СОДЕРЖАНИЕ

[СТРУКТУРА И ФУНКЦИИ ЧАСТЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ,](#_bookmark0) [КОТОРЫЕ МОГУТ БЫТЬ В СИСТЕМЕ 2](#_bookmark0)

Разработка специального программного обеспечения системы 3

ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ СИСТЕМЫ И ОПИСАНИЕ ЕЁ КЛАССОВ И МЕТОДОВ 4

Выбор средства управления данными 4

Разработка моделей данных 10

[ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СИСТЕМЫ –ИНТЕРФЕЙС И](#_bookmark1)

[ВХОДНАЯ-ВЫХОДНАЯ ИНОФРМАЦИЯ 11](#_bookmark1)

Графический интерфейс 11

Работа с информацией, файлами и их выходными данными 13

КОНТРОЛЬНЫЕ ВОПРОСЫ 14

СТРУКТУРА И ФУНКЦИИ ЧАСТЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, КОТОРЫЕ МОГУТ БЫТЬ В СИСТЕМЕ

Компьютеризированная система представляет собой комплекс технического, функционального, информационного и программного обеспечений. Программное же обеспечение включает:

* Специальное программного обеспечения системы.
* Общесистемное программное обеспечение системы.
* Сервисные программы и утилиты.
* Инструментальные средства.

Общая схема избранное программного обеспечения системы приведена на рис. 1.

Специальное программного обеспечения системы - это собственно та часть программного обеспечения системы всей системы, которую приходится писать и создавать каждому человеку. Ведь, при всём многообразии программных продуктов на мировом рынке, у каждого предприятия есть задачи и функции, которые специфичны только для него и не учтены в уже существующих продуктах. Бывают и другие ситуации. Например, когда лучше разработать новее и современнее программный продукт, чем использовать устаревший. Также есть такое понятие как «экономическая целесообразность». Ведь использовать программу, стоимость которой превышает несколько десятков или сотен тысяч рублей, может себе позволить далеко не каждое предприятие. Ведь поддержка её работоспособности тоже будет стоить не малых денег. Потому, в некоторых случаях, и необходима разработка нового программного продукта, который обладает индивидуальным набором функций.

Общесистемное программного обеспечения системы - те программные средства, которые предоставляют пользовательский интерфейс для работы с аппаратным обеспечением компьютеров.

Сервисные же программы предназначены для расширения функциональности общесистемного программного обеспечения.

А вот с помощью инструментальных средств разработчик и создает

специальное программное обеспечение.

Компоненты

ПО системы

Инструмен- тальные средства

Прилегающая

ПО системы

Сервисные программы и утилиты

Специальное ПО

системы

MS ACCESS

Windows

Антивирус Касперского

Модуль

«изменение данных»

Модуль «перейти к запросам»

Модуль «отчёты»

Internet Explorer

Рисунок 1 – Структурна схема ПО системы Разработка специального программного обеспечения системы

Для внедрения и эксплуатации системы разработано специальное программное обеспечение. Специальное ПО системы разделяется на ряд комплексов по модулям:

* Модуль «Изменение данных»;
* Модуль «Перейти к запросам»;
* Модуль «Отчёты».
* Модуль «Авторизация»

ОРГАНИЗАЦИЯ ОБРАБОТКИ ИНФОРМАЦИИ

Входными данными является информация получена о учениках.

Исходными данными являются выборка информации из базы данных о учениках.

Все данные хранятся на главном компьютере, который, как правило, защищен гораздо лучше большинства других. В данной системе работать одна группа пользователей: администрация. Поэтому для обеспечения конфиденциальности всей информации, которая будет хранится в БД, а также для разграничения прав доступа не требуется разрабатывать модуль авторизации.

На рис. 2 размещено скриншот с MS Access.

