open(lang.pathEn);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathEn;

return;

}

while (!fin.eof()) {

temp\_en = "";

getline(fin, temp\_en);

lang.language.push\_back(temp\_en);

lang.lang = 0;

}

if (!init\_fin) {

cout << "\nCant find this file : " << lang.pathEn\_init;

break;

}

init\_fin.open(lang.pathEn\_init);

while (!init\_fin.eof()) {

temp\_en = "";

getline(init\_fin, temp\_en);

lang.init\_words.push\_back(temp\_en);

}

lang.LoadLanguageMobilePhone(lang);

lang.LoadLanguagePhoneMode(lang);

lang.LoadLanguageSIM(lang);

lang.LoadLanguageSmartPhone(lang);

lang.LoadLanguageCover(lang);

lang.LoadLanguageShop(lang);

lang.LoadLanguagePerson(lang);

break;

}

for (int i = 0; i < lang.language.size(); i++) {

cout << lang.language[i] << endl;

}

fin.close();

lang.language.clear();

int choose;

int indexOfCiv = -1;

while (true) {

if (lang.lang == 1) {

fin.open(lang.pathRu\_menu);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathRu\_menu;

return;

}

while (!fin.eof()) {

temp\_ru = "";

getline(fin, temp\_ru);

lang.language.push\_back(temp\_ru);

}

}

else if (lang.lang == 0) {

fin.open(lang.pathEn\_menu);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathEn\_menu;

return;

}

while (!fin.eof()) {

temp\_en = "";

getline(fin, temp\_en);

lang.language.push\_back(temp\_en);

}

}

for (int i = 0; i < lang.language.size(); i++) {

cout << lang.language[i] << endl;

}

fin.close();

lang.language.clear();

if (lang.lang == 1) {

fin.open(lang.pathRu\_not);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathRu\_not;

return;

}

while (!fin.eof()) {

temp\_ru = "";

getline(fin, temp\_ru);

lang.language.push\_back(temp\_ru);

}

}

else if (lang.lang == 0) {

fin.open(lang.pathEn\_not);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathEn\_not;

return;

}

while (!fin.eof()) {

temp\_en = "";

getline(fin, temp\_en);

lang.language.push\_back(temp\_en);

}

}

choose = ReadInteger();

if (choose == 11)

break;

switch (choose) {

case 1:

{

Civil civ;

civils.push\_back(civ);

cout << lang.language[0] << endl;

break;

}

case 2:

if (civils.empty()) {

cout << lang.language[11] << endl;

break;

}

cout << lang.language[1] << endl;

indexOfCiv = -1;

while (indexOfCiv < 0 || indexOfCiv > civils.size()) {

indexOfCiv = ReadInteger();

}

Control(indexOfCiv-1);

break;

case 3:

{

char\* vend = new char[32];

char\* model = new char[32];

string ram;

string rom;

string processor;

cout << lang.init\_words[0] << endl;

cin >> vend;

cout << lang.init\_words[1] << endl;

cin >> model;

cout << lang.init\_words[2] << endl;

cin >> ram;

cout << lang.init\_words[3] << endl;

cin >> processor;

cout << lang.init\_words[4] << endl;

cin >> rom;

shop.AddProductInList(new SmartPhone(vend, model, ram, processor, rom));

cout << lang.language[2] << endl;

}

break;

case 4:

{

char\* vend = new char[32];

char\* model = new char[32];

string ram;

string rom;

string processor;

cout << lang.init\_words[0] << endl;

cin >> vend;

cout << lang.init\_words[1] << endl;

cin >> model;

shop.AddProductInList(new PushButtonPhone(vend,model,100));

cout << lang.language[3] << endl;

break;

}

case 5:

{

char\* vend = new char[32];

char\* model = new char[32];

string ram;

string rom;

string processor;

cout << lang.init\_words[0] << endl;

cin >> vend;

cout << lang.init\_words[1] << endl;

cin >> model;

cout << lang.init\_words[2] << endl;

cin >> ram;

cout << lang.init\_words[3] << endl;

cin >> processor;

cout << lang.init\_words[4] << endl;

cin >> rom;

shop.AddProductInList(new HybridPhone::SmartPhone(vend, model, ram, processor, rom));

cout << lang.language[4] << endl;

break;

}

case 6:

{

string color;

bool armored;

cout << lang.init\_words[5] << endl;

getline(cin,color);

cout << lang.init\_words[6] << endl;

armored = ReadInteger();

shop.AddCoverInList(new Cover(color,armored));

cout << lang.language[5] << endl;

break;

}

case 7:

{

char\* number = new char[32];

char\* Operator = new char[32];

cout << lang.init\_words[7] << endl;

cin >> number;

cout << lang.init\_words[8] << endl;

cin >> Operator;

shop.AddSIMInList(new SIM(number, Operator));

cout << lang.language[6] << endl;

break;

}

case 8:

{

cout << lang.language[1] << endl;

indexOfCiv = -1;

while (indexOfCiv < 0 || indexOfCiv > civils.size()) {

indexOfCiv = ReadInteger();

}

civils.erase(civils.begin() + indexOfCiv - 1);

cout << lang.language[7] << endl;

break;

}

case 9:

{

cout << lang.language[1] << endl;

while (indexOfCiv < 0 || indexOfCiv > shop.size) {

indexOfCiv = ReadInteger();

}

shop.DeleteProductFromList(indexOfCiv);

break;

}

case 0:

{

shop.CheckProductList();

Sleep(8000);

break;

}

case 10:

{

cout << lang.language[9] << endl;

for (auto i = 0; i < civils.size(); i++) {

cout << i + 1 << ")";

civils[i].GetInform();

}

Sleep(8000);

break;

}

case 12:

finTemp.open(lang.pathDescription);

if (!finTemp.is\_open()) {

cout << "\nCant find this file : " << lang.pathDescription;

break;

}

while (!finTemp.eof()) {

temp\_ru = "";

getline(finTemp, temp\_ru);

cout << temp\_ru << endl;

}

finTemp.close();

break;

default:

cout << lang.language[10] << endl;

break;

}

Sleep(1500);

fin.close();

lang.language.clear();

system("CLS");

}

}

void Control(int index) {

system("CLS");

ifstream control\_menu\_fin;

ifstream control\_init\_fin;

string temp\_ru;

string temp\_en;

SIM card\_temp;

if (lang.lang == 1) {

control\_menu\_fin.open(lang.pathRu\_control\_menu);

if (!control\_menu\_fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathRu\_control\_menu;

return;

}

while (!control\_menu\_fin.eof()) {

temp\_ru = "";

getline(control\_menu\_fin, temp\_ru);

lang.control.push\_back(temp\_ru);

}

}

else if (lang.lang == 0) {

control\_menu\_fin.open(lang.pathEn\_control\_menu);

if (!control\_menu\_fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathEn\_control\_menu;

return;

