Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСТИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Инженерно-экономический факультет

Кафедра экономической информатики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по курсу:

«Объектно-ориентированное программирование»

на тему:

«РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ТЕСТИРОВАНИЯ ПО РАЗЛИЧНЫМ ТЕМАМ»

1-40 05 01-12

Студент гр. 924401 Мельников М.А

Руководитель асситент каф. Салапура М.Н.

Минск 2021

**ВВЕДЕНИЕ**

История развития тестов своими корнями уходит в глубину веков, она связана с измерением различных способностей, знаний, умений и навыков. Эти измерения можно считать предысторией тестов. Уже в середине III тысячелетия до н. э. в Древнем Вавилоне проводились испытания выпускников в школах писцов на знание арифметических действий, распределять рационы, делить имущество и так далее.

Первым этапом применения тестов в мировой практике можно считать период с 80-х годов XIX века до 20-х годов XX века. Этот период характеризуется зарождением и становлением тестирования.

К концу XIX века практическая проблема исследования индивидуальных различий стала толчком к появлению первых тестов. С 80-х годов XX века по настоящее время начинается четвертый этап в развитии тестирования. Он характеризуется широким использованием компьютеров, математических моделей, средств программирования для обучения и контроля.

Основным недостатком тестирования – является разработка качественного тестового инструментария. Это длительный, трудоемкий и дорогостоящий процесс. Стандартные наборы тестов для большинства дисциплин ещё не разработаны, а разработанные обычно имеют очень низкое качество.

Целью данного курсового проекта является повышения качества тестирования, уменьшения времени на разработку вопросов, проверки ответов, вывода результатов для тестируемых. Эти задачи реализуются с помощью разделения ролей, предоставления каждой роли своего базового функционала, который позволяет для первой роли быстро создавать, удалять, редактировать и проверять различные тесты, а для второй - максимально быстро проходить тестирование и выводить результат в удобной для пользователя форме.

1 ОБЗОР АВТОМАТИЗИРУЕМОЙ ПРЕДМЕТНОЙ ОБЛАСТИ, ПРОГРАММНЫХ АНАЛОГОВ, МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

* 1. Обзор предметной области

Тестирование — это форма измерения знаний учащихся, основанная на применении педагогических тестов. Включает в себя подготовку качественных тестов, собственно проведение тестирования и последующую обработку результатов, которая даёт оценку обученности тестируемых.

Тест — это инструмент оценивания обученности учащихся, состоящий из системы тестовых заданий, стандартизованной процедуры проведения, обработки и анализа результатов.

Тесты можно классифицировать по различным признакам:

– по целям — информационные, диагностические, обучающие, мотивационные, аттестационные;

– по процедуре создания — стандартизованные, не стандартизованные;

– по способу формирования заданий — детерминированные, стохастические, динамические;

– по технологии проведения — бумажные, в том числе бумажные с использованием оптического распознавания, натурные, с использованием специальной аппаратуры, компьютерные;

– по форме заданий — закрытого типа, открытого типа, установление соответствия, упорядочивание последовательности;

– по наличию обратной связи — традиционные и адаптивные

Традиционный тест содержит список вопросов и различные варианты ответов. Каждый вопрос оценивается в определенное количество баллов. Результат традиционного теста зависит от количества вопросов, на которые был дан правильный ответ. Иными словами, традиционный тест — система заданий, предъявляемая в порядке увеличения сложности в одно и то же время, с одинаковой системой оценивания для всех тестируемых

Существуют различные **формы тестовых заданий**, а именно:

1 Задания с выбором ответов. Разделяются на задания с выбором одного правильного или неправильного ответа. Один вопрос и несколько вариантов ответа, где обязательно только один из них является (не)верным.

2 Задания на установления соответствия. Состоит из таблицы, где имеется две колонки: в первой находится либо вопрос, либо часть утверждения, во второй ответ на вопрос или вторая часть утверждения. Вопрос и соответствующий ответ практически всегда находятся в разных строках. Также два вопроса могут иметь один и тот же ответ.

3 Задания с выбором нескольких правильных ответов. Суть задания схожа заданием с выбором ответов, разница лишь в том, что правильных ответов может быть более одного.

4 Упорядочивание последовательности. Примером такого задания может служить расположение в хронологическом порядке исторических событий.

5 Задания с открытым ответом. Ответ на поставленный вопрос тестируемый дает в виде введенного предложения.

