**Введение**

Кодовые замки являются эффективным средством предотвращения доступа посторонних лиц к охраняемым помещениям. К их достоинствам можно отнести простоту в обращении, надёжность, возможность обеспечить высокую степень защиты, относительную лёгкость смены кода (по сравнению со сменой обычного механического замка). Также немаловажными являются отсутствие необходимости изготовления ключей при предоставлении доступа большому количеству людей и невозможность физической потери ключа. Недостатком таких систем можно назвать возможность для злоумышленника подсмотреть код или подобрать его. Однако, при большой разрядности кода или наличии конструктивных особенностей, препятствующих подбору кода, таких как ограничение количества попыток или введение временной задержки между неудачными попытками, эта задача сильно затрудняется, поэтому последний недостаток нельзя назвать существенным. В данном курсовом проекте осуществляется разработка электронного кодового замка.

* 1. **Описание предметной области**

Практическая ценность. Создание программы кодовый- замок решает следующие задачи:

* Помогает обеспечить защиту имущества.
* Позволяет связаться с жителями квартиры дистанционно.
* Позволяет изменять пароли, с помощью управляющих комбинаций.

В данной курсовой работе представлен пример объектно-ориентированной разработки программного обеспечения на основе проектирования Visual C# возможности для построения UML-диаграмм, анализа, моделирования и преобразования моделей и генерации кода.

Исходными данными для выполнения курсовой работы является постановка задачи, приведенная в пункте 2, результатом являются диаграммы вариантов использования, классов, состояний, деятельности, последовательностей, коопераций, компонентов и размещения, а так же сгенерированный код.

Функциональность объекта изначально ограничена, но полученные материалы могут быть использованы для программирования реально существующего объекта, то есть, данный способ разработки имеет практическую ценность.

Для достижения поставленной задачи требуется решить ряд конкретизирующих её задач:

* Выявить основные тенденции развития и особенности объектно-ориентированного программирования и проектирования;
* Выделить основные функции системы, объекты, взаимодействующие с ними, и реакции системы на действия этих объектов;
* Проанализировать основные компоненты системы и составить их подробное описание
* На основе полученных данных осуществить проектирование системы при помощи диаграмм на языке UML.
* Составить инструкцию пользователя для того, чтобы обеспечить легкость внедрения её в структуру предприятия и упростить обучение сотрудников работе в данной программе.

В качестве объекта исследования выбраны часы-будильник – в данной работе – это готовая автоматизированная информационная система предназначенная для помощи распределения времени.

**1.2 Неформальная постановка задачи**

Основные принципы и концепции объектной модели в программировании развивались в процессе эволюции множества разных объектных и объектно-ориентированных языков.

Основной идеей объектного подхода является объединение данных и производимых над этими данными операций в одно концептуально замкнутое понятие — класс. Данные класса не должны изменяться извне его, доступ к данным стоит осуществлять только через функции-члены (методы класса).

Класс - группировка объектов, которые имеют те же самые свойства, общее поведение и одинаковые отношения.

Класс определяет характеристики объектов. Однако значения могут быть назначены только после того, как объект будет создан. Только в этом случае появляется фактический образец объекта.

Объект - понятие или вещь с определенными границами, которая является уместной в данной проблеме, с которой мы имеем дело. Объекты позволяют выполнить две цели:

1.Они помогают понимать окружающий мир.

2. Они обеспечивают практическую реализацию для создаваемого нами приложения.

Программа, написанная на объектном языке, представляет собой совокупность объектов, каждый из которых принадлежит к определенному абстрактному типу данных (классу) и имеет интерфейс в виде набора методов для взаимодействия друг с другом (посылки сообщений).

Все объектно-ориентированные языки разбивают программу на части, многократно используемые и расширяемые. Программы представляют собой объекты многократного использования. Эти объекты могут группироваться различными способами, формируя новые программы.

Объектно-ориентированное проектирование — это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы.

Объектно-ориентированное проектирование дает возможность создавать расширяемые системы (extensible systems). Расширяемость (extensibility) означает, что существующую систему можно заставить работать с новыми компонентами без внесения в нее каких-либо изменений. . Это уникальное и очень мощное понятие, потому что реальная задача постоянно требует изменений. Это уникальное и очень мощное понятие, потому что реальная задача постоянно требует изменений.

Таким образом, чтобы составить корректную оценку объектно-ориентированной модели программного обеспечения, следует обратиться к описанию объекта исследования, или, иными словами, к постановке задачи, в которой описаны необходимые для реализации функции и устройство банкомата, для которого проектируется информационная система.

Электромеханический замок, позволяет контролировать доступ в помещение, посредством контрольного кода, и пароля.

В замке предусмотрена возможность голосовой связи с человеком, по ту сторону двери.

С внешней стороны имеется возможность изменить пароль (При открытой двери, нажав кнопку «КОНТРОЛЬ») и контрольный код (При открытой двери, нажав кнопку «ВЫЗОВ»).

## 1.3 Обзор существующих методов решения

На сегодняшний день существует множество всевозможных кодовых замков.

Вот некоторые из них :



**Рисунок 1 – Кодовый замок на дверь**

****

**Рисунок 2 – Кодовый замок на сейф**



**Рисунок 3- Кодовый замок на дверь**

## 1.4 План робот

Срок разработки ~ 4 недели:

16.11.2014 Постановка задачи и начало разработки.

20.11.2014 Первая версия программы.

22.11.2014 Тестирование программы.