}

while (!control\_menu\_fin.eof()) {

temp\_en = "";

getline(control\_menu\_fin, temp\_en);

lang.control.push\_back(temp\_en);

}

}

while (true) {

for (auto i = 0; i < lang.control.size(); i++) {

cout << lang.control[i] << endl;

}

if (lang.lang == 1) {

control\_init\_fin.open(lang.pathRu\_control\_init);

if (!control\_init\_fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathRu\_control\_init;

return;

}

while (!control\_init\_fin.eof()) {

temp\_ru = "";

getline(control\_init\_fin, temp\_ru);

lang.control\_init\_words.push\_back(temp\_ru);

}

}

else if (lang.lang == 0) {

control\_init\_fin.open(lang.pathEn\_control\_init);

if (!control\_init\_fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathEn\_control\_init;

return;

}

while (!control\_init\_fin.eof()) {

temp\_en = "";

getline(control\_init\_fin, temp\_en);

lang.control\_init\_words.push\_back(temp\_en);

}

}

int choose;

int indexOfProduct = -1;

char number[32];

int add = 0;

choose = ReadInteger();

if (choose == 0) {

break;

}

switch (choose) {

case 1:

if (IsPhoneExsist(index,lang)) {

break;

}

civils[index].phone->InsertSIM(card\_temp);

break;

case 2:

civils[index].phone->operator~();

break;

case 3:

{

if (IsPhoneExsist(index, lang)) {

break;

}

cout << lang.control\_init\_words[0] << endl;

cin >> number;

civils[index].phone->Call(number);

Sleep(3000);

break;

}

case 4:

{

if (IsPhoneExsist(index, lang)) {

break;

}

string sms;

cout << lang.control\_init\_words[0] << endl;

cin >> number;

cout << lang.control\_init\_words[1] << endl;

cin>>sms;

civils[index].phone->SendSMS(number, sms);

break;

}

case 5:

if (IsPhoneExsist(index, lang)) {

break;

}

civils[index].phone->GetSMS();

break;

case 6:

if (IsPhoneExsist(index, lang)) {

break;

}

civils[index].phone->GetConfiguration();

break;

case 7:

if (IsPhoneExsist(index, lang)) {

break;

}

civils[index].phone->Photograph();

break;

case 8:

if (IsPhoneExsist(index, lang)) {

break;

}

civils[index].phone->CheckPictures();

break;

case 9:

if (IsPhoneExsist(index, lang)) {

break;

}

civils[index].phone->Calculate();

break;

case 10:

if (IsPhoneExsist(index, lang)) {

break;

}

civils[index].phone->state.Button\_Power();

break;

case 11:

if (IsPhoneExsist(index, lang)) {

break;

}

civils[index].phone->state.GetLevelOfEnergy();

break;

case 12:

if (IsPhoneExsist(index, lang)) {

break;

}

civils[index].phone->state.Rechargin();

break;

case 13:

if (IsPhoneExsist(index, lang)) {

break;

}

civils[index].phone->state.AirPlaneMode();

break;

case 14:

if (IsPhoneExsist(index, lang)) {

break;

}

civils[index].phone->state.GetState();

break;

case 15:

cout << lang.language[1] << endl;

while (indexOfProduct < 0) {

indexOfProduct = ReadInteger();

}

shop.BuyProduct(&civils[index], indexOfProduct);

break;

case 16:

cout << lang.control\_init\_words[2] << endl;

while (add <= 0) {

add = ReadInteger();

}

civils[index].AddMoney(add);

break;

default:

cout << lang.control\_init\_words[3] << endl;

}

Sleep(1500);

system("CLS");

}

lang.control.clear();

}

bool IsPhoneExsist(int index,Lang lang) {

if (civils[index].phone == NULL) {

cout << lang.control\_init\_words[4] << endl;

return true;

}

return false;

}

#pragma once

#include "SIM.h"

#include <iostream>

#include <string>

#include <stdio.h>

#include <windows.h>

#include "PhoneMode.h"

#include "Cover.h"

#include <array>

#include "Lang.h"

using namespace std;

class MobilePhone

{

friend class Shop;

public:

PhoneMode state = PhoneMode(100,false,false);

Lang language;

SIM\* card = NULL;

Cover\* cover = NULL;

SIM\* SIM\_inventory = NULL;

void Call(char\* number);

void SendSMS(char\* number, string sms);

void GetSMS();

virtual void InsertSIM(SIM card);

virtual void GetInfo();

virtual void GetVendor();

virtual void GetModel();

virtual void GetConfiguration();

virtual void Photograph();

virtual void CheckPictures();

virtual void Calculate();

MobilePhone(char\* Vendor, char\* Model);

virtual MobilePhone& operator~();

protected:

virtual char\* SetVendor(char\* Vendor);

virtual char\* SetModel(char\* Model);

char\* Vendor = new char();

char\* Model = new char();

string getSMS = "";

string RAM = "";

string Processor = "";

string ROM = "";

array<string, 16> pictures;

};

#include "MobilePhone.h"

#include "SIM.h"

#include <iostream>

#include <string>

#include <stdio.h>

#include <windows.h>

#include "PhoneMode.h"

using namespace std;

MobilePhone::MobilePhone(char\* Vendor, char\* Model) {

this->Vendor = SetVendor(Vendor);

this->Model = SetModel(Model);

/\*state = PhoneMode(100, false, false); \*/

}

MobilePhone& MobilePhone::operator~()

{

this->SIM\_inventory = this->card;

this->card = NULL;

cout << language.MobilePhone\_lang[16] << endl;

return \*this;

}

char\* MobilePhone::SetVendor(char\* Vendor) {

char\* vendor = Vendor;

while(vendor == NULL) {

cout << language.MobilePhone\_lang[0];

fgets(vendor, 32, stdin);

}

fflush(stdin);

cout << endl;

return vendor;

}

char\* MobilePhone::SetModel(char\* Model) {

char\* model = Model;

while(model == NULL) {

cout << language.MobilePhone\_lang[1];

fgets(model, 32, stdin);

}

fflush(stdin);

cout << endl;

return model;

}

void MobilePhone::GetVendor() {

cout << language.MobilePhone\_lang[2];

puts(this->Vendor);

}

void MobilePhone::GetModel() {

cout << language.MobilePhone\_lang[3];

puts(this->Model);

}

void MobilePhone::GetInfo() {

cout << language.MobilePhone\_lang[4] << endl;

GetVendor();

GetModel();

this->card->GetNumber();

this->card->GetOperator();

}

void MobilePhone::GetConfiguration()

{

}

void MobilePhone::Photograph()

{

}

void MobilePhone::Calculate()

{

}

void MobilePhone::CheckPictures()

{

}

void MobilePhone::Call(char\* number) {

SIM temp;

if (this->card == NULL) {

cout << language.MobilePhone\_lang[5] << endl;

return;