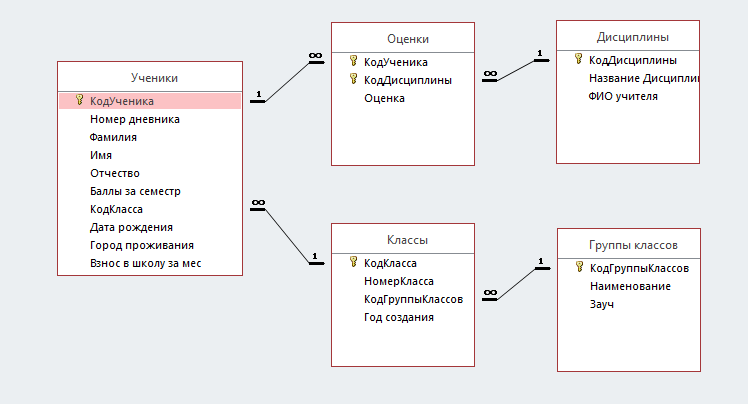


Рисунок 2 - Схема данных

ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ СИСТЕМЫ И ОПИСАНИЕ ЕЁ КЛАССОВ И МЕТОДОВ

Выбор средства управления данными

Иерархическая модель данных представляет собой иерархию в виде дерева. Данная модель данных базируется на сегменте, который представляет собой совокупность полей, характеризующих данный сегмент. Сегменты различаются по типу, а каждый тип характеризуется фиксированной длиной и конкретным разбивкой на поля данных. Два связанных сегмента, расположенных на смежных уровнях называются исходным (более высокого

уровня) и порожденным (более низкого). Иерархическая запись - система взаимосвязанных сегментов, в которой каждый порожденный сегмент представлен столько раз, сколько необходимо для полного раскрытия данного сегмента. В иерархической структуре есть сегмент, который не имеет исходного и называется главным или корневым. В этом сегменте обычно располагается идентификатор объекта, свойства которого раскрываются в сегментах второго и более низких уровней иерархии.

Сетевая модель - более общая структура по сравнению с иерархией. Узлами сети есть отдельные экземпляры записи. Узлы записи являются единицей доступа к БД. Поскольку отдельный узел может иметь несколько непосредственно старших узлов, так же, как и несколько непосредственно подчиненных, то данная структура обеспечивает прямое представление отношения "многие ко многим". Для связи между записями-узлами существует связывающий запись, все экземпляры которой помещаются в цепочку для связи двух экземпляров. Реляционная база данных это база данных, построенная на основе реляционной модели, то есть БД, имеет табличный способ представления данных, а на внешнем уровне, задается набором однородных таблиц. Каждый объект записывается строкой в таблице. Строка называется записью. Запись состоит из полей разного типа. Реляционная база данных создается и затем управляется с помощью специальных средств - реляционных систем управления базами данных (СУБД). Строки таких таблиц называются кортежами, имена столбцов - атрибутами. Следует отметить, что все кортежи различны, а порядок столбцов произволен, чем упрощается процесс обработки кортежей. В отношении (таблице) выделяется несколько атрибутов так называемые ключевые поля. Особенность реляционной модели заключается в том, что в отличие от электрической и иерархической моделей реальные объекты и взаимосвязи между ними представляются в базе данных единообразно в виде нормализованных отношений. При разработке системы использовано реляционную технологию. Реляционная технология значительно упрощает эту задачу. Разделением логического и физического уровней системы она

упрощает процесс отображения "уровня реального мира", в структуру, которую система может прямо поддерживать. Поскольку реляционная структура сама по себе концептуально проста, она позволяет реализовывать небольшие и простые (и поэтому легкие для создания) БД, сама возможность реализации которых никогда даже бы не рассматривалась в старых более сложных системах.

MS Access в настоящее время является одной из самых популярных среди настольных программных систем управления базами данных. Среди причин такой популярности следует отметить:

* высокая степень универсальности и продуманности интерфейса, который рассчитан на работу с пользователями самой различной квалификации. В частности, реализована система управления объектами базы данных, позволяет гибко и оперативно переходить из режима конструирования в режим их непосредственной эксплуатации;
* глубоко развитые возможности интеграции с другими программными продуктами, которые входят в состав МS Office, а также с любыми программными продуктами, поддерживающими технологию OLE;
* богатый набор визуальных средств разработки.

Специфической особенностью СУБД MS Access является то, что вся информация, относящаяся к одной базе данных, хранится в едином файле. Такой файл имеет расширение .Mdb. Возможные источники данных для таблиц: 1) непосредственное введение, 2) импорт из других баз данных (FoxPro, dBASE и другие) или электронных таблиц (Excel, Lotus).

По вышеперечисленным причинам, для разработки ПО по учету успеваемости учеников в школе, будет выбран MS Access.

Типы данных, используемые при заполнении таблиц СУБД MS Access, описаны в таблице 1.