Тестирование выполняет две основные взаимосвязанные **функции**: диагностическую и обучающую:

1 Диагностическая функция заключается в выявлении уровня знаний, умений, навыков тестируемого. Это основная и самая очевидная функция тестирования. По объективности, широте и скорости диагностирования, тестирование превосходит все остальные формы педагогического контроля.

2 Обучающая функция тестирования состоит в мотивировании тестируемого к активизации работы по усвоению предоставленного материала. Для усиления обучающей функции тестирования могут быть использованы дополнительные меры стимулирования студентов, такие как: раздача преподавателем примерного перечня вопросов для самостоятельной подготовки, наличие в самом тесте наводящих вопросов и подсказок, совместный разбор результатов теста.

**Достоинства:**

1 Тестирование является более качественным и объективным способом оценивания, его объективность достигается путём стандартизации процедуры проведения, проверки показателей качества заданий и тестов целиком.

2 Тесты — это более объёмный инструмент, поскольку тестирование может включать в себя задания по всем темам курса, в то время как на устный экзамен обычно выносится 2-4 темы, а на письменный — 3-5. Это позволяет выявить знания учащегося по всему курсу, исключив элемент случайности при вытаскивании билета. При помощи тестирования можно установить уровень знаний учащегося по предмету в целом и по отдельным его разделам.

3 Тестирование более эффективно с экономической точки зрения. Основные затраты при тестировании приходятся на разработку качественного инструментария, то есть имеют разовый характер. Затраты же на проведение теста значительно ниже, чем при письменном или устном контроле. Проведение тестирования и контроль результатов в группе из 30 человек занимает полтора два часа, устный или письменный экзамен — не менее четырёх часов.

**Недостатки**:

1 Данные, получаемые преподавателем в результате тестирования, хотя и включают в себя информацию о пробелах в знаниях по конкретным разделам, но не позволяют судить о причинах этих пробелов.

2 Тест не позволяет проверять и оценивать высокие, продуктивные уровни знаний, связанные с творчеством, то есть вероятностные, абстрактные и методологические знания.

3 Обеспечение объективности и справедливости теста требует принятия специальных мер по обеспечению конфиденциальности тестовых заданий. При повторном применении теста желательно внесение в задания изменений.

* 1. Обзор программных аналогов

В качестве программного аналога рассматривается программа для создания тестов и онлайн тестирования INDIGO.

Система тестирования INDIGO – это профессиональный инструмент автоматизации процесса тестирования и обработки результатов, который предназначен для решения широкого спектра задач:

1 Тестирование, контроль знаний учеников и студентов.

2 Определение профессионального уровня сотрудников, оценка персонала (HR).

3 Подготовка к сдаче экзаменов и аттестаций.

4 Психологическое тестирование.

5 Организация и проведение опросов, олимпиад, конкурсов.

**Функциональные возможности.** Данная система тестирования предоставляет широкий спектр возможностей, однако далее будут перечислены лишь важнейшие:

1 Система тестирования устанавливается на главный компьютер (сервер тестирования) с помощью инсталляционного пакета.

2 Система может работать как на изолированном компьютере, так и в локальной сети или через Интернет.

3 Центр тестирования можно развернуть на Вашем компьютере или в облаке на наших Интернет-серверах.

4 Все данные хранятся централизованно в базе данных системы.

5 Администраторы работают через программу клиент.

6 Одновременно могут работать сколько угодно администраторов с разных компьютеров.

7 Система имеет многоязычный пользовательский web-интерфейс и полностью поддерживает символы всех языков (Unicode).

**Функции администратора:** создание и редактирование тестов;управление базой пользователей;назначение тестов пользователям;проведение тестирования; просмотр результатов;построение отчетов и анализ статистики.

**Функции пользователя**: регистрация и авторизация; выбор теста; прохождение тестирования; просмотр результатов и ошибок.

Доступно 578 бесплатных тестов (на 66400 вопросов с пояснениями) на различные темы, начиная от психологических тестов и заканчивая ЕГЭ.

**Функции управления тестами:** добавление, редактирование, удаление; перемещение между группами; поиск; копирование; блокировка, разблокировка; защита на редактирование паролем; разграничение прав доступа между администраторами.

**Функции управления пользователями**: добавление, редактирование, удаление, перемещение между группами; поиск; блокировка, разблокировка; просмотр журнала событий; разграничение прав доступа между администраторами.