}

if (state.Power == true && state.airplaneMode == false) {

for (auto i = 0; i < temp.subs.size(); i++) {

if ((strcmp(number,temp.subs[i].Number) == 0) && strcmp(this->card->Operator, temp.subs[i].Operator) == 0) {

cout << language.MobilePhone\_lang[6] << endl;

Sleep(300);

cout << language.MobilePhone\_lang[7] << temp.subs[i].Number<<endl;

Sleep(1000);

cout << language.MobilePhone\_lang[8];

}

}

}

else {

cout << language.MobilePhone\_lang[9];

}

}

void MobilePhone::SendSMS(char\* number, string sms) {

SIM temp;

if (this->card == NULL) {

cout << language.MobilePhone\_lang[5] << endl;

return;

}

if (state.Power == true && state.airplaneMode == false) {

for (auto i = 0; i < temp.subs.size(); i++) {

if ((strcmp(number, temp.subs[i].Number) == 0) && strcmp(this->card->Operator, temp.subs[i].Operator) == 0) {

cout << language.MobilePhone\_lang[10] << endl;

temp.GetToDataSMS(number, sms);

Sleep(1000);

cout << language.MobilePhone\_lang[11];

}

}

}

else {

cout << language.MobilePhone\_lang[9];

}

}

void MobilePhone::GetSMS() {

SIM temp;

if (this->card == NULL) {

cout << language.MobilePhone\_lang[5] << endl;

return;

}

if (state.Power == true && state.airplaneMode == false) {

for (auto i = 0; i < temp.dataBase.size(); i++) {

if (this->card->Number == temp.dataBase[i].first) {

this->getSMS = temp.dataBase[i].second;

cout << language.MobilePhone\_lang[12] << getSMS << endl;

temp.dataBase.erase(temp.dataBase.begin() + i);

getSMS = "";

break;

}

} }

else {

cout << language.MobilePhone\_lang[9];

}

}

void MobilePhone::InsertSIM(SIM card) {

if (this->card != NULL) {

cout << language.MobilePhone\_lang[13] << endl;

return;

}

if (this->SIM\_inventory != NULL) {

this->card = this->SIM\_inventory;

this->card->GetToData(\*this->card);

cout << language.MobilePhone\_lang[14] << endl;

return;

}

else {

cout << language.MobilePhone\_lang[15] << endl;

}

}

#pragma once

#include <iostream>

#include <stdio.h>

#include <windows.h>

#include "Lang.h"

using namespace std;

class PhoneMode

{

friend class MobilePhone;

friend class SmartPhone;

friend class PushButtonPhone;

public:

Lang language;

void Button\_Power();

PhoneMode(double Energy,bool Power,bool airplaneMode);

PhoneMode();

void GetLevelOfEnergy();

void Rechargin();

void AirPlaneMode();

void GetState();

void Discharge();

protected:

double Energy;

bool Power;

bool airplaneMode;

};

#include "PhoneMode.h"

#include <thread>

#include <chrono>

PhoneMode::PhoneMode(double Energy, bool Power, bool airplaneMode) {

this->Energy = Energy;

this->Power = Power;

this->airplaneMode = airplaneMode;

}

PhoneMode::PhoneMode() {

}

void PhoneMode::AirPlaneMode() {

if (Power == true && airplaneMode == false) {

airplaneMode = true;

cout << language.PhoneMode\_lang[0] << endl;

}

else if (Power == true && airplaneMode == true) {

airplaneMode = false;

cout << language.PhoneMode\_lang[1] << endl;

}

else if (Power == false) {

cout << language.PhoneMode\_lang[2] << endl;

}

}

void PhoneMode::GetState()

{

cout << language.PhoneMode\_lang[3] << endl;

cout << language.PhoneMode\_lang[4] << this->Energy << "%" << endl;

cout << language.PhoneMode\_lang[5] << this->airplaneMode << endl;

cout << language.PhoneMode\_lang[6] << this->Power << endl;

}

void PhoneMode::Discharge()

{

for (; Energy > 0; Energy--) {

if (Power == false) {

return;

}

std::this\_thread::sleep\_for(chrono::milliseconds(500));

}

if (Energy == 0) {

Power = false;

}

}

void PhoneMode::Button\_Power() {

std::thread discharge;

if (Power == false && Energy > 0) {

Power = true;

cout << language.PhoneMode\_lang[7] << endl;

discharge = thread(&PhoneMode::Discharge, this);

discharge.detach();

}

else if (Power == false && Energy == 0) {

cout << language.PhoneMode\_lang[8] << endl;

}

else if (Power == true && Energy > 0) {

Power = false;

cout << language.PhoneMode\_lang[9] << endl;

}

else if (Power == true && Energy == 0) {

Power = false;

cout << language.PhoneMode\_lang[10] << endl;

}

}

void PhoneMode::GetLevelOfEnergy() {

cout << language.PhoneMode\_lang[11] << Energy << "%" << endl;

}

void PhoneMode::Rechargin() {

cout << language.PhoneMode\_lang[12] << endl;

for (; Energy < 100; Energy++) {

cout << Energy << "%" << endl;

Sleep(300);

}

cout << language.PhoneMode\_lang[13] << endl;

cout << endl;

}

#pragma once

#include <iostream>

#include <vector>

#include <stdio.h>

#include "Lang.h"

using namespace std;

class SIM

{

friend class MobilePhone;

friend class SmartPhone;

friend class HybridPhone;

friend class PushButtonPhone;

public:

Lang language;

char\* Number;

char\* Operator;

SIM(char\* Operator, char\* Number);

SIM();

void GetNumber();

void GetOperator();

static void GetToData(SIM sub);

static void GetToDataSMS(char\* number, string sms);

/\*char\* SetNumber();

char\* SetOperator();\*/

static vector<SIM> subs;

static vector<pair<string, string>> dataBase;

};

#include "SIM.h"

#include <iostream>

#include <vector>

#include <stdio.h>

SIM::SIM(char\* Number, char\* Operator) {

this->Number = Number;

this->Operator = Operator;

}

SIM::SIM() {

}

vector<SIM> SIM::subs;

vector<pair<string, string>> SIM::dataBase;

void SIM::GetToData(SIM sub) {

subs.push\_back(sub);

}

void SIM::GetToDataSMS(char\* number, string sms) {

pair<string, string> temp;

temp.first = number;

temp.second = sms;

dataBase.push\_back(temp);

}

void SIM::GetNumber() {

if (this->Number != NULL) {

cout << language.SIM\_lang[0];

puts(this->Number);

cout << endl;

}

else {

cout << language.SIM\_lang[1] << endl;

}

}

void SIM::GetOperator() {

if (this->Operator != NULL) {

cout << language.SIM\_lang[2];

puts(this->Operator);

cout << endl;

}

else {

cout << language.SIM\_lang[1] << endl;