Таблица 1 – Таблица сущностей

|  |  |
| --- | --- |
| Сущность | Описание |
| Группы классов | Информация о группах классов и  заведующих ими |
| Классы | Информация о том, какой класс к  какой группе классов относится |
| Дисциплины | Информация о дисциплинах |
| Оценки | Информация о оценках учеников по  разным дисциплинам |
| Ученики | Информация об учениках |

Соответствие типов данных таблиц базы данных представлены в таблице 2.

Таблица 2 – Таблица соответствия сущностей таблицам

|  |  |  |
| --- | --- | --- |
| Таблица | Атрибуты | Тип |
| Группы классов | Зауч | Короткий текст |
| Наименование | Короткий текст |
| КодГруппыКлассов | Счетчик |
| Оценки | КодУченика | Числовой |
| КодДисциплины | Числовой |
| Оценка | Числовой |
| Дисциплины | КодДисциплины | Счетчик |
| Название  дисциплины | Короткий текст |
| ФИО учителя | Короткий текст |
| Классы | КодКласса | Счетчик |
| НомерКласса | Короткий текст |
| КодГруппыКлассов | Числовой |
| Год создания | Числовой |
| Ученики | КодУченика | Счётчик |
| Номер дневника | Короткий текст |

|  |  |  |
| --- | --- | --- |
|  | Фамилия | Короткий текст |
| Имя | Короткий текст |
| Отчество | Короткий текст |
| Баллы за семестр | Числовой |
| КодКласса | Числовой |
| Дата рождения | Дата и время |
| Город проживания | Короткий текст |
| Взнос в школу за  мес | Денежный |

Таблица 3 - Типы данных MS Access.

|  |  |  |
| --- | --- | --- |
| Тип данных | Содержание поля | Размер |
| 1. Текстовый | Значение по умолчанию. Текст или числа, не требующие проведения расчетов, например, номера телефонов | Число символов, не превышает 255. MS Access не хранит пустые символы в  неиспользуемой части  поля |
| 2. Поле Мемо | Длинный текст или  комбинация текста и цифр | До 65535 символов |

|  |  |  |
| --- | --- | --- |
| 3. Числовой | Числовые данные,  используемые для проведения расчетов | 1, 2, 4, 8 байт |
| 4. Дата / время | Даты и время, связанные с лет  с 100 по 9999, включая | 8 байт |
| 5. Денежный | Денежные значения и числовые данные, которые указываются в математических  расчетах, проводимых с | 8 байт |

|  |  |  |
| --- | --- | --- |
|  | точностью до 15 знаков в целой  и до 4 знаков в дробной части |  |
| 6. Счетчик | Уникальные последовательно возрастающие (на 1) или случайные числа, автоматически вводятся при добавлении каждой новой записи в таблицу. Значения полей типа Счетчик обновлять  нельзя | 4 байта |
| 7. Логический | Логические значения, а также поля, которые могут содержать одно из двух возможных значений (Да / Нет, Истина /  Ложь, Вкл / Выкл) | 1 бит |

|  |  |  |
| --- | --- | --- |
| 8. Поле объекта OLE | Объект (например, электрон- тронная таблица MS Excel, документ MS Word, рисунок, звукозапись или другие данные в двоичном формате), связанный или внедренный в  таблицу MS Access | До 1 Гбайт (ограниченно объемом диска) |
| 9.  Гиперссылка | Строка, состоящая из букв и цифр, и представляет адрес гиперссылки. | Каждая из трех частей в типе Гиперссылка может содержать до 2048  символов |
| 10. Мастер подстановок | Создает поле, в котором предлагается выбор значений из списка, или поля со списком, содержащий набор постоянных значений или значений из  другой таблицы. | Тот же размер, что и в ключевого поля, используемого в подстановке (обычно 4 байт) |

Разработка моделей данных

Определим основные сущности и сведем их описание табл. 4.