**Функции, связанные с результатами:** сортировка записей; многоуровневая группировка записей (по выбранным столбцам); фильтрация (выборку) записей по сложным условиям; экспорт всей таблицы или выделенных записей в файл (форматы Excel, HTML, XML, TXT); поиск записей в таблице; настройку выводимых столбцов.

**Общие функции:** просмотр протокола тестирования, который включает:

– общую информацию (о пользователе, тестировании и результатах);

– подробную информацию в виде образа теста, который проходил пользователь с учетом всех перемешиваний и случайных выборок, а также ответы пользователя на каждый вопрос;

создание отчетов (с возможностью печати или экспорта в Word):

– отчет по результату;

– отчет по пользователю;

– общий отчет по выборке результатов.

**1.3 Обзор методов и алгоритмов решения поставленной задачи**

Целью данного курсового проекта является создание автоматизированной системы тестирования. Программа разрабатывалась с учетом объектно-ориентированного программирования. Система тестирования поддерживает разделение ролей, а именно: студент и преподаватель, каждый из которых представляет класс (Student, Teacher) с собственным набором методов. Основным используемым объектом в системе является тест, также представленный отдельным классом (Test), который, в свою очередь, содержит список объектов класса Question. Для хранения как пользователей (студентов и преподавателей) и тестов используется отдельный пользовательский класс (Database), который перед предоставлением интерфейса считывает необходимую информацию из файлов, а после завершения пользовательского сеанса записывает измененную информацию обратно в файлы.

Функционал студента представлен такими действиями как:

– решение тестов;

– просмотр доступных тестов;

– просмотра личной информации;

– просмотра решенных тестов кратко и отдельно каждый подробно;

– вывод доступных и уже решенных тестов в отсортированном виде.

Перед предоставлением списка доступных тестов класс Database производит фильтрацию тестов по личным характеристикам студента, а именно по его группе и перечню изучаемых дисциплин.

Функционал преподавателя представлен следующими возможностями:

– создание, удаление, редактирование тестов;

– просмотр личной информации;

– просмотр доступных тестов;

– просмотр всех студентов, в которых он преподает;

– просмотр решенных тестов конкретного студента;

– вывод на экран студентов и доступных тестов в отсортированном виде.

Сортировка производится с помощью базовых методов сортировки контейнеров STL, для которых необходимо написать собственный компаратор. Поиск по контейнерам, хранящим информацию о тестах, студентах и преподавателях ведется в основном линейно, так как большая часть представлен контейнером list. Фильтрация производится на основе сравнения характеристик, которые должны совпадать как у фильтрованного объекта, так и у объекта, которому вышеназванный необходим для взаимодействия.

В процессе работы программы под хранение данных в классе Database динамически выделяется и очищается память. Для оптимизации работы большинства функций в качестве параметров передаются не сами объекты, а указатели, хранящие адрес на динамически выделенный участок память под объект.

2 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0

3 СТРУКТУРА ИСПОЛЬЗУЕМЫХ ДАННЫХ

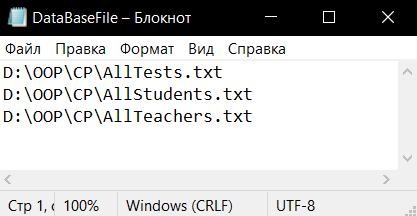
Все входные данные программы хранятся в файлах с расширением «.txt». Файл DataBaseFile хранит три строки, обозначающие пути к файлам, хранящим все пути файлов с тестами, студентами и преподавателями.

Рисунок 3.1 – Содержимое файла DataBaseFile

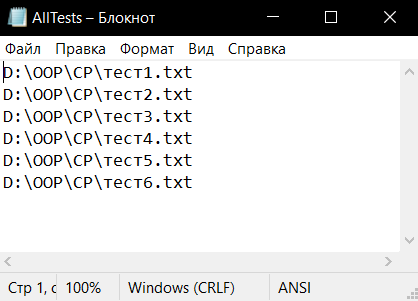
На рисунке 3.2 представлено содержимое файла AllTests, а именно пути ко всем существующим тестам.