}

}

#pragma once

#include "MobilePhone.h"

#include <iostream>

#include <array>

#include "Cover.h"

using namespace std;

class SmartPhone:public MobilePhone

{

public:

Lang language;

SmartPhone(char\* Vendor, char\* Model, string RAM, string Processor, string ROM);

void GetConfiguration() override;

void Photograph() override;

void Calculate() override;

void CheckPictures() override;

Cover\* cover = NULL;

protected:

string RAM = "";

string Processor = "";

string ROM = "";

array<string,16> pictures;

int ReadInteger();

double ReadDouble();

virtual void SetConfiguration(string RAM, string Processor, string ROM);

};

#include "SmartPhone.h"

SmartPhone::SmartPhone(char\* Vendor, char\* Model, string RAM, string Processor, string ROM):MobilePhone(Vendor,Model) {

SetConfiguration(RAM, Processor, ROM);

}

double SmartPhone::ReadDouble() {

double number;

char c;

while (1) {

if (!scanf\_s("%lf", &number) || getchar() != '\n') {

while ((c = getchar()) != '\n' && c != EOF);

}

else {

return number;

}

cout << language.SmartPhone\_lang[0];

}

}

int SmartPhone::ReadInteger() {

int number;

char c;

while (1) {

if (!scanf\_s("%d", &number) || getchar() != '\n') {

while ((c = getchar()) != '\n' && c != EOF);

}

else {

return number;

}

cout << language.SmartPhone\_lang[0];

}

}

void SmartPhone::CheckPictures()

{

if (this->state.Power == false) {

return;

}

cout << language.SmartPhone\_lang[1];

for (auto i = 0; i < this->pictures.size(); i++) {

cout << i + 1 << ")" << this->pictures[i] << endl;

}

cout << endl;

}

void SmartPhone::GetConfiguration()

{

if (this->RAM == "" || this->Processor == "" || this->ROM == "") {

cout << language.SmartPhone\_lang[2] << endl;

}

cout << language.SmartPhone\_lang[3] << endl;

cout << language.SmartPhone\_lang[4] << this->Processor << endl;

cout << "RAM:" << this->RAM << endl;

cout << "ROM:" << this->ROM << endl;

cout << endl;

}

void SmartPhone::SetConfiguration(string RAM, string Processor, string ROM)

{

this->RAM = RAM;

while (this->RAM == "") {

cout << language.SmartPhone\_lang[5];

cin >> this->RAM;

}

this->Processor = Processor;

while (this->Processor == "") {

cout << language.SmartPhone\_lang[6];

cin >> this->Processor;

}

this->ROM = ROM;

while (this->ROM == "") {

cout << language.SmartPhone\_lang[7];

cin >> this->ROM;

}

cout << endl;

}

void SmartPhone::Photograph()

{

if (this->state.Power == false) {

cout << language.SmartPhone\_lang[8] << endl;

cout << endl;

return;

}

if (this->pictures.back().empty() == false) {

cout << language.SmartPhone\_lang[9] << endl;

char choose;

int i = -1;

cin >> choose;

switch (choose) {

case 'y':

cout << language.SmartPhone\_lang[10];

while (i < 0 || i > 15) {

i = ReadInteger();

}

this->pictures[i] = "";

cout << language.SmartPhone\_lang[11] << i << language.SmartPhone\_lang[12] << endl;

break;

case 'n':

return;

break;

}

}

for (auto i = 0; i < this->pictures.size(); i++) {

if (this->pictures[i].empty() == true) {

cout << language.SmartPhone\_lang[13];

string name;

cin >> name;

this->pictures[i] = name;

break;

}

}

cout << language.SmartPhone\_lang[14] << endl;

cout << endl;

}

void SmartPhone::Calculate()

{

if (this->state.Power == false) {

cout << language.SmartPhone\_lang[8] << endl;

cout << endl;

return;

}

double x, y = 0;

cout << language.SmartPhone\_lang[15] << endl;

cout << language.SmartPhone\_lang[16];

x = ReadDouble();

cout << language.SmartPhone\_lang[17];

while (y == 0) {

y = ReadDouble();

}

cout << language.SmartPhone\_lang[18];

char operation;

cin >> operation;

switch (operation) {

case '+':

cout << "x + y = " << x + y << endl;

break;

case '-':

cout << "x - y = " << x - y << endl;

break;

case '\*':

cout << "x \* y = " << x \* y << endl;

break;

case '/':

cout << "x / y = " << x / y << endl;

break;

default:

cout << language.SmartPhone\_lang[19] << endl;

break;

}

cout << endl;

}

#pragma once

#include "MobilePhone.h"

class PushButtonPhone :public MobilePhone

{

public:

PushButtonPhone(char\* Vendor, char\* Model,double Energy);

};

#include "PushButtonPhone.h"

PushButtonPhone::PushButtonPhone(char\* Vendor, char\* Model,double Energy) :MobilePhone(Vendor, Model) {

this->state.Energy = Energy;

}

#pragma once

#include <string>

#include "Lang.h"

using namespace std;

class Cover

{

friend class SmartPhone;

public:

Lang lang;

Cover(string color, bool armored);

void GetInfo();

int ReadInteger();

protected:

string color;

bool armored;

string SetColor(string color);

bool SetArmored(bool armored);

};

#include "Cover.h"

#include <iostream>

using namespace std;

int Cover::ReadInteger() {

int number;

char c;

while (1) {

if (!scanf\_s("%d", &number) || getchar() != '\n') {

while ((c = getchar()) != '\n' && c != EOF);

}

else {

return number;

}

cout << lang.Cover\_lang[0];

}

}

Cover::Cover(string color,bool armored) {

this->color = SetColor(color);

this->armored = SetArmored(armored);

}

void Cover::GetInfo()

{

cout << lang.Cover\_lang[1] << endl;

cout << lang.Cover\_lang[2] << this->color << endl;

cout << lang.Cover\_lang[3] << this->armored << endl;

cout << endl;

}

string Cover::SetColor(string color)

{

if (color.empty() == 1) {

while (color.empty() == 1) {

cout << lang.Cover\_lang[4] << endl;

cin >> color;

}

}

else {

return color;

}

}

bool Cover::SetArmored(bool armored)

{

if (armored > 1 || armored < 0) {

while (armored > 1 || armored < 0) {

cout << lang.Cover\_lang[5] << endl;

armored = ReadInteger();

}

}

else {

return armored;

}

}

#pragma once

#include <string>

#include "Lang.h"

using namespace std;

class Person

{

public:

Person();

static Lang language;

virtual void GetInform();

protected:

string name;

int age = 0;

virtual void SetInfom();

int ReadInteger();

};

#include "Person.h"

#include <iostream>

using namespace std;

int Person::ReadInteger() {

int number;

char c;

while (1) {

if (!scanf\_s("%d", &number) || getchar() != '\n') {

while ((c = getchar()) != '\n' && c != EOF);