Таблица 4 - Сущности модели «Учет успеваемости учеников»

|  |  |  |
| --- | --- | --- |
| № | Сущность | Описание |
| 1 | Ученики | Информация о учениках |
| 2 | Оценки | Информация о оценках |
| 3 | Дисциплины | Информация о дисциплинах и кто их ведёт |
| 4 | Классы | Информация о классах |
| 5 | Группы классов | Информация о подвиде класса и его  руководителе |

Определим основные атрибуты сущностей и опишем их (табл. 5):

Таблица 5 - Атрибуты сущностей

|  |  |
| --- | --- |
| Сущность | Атрибуты сущности |
| Ученики | \*КодУченика, Номер дневника, Фамилия, Имя, Отчество, Баллы за семестр, КодКласса, Дата рождения, Город проживания, Взнос в школу  за мес |
| Оценки | \*КодУченика, \*КодДисциплины,  Оценка |
| Дисциплины | \*Код дисциплины, Название  Дисциплины, ФИО учителя |
| Классы | \*Код класса, НомерКласса,  КодГруппыКлассов, Год создания |
| Группы классов | \*КодГруппыКлассов,  Наименование, Зауч |

Связи между сущностями представим в табл. 6.

Таблица 6 - Связи между сущностями

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Родительская сущность | | Дочерняя сущность | | Тип связи |
| Название | Атрибут | Название | Атрибут |
| Ученики | КодУченика | Оценки | КодУченика | 1 до ∞ |
| Дисциплины | КодДисциплин ы | Оценки | КодДисциплин ы | 1 до ∞ |
| Классы | КодКласса | Ученики | КодКласса | 1 до ∞ |
| Группы классов | КодГруппыКла ссов | Классы | КодГруппыКла ссов | 1 до ∞ |

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СИСТЕМЫ –ИНТЕРФЕЙС И ВХОДНАЯ-ВЫХОДНАЯ ИНОФРМАЦИЯ

Графический интерфейс

На рис. 3 изображена возможная главная кнопочная форма, на которой находятся различные кнопки. Также находятся системная кнопка - это

«Выход». При нажатии на кнопку «Выход» программа полностью закрывается с сохранением последних изменений.

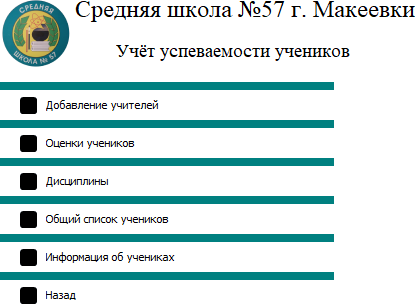


Рисунок 3 - Главная кнопочная форма

Ввод данных будет осуществляется с помощью одноименных форм. Перейти к вводу данных можно прямо с главной формы, щелкнув по кнопке

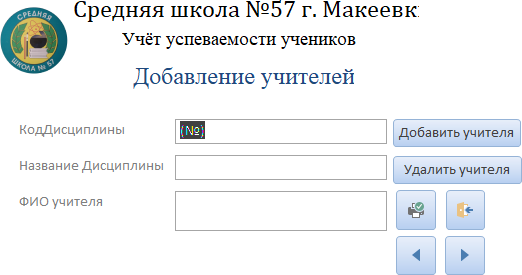
«Изменение данных», а после нажав на кнопку «Добавление учителей» или допустим «Оценки учеников». В результате этого откроются две формы, отражено на рис. 4 и рис. 5.

Рисунок 4 – Форма «Добавление учителей»

На форме есть поля со списком для удобства ввода некоторых данных. Поле со списком позволяет выбирать значение из перечня предложенных вариантов, а не делать это вручную. Эти элементы управления очень удобны при необходимости ввода часто повторяющихся значений.

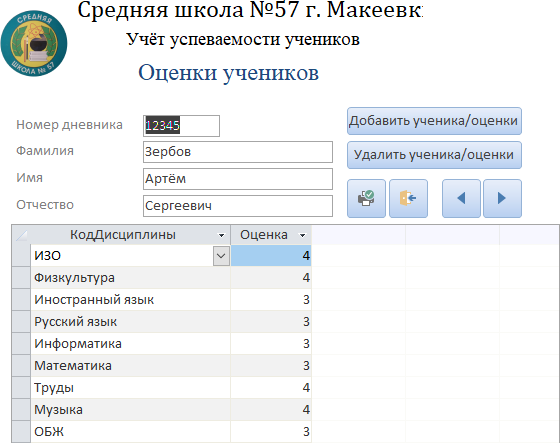


Рисунок 5 – Форма «Оценки учеников»

Работа с информацией, файлами и их выходными данными.

Эта часть ПО будет служит для организации доступа к различным запросам и таблицам, т.е. выходным данным.

Допустим, при нажатии на кнопку «Поиск по Фамилии и Имени» появится запрос, изображенный на рис. 6. Пользователь должен внести фамилию или имя, которые необходимые и на экран будет выведен отчет, изображенный на рис. 7.