22.11.2014 Устранение недостатков, новая версия программы.

23.11.2014 Тестирование программы.

28.11.2014 Финальная версия программы.

## Требования к окружению

**2.1 Требования к аппаратному обеспечению**

Минимальные требования к аппаратному обеспечению:

|  |
| --- |
| 1.Процессор з тактовой частотой не меньше 2 ГГц. |
| 2.Оперативна память объемом 512 Мб. |
| 3.Жесткий диск с доступными 1 Гб памяти |
| 4.Монитор с разрешением 1024х768. |

## 2.2 Требования к програмному обеспечению

Windows система Windows 8.

Платформа .NET Framework 4.0

## 2.3 Требования к пользователю

Пользователь должен пройти базовый курс использования системы Windows, в другом случае, программа не требует особенных навыков работы с Windows.

## 2.4 Организационные требования

Для использования программы, необходим человек, с базовыми познаниями Windows, и опытом использования ПК.

Архитектура системы

Анализируя поставленную задачу было решено определить такие классы:

* Lock; - Класс, отвечающий за замок
* MainWindow; - Класс отображает окна программы
* ProcessRun; - Класс работает со временем и состояниями
* NumberBoard; - Класс работает с внешней панелью
* Inner; - Класс отвечающий внутренней панели

Полученная Диаграмма классов:

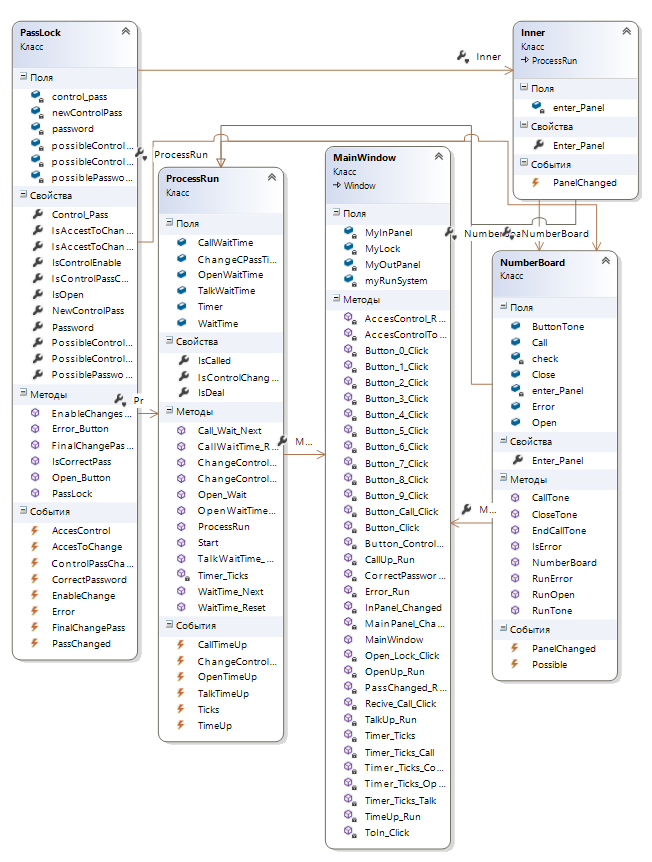


Рисунок 4 - Диаграмма вариантов использования

Варианты использования

Модель вариантов использования представляет собой модель взаимодействия пользователей для решения проблем и задач.

Модель вариантов использования описывает цели пользователей, взаимодействие между пользователями и системой и требуемое поведение системы для удовлетворения этих целей.

Модель вариантов использования состоит из ряда модельных элементов (или графических примитивов). К наиболее важным из них относятся варианты использования [use cases], действующие лица или акторы [actors] и отношения (связи) между ними [relationships].

Для того, чтобы описать то, как может использоваться данная проектируемая система, необходимо выявить действующие лица – то есть, все объекты, которые будут взаимодействовать с системой. В данном случае, выделяется один объект и рассматривается его поведение в системе.

Действующее лицо:

Пользователь – Любой человек, пользующееся программным обеспечением будильника.

Варианты использования:

Исходя из потребностей пользователя, выделяются следующие варианты использования:

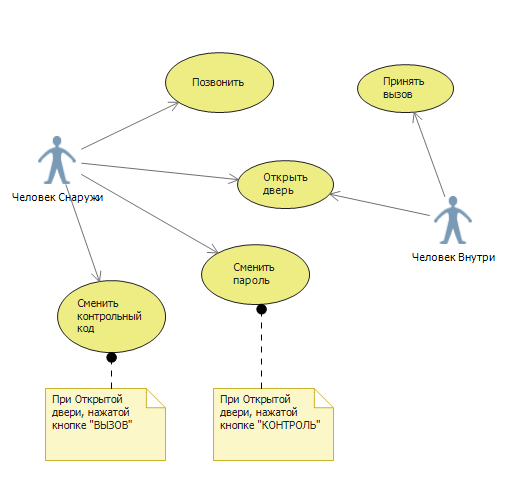


Рисунок 5 - Диаграмма вариантов использования

# Проект программной системы

## 5.1 Средства реализации

Модель программного обеспечения была реализована посредством языка С#

Среда разработки: Microsoft Visual Studio 2013.

## 5.2 Проект Интерфейса

Интерфейс программы был выполнен в умеренных тонах. Область видимости разделена на 2 состояния- «Внутри», и «Снаружи».

***Главное Окно:***



***Внутренняя Панель:***



***Дверь Открыта:***



***Смена пароля:***