}

else {

return number;

}

cout << Person::language.Person\_lang[0];

}

}

Person::Person()

{

SetInfom();

}

void Person::GetInform()

{

cout << Person::language.Person\_lang[1] << endl;

cout << Person::language.Person\_lang[2] << this->name << endl;

cout << Person::language.Person\_lang[3] << this->age << endl;

}

void Person::SetInfom()

{

while (name.empty() == 1) {

cout << Person::language.Person\_lang[4];

cin >> this->name;

}

while (this->age <= 0 || this->age > 99) {

cout << Person::language.Person\_lang[5];

this->age = ReadInteger();

}

}

#pragma once

#include "Person.h"

#include <iostream>

#include "MobilePhone.h"

using namespace std;

class Civil:public Person

{

friend class Shop;

public:

MobilePhone\* phone = NULL;

Civil();

void AddMoney(double add);

void SetInfom() override {

Person::SetInfom();

while (balance < 0) {

cout << Person::language.Person\_lang[6];

balance = ReadDouble();

}

}

void GetInform() override {

Person::GetInform();

cout << Person::language.Person\_lang[6] << balance << endl;

}

protected:

double ReadDouble();

double balance = -1;

};

#include "Civil.h"

Civil::Civil()

{

SetInfom();

}

void Civil::AddMoney(double add)

{

}

double Civil::ReadDouble()

{

double number;

char c;

while (1) {

if (!scanf\_s("%lf", &number) || getchar() != '\n') {

while ((c = getchar()) != '\n' && c != EOF);

}

else {

return number;

}

printf\_s("Incorrect number.\nTry again: ");

}

}

#pragma once

#include<vector>

#include<utility>

#include "MobilePhone.h"

#include "Civil.h"

#include "Lang.h"

using namespace std;

class Shop

{

friend class MobilePhone;

friend class SmartPhone;

friend class HybridPhone;

friend class PushButtonPhone;

public:

Shop();

static Lang language;

void BuyProduct(Civil\* civ,int index);

void AddProductInList(MobilePhone\* phone);

void AddCoverInList(Cover\* cover);

void AddSIMInList(SIM\* card);

void DeleteProductFromList(int index);

void DeleteCoverFromList(int index);

void DeleteSIMFromList(int index);

void CheckProductList();

int size = ProductList.size();

private:

static vector<pair<MobilePhone\*,double>> ProductList;

static vector<pair<SIM\*, double>> SIMList;

static vector<pair<Cover\*, double>> CoverList;

};

#include "Shop.h"

using namespace std;

double ReadDouble() {

double number;

char c;

while (1) {

if (!scanf\_s("%lf", &number) || getchar() != '\n') {

while ((c = getchar()) != '\n' && c != EOF);

}

else {

return number;

}

cout << Shop::language.Shop\_lang[0];

}

}

Shop::Shop()

{

}

void Shop::BuyProduct(Civil\* civ,int index)

{

cout << Shop::language.Shop\_lang[1] << endl;

cout << Shop::language.Shop\_lang[2] << endl;

cout << Shop::language.Shop\_lang[3] << endl;

char choose;

cin >> choose;

switch (choose) {

case '1':

if (index < 0 || index > ProductList.size())

break;

if (civ->balance >= ProductList[index - 1].second) {

civ->balance -= ProductList[index - 1].second;

civ->phone = ProductList[index - 1].first;

cout << Shop::language.Shop\_lang[4] << endl;

ProductList.erase(ProductList.begin() + index - 1);

}

else {

cout << Shop::language.Shop\_lang[5] << endl;

}

break;

case '2':

if (index < 0 || index > CoverList.size())

break;

if (civ->balance >= CoverList[index - 1].second) {

civ->balance -= CoverList[index - 1].second;

civ->phone->cover = CoverList[index - 1].first;

cout << Shop::language.Shop\_lang[6] << endl;

CoverList.erase(CoverList.begin() + index - 1);

}

else {

cout << Shop::language.Shop\_lang[5] << endl;

}

break;

case '3':

if (index < 0 || index > SIMList.size())

break;

if (civ->balance >= SIMList[index - 1].second) {

civ->balance -= SIMList[index - 1].second;

cout << Shop::language.Shop\_lang[7] << endl;

civ->phone->SIM\_inventory = SIMList[index - 1].first;

SIMList.erase(SIMList.begin() + index - 1);

}

else {

cout << Shop::language.Shop\_lang[5] << endl;

}

break;

default:

cout << Shop::language.Shop\_lang[8] << endl;

break;

}

}

void Shop::CheckProductList() {

cout << Shop::language.Shop\_lang[9] << endl;

cout << Shop::language.Shop\_lang[10] << endl;

for (auto i = 0; i < ProductList.size(); i++) {

cout << (i + 1) << ")" << Shop::language.Shop\_lang[11] << ProductList[i].second << endl;

ProductList[i].first->GetVendor();

ProductList[i].first->GetModel();

}

cout << Shop::language.Shop\_lang[12] << endl;

for (auto i = 0; i < CoverList.size(); i++) {

cout << (i + 1) << ")" << Shop::language.Shop\_lang[11] << CoverList[i].second << endl;

CoverList[i].first->GetInfo();

}

cout << Shop::language.Shop\_lang[13] << endl;

for (auto i = 0; i < SIMList.size(); i++) {

cout << (i + 1) << ")" << Shop::language.Shop\_lang[11] << SIMList[i].second << endl;

SIMList[i].first->GetNumber();

SIMList[i].first->GetOperator();

}

}

void Shop::AddProductInList(MobilePhone\* phone)

{

double price = 0;

cout << Shop::language.Shop\_lang[14];

while (price <= 0) {

price = ReadDouble();

}

pair<MobilePhone\*,double> temp(phone,price);

ProductList.push\_back(temp);

cout << Shop::language.Shop\_lang[15] << endl;

}

void Shop::AddCoverInList(Cover\* cover)

{

double price = 0;

cout << Shop::language.Shop\_lang[16];

while (price <= 0) {

price = ReadDouble();

}

pair<Cover\*, double> temp(cover, price);

CoverList.push\_back(temp);

cout << Shop::language.Shop\_lang[15] << endl;

}

void Shop::AddSIMInList(SIM\* card)

{

double price = 0;

cout << Shop::language.Shop\_lang[17];

while (price <= 0) {

price = ReadDouble();

}

pair<SIM\*, double> temp(card, price);

SIMList.push\_back(temp);

cout << Shop::language.Shop\_lang[15] << endl;

}

void Shop::DeleteProductFromList(int index)

{

ProductList.erase(ProductList.begin() + index-1);

}

void Shop::DeleteCoverFromList(int index)

{

CoverList.erase(CoverList.begin() + index - 1);