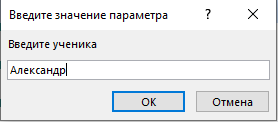


Рисунок 6 – Запрос «Поиск по Фамиии и Имени»

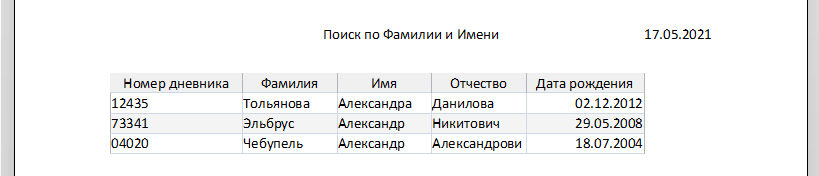


Рисунок 7 – Отчет «Поиск по Фамиии и Имени»

При нажатии на кнопку «Поиск учеников по дате рождения» пользователь должен ввести дату - рис.8. После ему на экран будет выведен список учеников, которые родились до этой даты - рис.9.

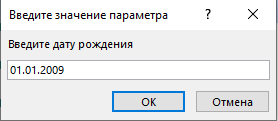


Рисунок 8 – Запрос «Поиск учеников по дате рождения»

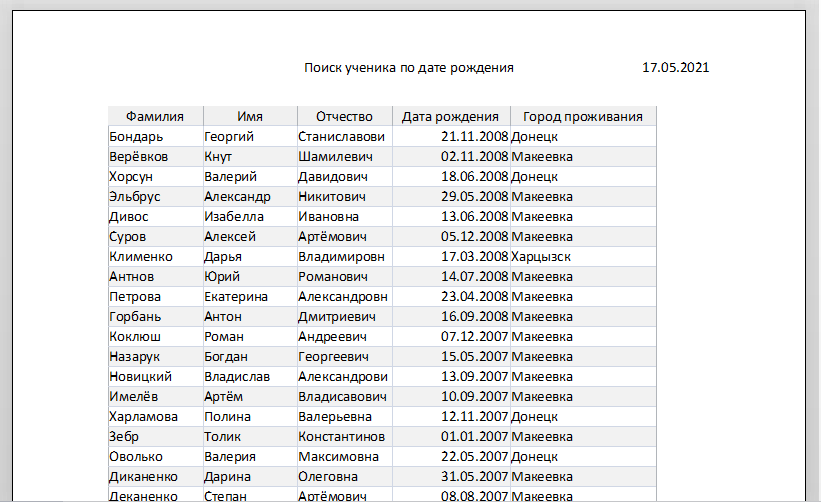


Рисунок 9 – Отчет «Поиск учеников по дате рождения» КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Предварительное проектирование формирует абстракции архитектурного уровня, детальное проектирование уточняет эти абстракции, добавляет подробности алгоритмического уровня. Кроме того, во многих случаях выделяют интерфейсное проектирование, цель которого — сформировать графический интерфейс пользователя (GUI).
2. Системные паттерны (system patterns) описывают системный (верхний) уровень приложения. Их использование может обеспечить построение более эффективных систем. Основные виды системных паттернов: Модель-Вид- Контроллер - разделение приложения или системы на три функциональные части (модель данных, представление (пользовательский интерфейс) и контроллер (управляющую логику)). Сессия, Рабочая нить - в системах распределѐнной обработки обеспечивает возможность серверам различать клиентов, что позволяет повысить эффективность многопоточных приложений Обратный вызов - организация асинхронного взаимодействия между клиентом и сервером, что снижает загрузку на сеть, повышает эффективность в использовании процессорного времени как клиента, так и сервера. Постоянное обновление - обеспечение

возможности для клиента постоянного автоматического получения обновлений от сервера.

1. Паттерны управления разделены на паттерны централизованного управления (то есть паттерны, в которых одна из систем полностью отвечает за управление, запускает и завершает работу остальных систем) и паттерны управления, подразумевающие децентрализованное реагирование на события, (согласно этим паттернам на внешние события отвечает соответствующая система.). Вызов – возврат - вызов программных процедур осуществляется "сверху - вниз", то есть управление начинается на вершине иерархии процедур и через вызовы передается на нижние уровни иерархии. Диспетчер - один системный компонент назначается диспетчером и управляет запуском и завершением других процессов системы и координирует эти процессы. Процессы могут протекать параллельно. Передача сообщений - В рамках данного паттерна событие представляет собой передачу сообщения всем системам. Любая система, которая обрабатывает данное событие, отвечает на него. Управляемый прерываниями

- при использовании данного паттерна внешние прерывания регистрируются обработчиком прерываний, а обрабатываются другим системным компонентом.