}

void Shop::DeleteSIMFromList(int index)

{

SIMList.erase(SIMList.begin() + index - 1);

}

vector<pair<MobilePhone\*, double>> Shop::ProductList;

vector<pair<Cover\*, double>> Shop::CoverList;

vector<pair<SIM\*, double>> Shop::SIMList;

#pragma once

#include "PushButtonPhone.h"

#include "SmartPhone.h"

class HybridPhone:public PushButtonPhone,public SmartPhone

{

public:

HybridPhone(char\* Vendor, char\* Model, string RAM, string Processor, string ROM, double Energy = 100);

};

#include "HybridPhone.h"

#include <iostream>

using namespace std;

HybridPhone::HybridPhone(char\* Vendor, char\* Model, string RAM, string Processor, string ROM, double Energy) :SmartPhone(Vendor, Model, RAM, Processor, ROM), PushButtonPhone(Vendor,Model,Energy)

{

}

#pragma once

#include <vector>

#include <string>

#include <fstream>

using namespace std;

class Lang

{

public:

const string pathRu = "lan\_ru.txt";

const string pathEn = "lan\_en.txt";

const string pathRu\_menu = "menu\_ru.txt";

const string pathEn\_menu = "menu\_en.txt";

const string pathRu\_not = "notification\_creation\_ru.txt";

const string pathEn\_not = "notification\_creation\_en.txt";

const string pathRu\_init = "init\_ru.txt";

const string pathEn\_init = "init\_en.txt";

const string pathRu\_control\_menu = "control\_menu\_ru.txt";

const string pathEn\_control\_menu = "control\_menu\_en.txt";

const string pathRu\_control\_init = "control\_init\_ru.txt";

const string pathEn\_control\_init = "control\_init\_en.txt";

const string pathRu\_MobilePhone = "MobilePhone\_ru.txt";

const string pathEn\_MobilePhone = "MobilePhone\_en.txt";

const string pathRu\_PhoneMode = "PhoneMode\_ru.txt";

const string pathEn\_PhoneMode = "PhoneMode\_en.txt";

const string pathRu\_SIM = "SIM\_ru.txt";

const string pathEn\_SIM = "SIM\_en.txt";

const string pathRu\_SmartPhone = "SmartPhone\_ru.txt";

const string pathEn\_SmartPhone = "SmartPhone\_en.txt";

const string pathRu\_Cover = "Cover\_ru.txt";

const string pathEn\_Cover = "Cover\_en.txt";

const string pathRu\_Shop = "Shop\_ru.txt";

const string pathEn\_Shop = "Shop\_en.txt";

const string pathRu\_Person = "Person\_ru.txt";

const string pathEn\_Person = "Person\_en.txt";

const string pathDescription = "descrp.txt";

bool lang = 1;//1 - rus/0 - eng

static vector<string> language;

static vector<string> init\_words;

static vector<string> control\_init\_words;

static vector<string> control;

static vector<string> MobilePhone\_lang;

static vector<string> PhoneMode\_lang;

static vector<string> SIM\_lang;

static vector<string> SmartPhone\_lang;

static vector<string> Cover\_lang;

static vector<string> Shop\_lang;

static vector<string> Person\_lang;

void LoadLanguageMobilePhone(Lang lang);

void LoadLanguagePhoneMode(Lang lang);

void LoadLanguageSIM(Lang lang);

void LoadLanguageSmartPhone(Lang lang);

void LoadLanguageCover(Lang lang);

void LoadLanguagePerson(Lang lang);

void LoadLanguageShop(Lang lang);

};

#include "Lang.h"

#include <iostream>

void Lang::LoadLanguageMobilePhone(Lang lang)

{

string temp\_ru;

string temp\_en;

ifstream fin;

if (lang.lang == 1) {

fin.open(lang.pathRu\_MobilePhone);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathRu\_MobilePhone;

return;

}

while (!fin.eof()) {

temp\_ru = "";

getline(fin, temp\_ru);

lang.MobilePhone\_lang.push\_back(temp\_ru);

}

}

else if (lang.lang == 0) {

fin.open(lang.pathEn\_MobilePhone);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathEn\_MobilePhone;

return;

}

while (!fin.eof()) {

temp\_en = "";

getline(fin, temp\_en);

lang.MobilePhone\_lang.push\_back(temp\_en);

}

}

}

void Lang::LoadLanguagePhoneMode(Lang lang)

{

string temp\_ru;

string temp\_en;

ifstream fin;

if (lang.lang == 1) {

fin.open(lang.pathRu\_PhoneMode);

if(!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathRu\_PhoneMode;

return;

}

while (!fin.eof()) {

temp\_ru = "";

getline(fin, temp\_ru);

lang.PhoneMode\_lang.push\_back(temp\_ru);

}

}

else if (lang.lang == 0) {

fin.open(lang.pathEn\_PhoneMode);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathEn\_PhoneMode;

return;

}

while (!fin.eof()) {

temp\_en = "";

getline(fin, temp\_en);

lang.PhoneMode\_lang.push\_back(temp\_en);

}

}

}

void Lang::LoadLanguageSIM(Lang lang)

{

string temp\_ru;

string temp\_en;

ifstream fin;

if (lang.lang == 1) {

fin.open(lang.pathRu\_SIM);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathRu\_SIM;

return;

}

while (!fin.eof()) {

temp\_ru = "";

getline(fin, temp\_ru);

lang.SIM\_lang.push\_back(temp\_ru);

}

}

else if (lang.lang == 0) {

fin.open(lang.pathEn\_SIM);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathEn\_SIM;

return;

}

while (!fin.eof()) {

temp\_en = "";

getline(fin, temp\_en);

lang.SIM\_lang.push\_back(temp\_en);

}

}

}

void Lang::LoadLanguageSmartPhone(Lang lang)

{

string temp\_ru;

string temp\_en;

ifstream fin;

if (lang.lang == 1) {

fin.open(lang.pathRu\_SmartPhone);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathRu\_SmartPhone;

return;

}

while (!fin.eof()) {

temp\_ru = "";

getline(fin, temp\_ru);

lang.SmartPhone\_lang.push\_back(temp\_ru);

}

}

else if (lang.lang == 0) {

fin.open(lang.pathEn\_SmartPhone);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathEn\_SmartPhone;

return;

}

while (!fin.eof()) {

temp\_en = "";

getline(fin, temp\_en);

lang.SmartPhone\_lang.push\_back(temp\_en);

}

}

}

void Lang::LoadLanguageCover(Lang lang)

{

string temp\_ru;

string temp\_en;

ifstream fin;

if (lang.lang == 1) {

fin.open(lang.pathRu\_Cover);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathRu\_Cover;

return;

}

while (!fin.eof()) {

temp\_ru = "";

getline(fin, temp\_ru);

lang.Cover\_lang.push\_back(temp\_ru);

}

}

else if (lang.lang == 0) {

fin.open(lang.pathEn\_Cover);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathEn\_Cover;

return;

}

while (!fin.eof()) {

temp\_en = "";

getline(fin, temp\_en);

lang.Cover\_lang.push\_back(temp\_en);

}

}

}

void Lang::LoadLanguagePerson(Lang lang)

{

string temp\_ru;

string temp\_en;

ifstream fin;

if (lang.lang == 1) {

fin.open(lang.pathRu\_Person);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathRu\_Person;

return;

}

while (!fin.eof()) {

temp\_ru = "";

getline(fin, temp\_ru);

lang.Person\_lang.push\_back(temp\_ru);

}

}

else if (lang.lang == 0) {

fin.open(lang.pathEn\_Person);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathEn\_Shop;

return;

}

while (!fin.eof()) {

temp\_en = "";

getline(fin, temp\_en);

lang.Person\_lang.push\_back(temp\_en);

}

}

}

void Lang::LoadLanguageShop(Lang lang)

{

string temp\_ru;

string temp\_en;

ifstream fin;

if (lang.lang == 1) {

fin.open(lang.pathRu\_Shop);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathRu\_Shop;

return;

}

while (!fin.eof()) {

temp\_ru = "";

getline(fin, temp\_ru);

lang.Shop\_lang.push\_back(temp\_ru);

}

}

else if (lang.lang == 0) {

fin.open(lang.pathEn\_Shop);

if (!fin.is\_open()) {

cout << "\nCant find this file : " << lang.pathEn\_Shop;

return;

}

while (!fin.eof()) {

temp\_en = "";

getline(fin, temp\_en);

lang.Shop\_lang.push\_back(temp\_en);

}

}

}

vector<string> Lang::language;

vector<string> Lang::init\_words;

vector<string> Lang::control\_init\_words;

vector<string> Lang::control;

vector<string> Lang::MobilePhone\_lang;

vector<string> Lang::PhoneMode\_lang;

vector<string> Lang::SIM\_lang;

vector<string> Lang::SmartPhone\_lang;

vector<string> Lang::Cover\_lang;

vector<string> Lang::Shop\_lang;

vector<string> Lang::Person\_lang;

Приложение Ж Файлы исходных данных

Ж.1 Файл conf.txt

Все файлы локализации должны находиться в папке …KursovoyProekt\KursovoyProekt

;

"lan\_ru.txt";

"lan\_en.txt";

"menu\_ru.txt";

"menu\_en.txt";

"notification\_creation\_ru.txt";

"notification\_creation\_en.txt";

"init\_ru.txt";

"init\_en.txt";

"control\_menu\_ru.txt";

"control\_menu\_en.txt";

"control\_init\_ru.txt";

"control\_init\_en.txt";

“MobilePhone\_ru.txt";

"MobilePhone\_en.txt";

"PhoneMode\_ru.txt";

"PhoneMode\_en.txt";

"SIM\_ru.txt";

"SIM\_en.txt";

"SmartPhone\_ru.txt";

"SmartPhone\_en.txt";

"Cover\_ru.txt";

"Cover\_en.txt";

"Shop\_ru.txt";

"Shop\_en.txt";

"Person\_ru.txt";

"Person\_en.txt";

;

Ж.2 Файл control\_init\_ru.txt

Введите номер

Введите сообщение

Введите количество денег

Неверный ввод

У вас нет телефона

control\_menu\_ru.txt

-Список действий-

0)Выйти

1)Вставить SIM-карту

2)Вытащить SIM-карту

3)Позвонить

4)Отправить смс

5)Принять смс

6)Посмотреть конфигурацию

7)Сфотографировать

8)Посмотреть список фотографий

9)Калькулятор

10)Включить/Выключить телефон

11)Узнать уровень заряда

12)Зарядить телефон

13)Режим полёта

14)Посмотреть состояние телефона

15)Купить продукт(телефон,SIM-карту,чехол)

16)Добавить денег на счёт

Cover\_ru.txt

Неверное число,попробуйте снова:

-Информация о чехле-

Цвет:

Армированный:

Введите цвет чехла:

Армирован ли чехол?

init\_ru.txt

Введите производителя

Введите модель

Введите RAM

Введите название процессора

Введите ROM

Введите цвет чехла

Армирован ли чехол(1-да|0-нет)

Введите номер

Введите оператора

lan\_ru.txt

ДонНТУ

Кафедра ПИ

ПИ-21в

Рустамов Владислав Русланович

menu\_ru.txt

-Выберите действие-

1)Создать гражданина

2)Выбрать гражданина

(3-7)Автоматическое добавление в магазин

3)Создать Смартфон

4)Создать Кнопочный телефон

5)Создать Гибридный телефон

6)Создать Чехол

7)Создать Сим-карту

8)Удалить Гражданина

9)Удалить телефон из магазина

0)Проверить список товаров магазина

10)Проверить список гражданских

11)Выйти

MobilePhone\_ru.txt

Введите название производителя:

Введите название модели:

Производитель:

Модель:

Информация о телефоне:

SIM-карта отсутствует!

Связываем вас с абонентом,пожалуйста подождите...

Вы связались в абонентом с номером

Разъединение...

Телефон выключен или включен режим полета!

Отправляем сообщение абоненту,пожалуйста подождите...

Сообщение отправлено

Сообщение:

SIM-карта уже установлена!

SIM-карта установлена в телефон

У вас нет SIM-карты

SIM-карта изъята

notification\_creation\_ru.txt

Гражданский добавлен

Введите индекс

Смартфон создан

Кнопочный телефон создан

Гибридный телефон создан

Чехол создан

Сим-карта создана

Гражданин удален

Телефон удален из магазина

-Список гражданских-

Данная функция отсутствует

Список гражданских пуст

Person\_ru.txt

Неверное число, попробуйте снова:

-Информация о человеке-

Имя:

Возраст:

Введите имя:

Введите возраст:

Введите баланс:

Текущий баланс:

PhoneMode\_ru.txt

Режим полета включен!

Режим полета выключен!

Телефон выключен, невозможно включить режим полета!

Состояние телефона

Уровень заряда:

Режим полета:

Вкл/выкл:

Телефон включен.

Нужно зарядить телефон!

Телефон выключен.

Телефон выключен, заряд на нуле!

Уровень заряда телефона:

Заряжаем телефон подождите!

Телефон заряжен!

Shop\_ru.txt

Неверное число, попробуйте снова:

1)Телефоны

2)Чехлы

3)SIM-карты

Телефон приобретен

Недостаточно средств

Чехол приобретен

SIM-карта приобретена

Неверный ввод

-Список Товаров-

[Телефоны]

Цена:

[Чехлы]

[SIM-карты]

Введите цену телефона:

Товар добавлен в список

Введите цену чехла:

Введите цену сим-карты:

SIM\_ru.txt

Номер:

Возможно в вашем телефоне не установлена SIM

Оператор:

SmartPhone\_ru.txt

Неверное число, попробуйте снова:

-Галерея-

Информация о конфигурации телефона отсутствует

Конфигурация телефона:

Процессор:

Введите RAM:

Введите процессор:

Введите ROM:

Смартфон выключен!