1. Связанность модуля является мерой взаимозависимости модулей. При создании систем необходимо стремиться к максимальной независимости модулей, т.е. связанность модулей должна быть минимальной. Модули связаны по данным, если они взаимодействуют через передачу параметров и при этом каждый параметр является элементарным информационным объектом. Это наиболее предпочтительный тип связанности (сцепления). Модули связаны по образцу если один модуль посылает другому составной информационный объект (например, объект – библиографическая запись, которая содержит имя автора, название книги и т.д.). Модули связаны по управлению, если один посылает другому информационный объект – флаг, предназначенный для управления его внутренней логикой. Модули связаны по общей области в том случае, если они ссылаются на одну и ту же область

глобальных данных. Связанность (сцепление) по общей области является нежелательным, так как, во-первых, ошибка в модуле, использующем глобальную область, может неожиданно проявиться в любом другом модуле; во-вторых, такие программы трудны для понимания, так как программисту трудно определить какие именно данные используются конкретным модулем. Модули связаны по содержимому в том случае, если один из них ссылается внутрь другого. Это недопустимый тип сцепления, ибо полностью противоречит принципу модульности, т.е. представления модуля в виде черного ящика.

1. Сцепление - мера взаимозависимости модулей по данным [3]. Сцепление - внешняя характеристика модуля, которую желательно уменьшать. Независимые модули могут быть модифицированы без переделки каких-либо других модулей. Количественно сцепление измеряется степенью сцепления (СЦ). Выделяют 7 типов сцепления.
2. Полностью независимые модули (СЦ = 0). Модули, не вызывающие друг друга и не использующие общих данных, не сцеплены и являются полностью независимыми. Чем больше информации о других модулях используется в них, тем менее они независимы и тем сильнее сцеплены.
3. Сцепление по данным. Модуль А вызывает модуль В. Все входные и выходные параметры вызываемого модуля - простые элементы данных.
4. Сцепление по образцу. В этом случае модули ссылаются на одну и ту же глобальную структуру данных. Недостатком такого сцепления является то, что оба модуля должны знать о внутренней структуре данных.
5. Сцепление по общей области. Модули разделяют одну и ту же глобальную структуру данных. В этом случае возможностей для появления ошибок при модификации структуры данных в одном модуле много больше.
6. Сцепление по управлению. Модуль А явно управляет функционированием модуля В с помощью передачи флагов, переключателей или кодов, посылая ему управляющие данные. Возвращение флага состояния как неявной переменной не означает сцепления по управлению. Если модуль передает информацию о самом себе или об обработанных данных, то это не

всегда служит проявлением сцепления по управлению. Передача флага конца файла позволяет решить вопрос о возможности обработки этого файла.

1. Сцепление по внешним ссылкам. Модуль имеет сцепление по внешним ссылкам, если у него есть доступ к данным в другом модуле через внешнюю точку входа.
2. Сцепление по кодам. Один модуль прямо ссылается на содержание другого модуля (не через его точку входа). Например, коды их команд перемежаются друг с другом.
3. Делегирование — основной паттерн проектирования, в котором объект внешне выражает некоторое поведение, но в реальности передаѐт ответственность за выполнение этого поведения связанному объекту. Шаблон делегирования является фундаментальной абстракцией, на основе которой реализованы другие шаблоны - композиция (также называемая агрегацией), примеси и аспекты. Неизменяемый объект — в объектно- ориентированном программировании объект, который не может быть изменѐн после своего создания. Интерфейс — основной паттерн проектирования, являющийся общим методом для структурирования компьютерных программ для того, чтобы их было проще понять. В общем, интерфейс — это класс, который обеспечивает программисту простой или более программно-специфический способ доступа к другим классам. MVC расшифровывается как модель-представление-контроллер. Это способ организации кода, который предполагает выделение блоков, отвечающих за решение разных задач. Один блок отвечает за данные приложения, другой отвечает за внешний вид, а третий контролирует работу приложения.