Память смартфона переполнена!Хотите удалить один из объектов?y/n

Введите индекс(0-15):

Элемент

удален!

Введите название картинки:

Сделан снимок!

-Калькулятор-

Введите x:

Введите y:

Введите операцию(+,-,\*,/):

Неверный ввод!

Приложение Л Экранные формы

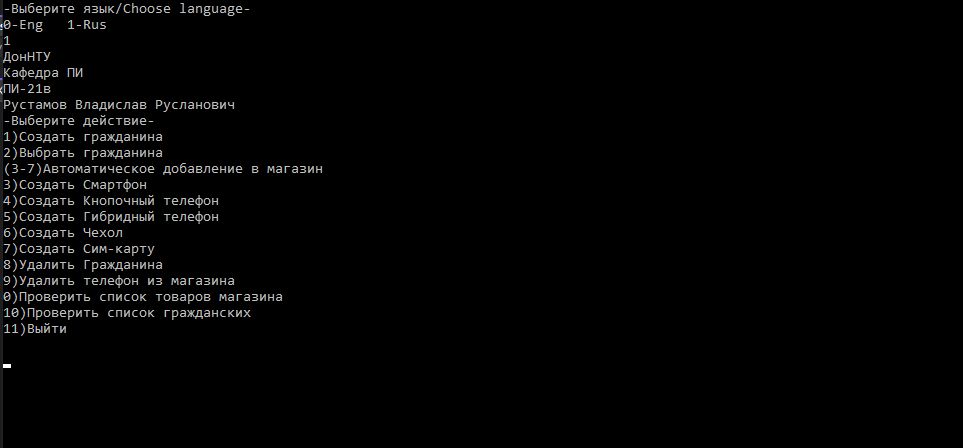


Рисунок Л.1 – Заставка, отображаемая при загрузке системы

И главное меню

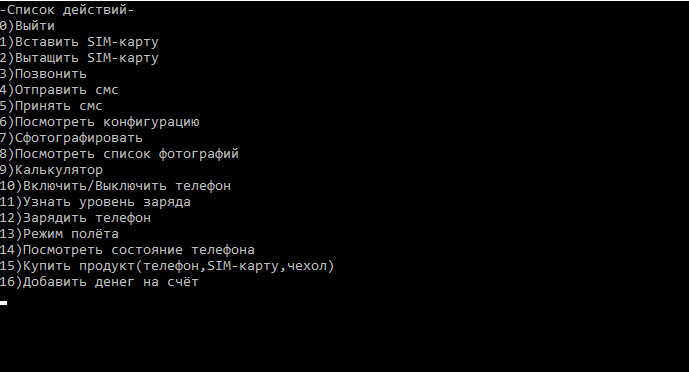


Рисунок Л.2 – Внутреннее меню

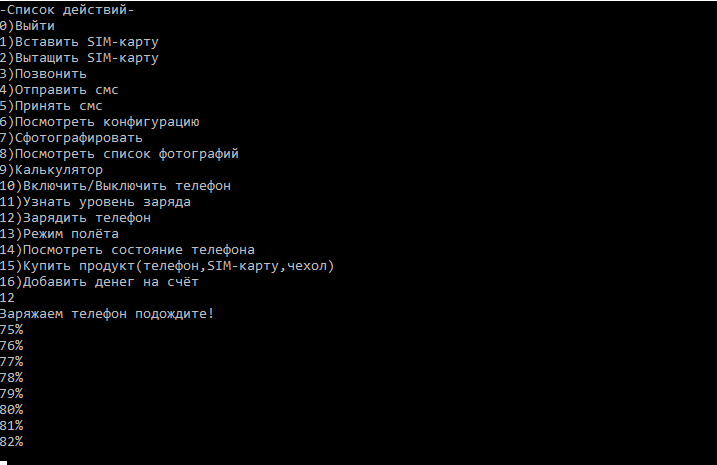


Рисунок Л.3 – Демонстрации работы системы

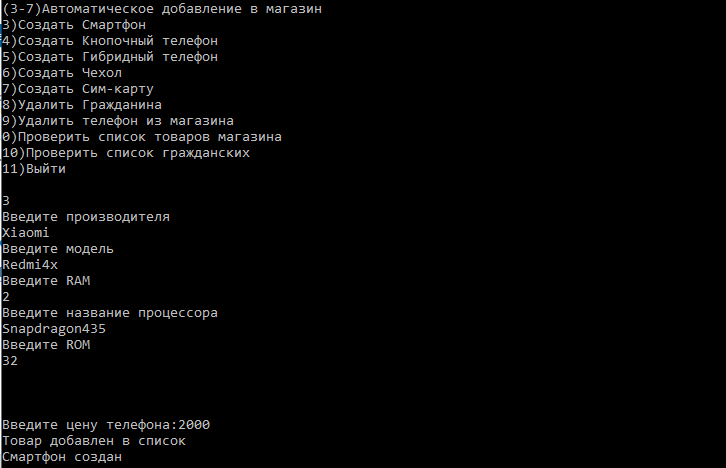


Рисунок Л.4 – Меню “настройки смартфона “

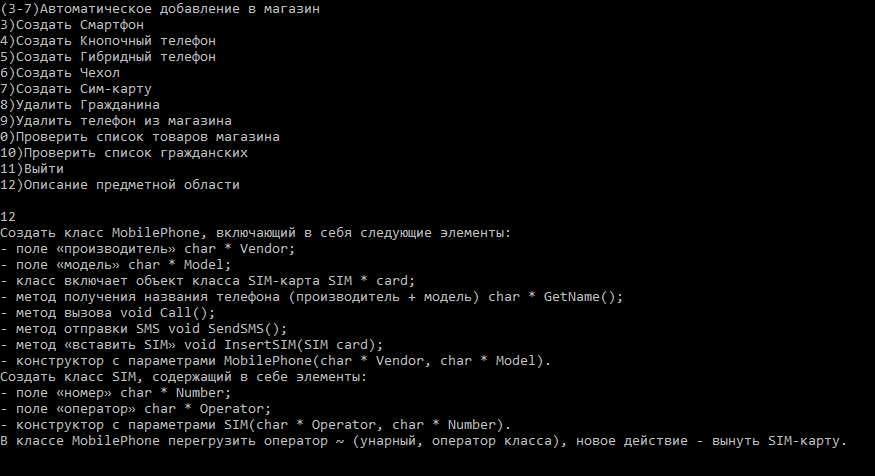


Рисунок Л.5 – Меню “Описание предметной области”