Рисунок 3.2 – Содержимое файла AllTests

В файлах AllStudents и AllTeachers также находятся пути ко всем существующим студентам и преподавателям, соответственно. На рисунке 3.3 приведен пример файла с тестом, где формат заполнения следующий:

– количество ответов в вопросах;

– курс, для которого тест предназначен;

– ID теста;

– предмет, для которого тест предназначен;

– краткое описание;

– вопрос;

– варианты ответа;

– количество баллов за вопрос;

– правильный вариант ответа.

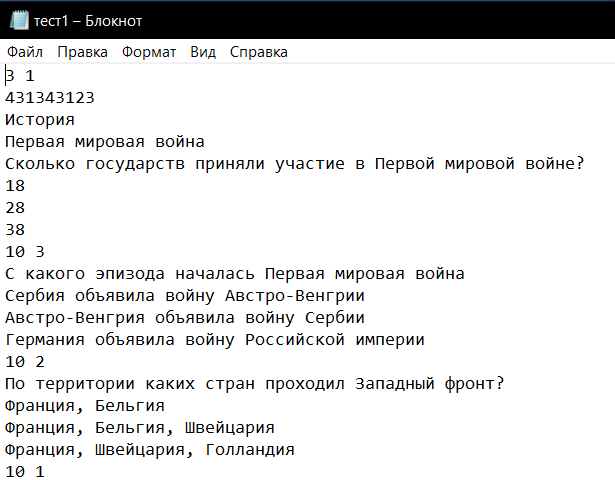
Последние четыре пункта повторяются столько раз, сколько находится вопросов в самом тесте.

Рисунок 3.3 – Заполнение файла для теста

На рисунке 3.4 приведен пример файла студента, с следующим форматом заполнения:

– ФИО;

– пароль в хешированном виде;

– ID;

– факультет;

– группа;

– курс;

– количество учебных дисциплин и их названия, каждая с новой строки;

– количество решенных тестов;

– ID теста;

– предмет, для которого тест предназначен;

– краткое описание;

– количество набранных и максимально возможных баллов;

– количество ответов и сами ответы (в цифрах).

Последние пять пунктов повторяются столько же раз, сколько тестов решил студент.

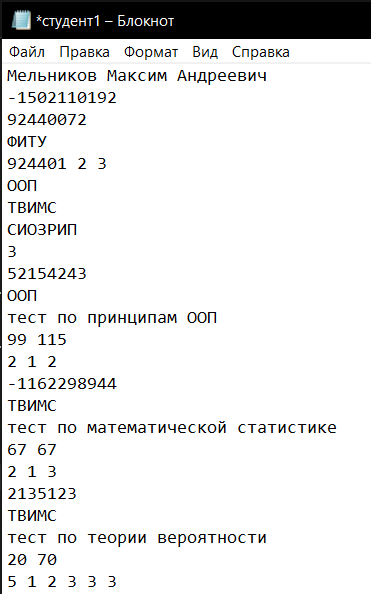


Рисунок 3.4 – Пример файла с информацией о студенте

Последним типом файлов является файл, в котором содержится информация о преподавателе. Пример приведен на рисунке 3.5, где формат заполнения следующий:

– ФИО;

– пароль в хешированном виде;

– ID;

– преподаваемый предмет;

– количество групп, в которых он преподает;

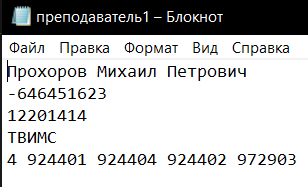
– номера групп.

Рисунок 3.5 – Пример файла с информацией о преподавателе

Считывание данных происходит при создании объекта класса Database, точнее при вызове его конструктора, так как этот класс хранит все существующие тесты, студентов и преподавателей. Перед выходом из программы происходит запись измененных данных в те же файлы, откуда они были считаны.

4 РАЗРАБОТКА И ОПИСАНИЕ ДИАГРАММЫ КЛАССОВ ПРИЛОЖЕНИЯ

После анализа предметной области для реализации автоматизированной системы тестирования по различным темам было разработано семь классов, диаграмма которых представлена в приложении Б на рисунке Б1. Названия всех классов:

– User;

– Student;

– Teacher;

– DataBase;

– Test;

– Question;

– SolvedTest.

1 Класс User – абстрактный класс, предназначенный для наследования дочерними классами, Student и Teacher.

Данный класс содержит набор следующих полей:

– entityCount – статическое поле, предназначенное для отсчета количества текущих объектов данного класса, в том числе и дочерних, служит для создания уникального идентификатора.

– fullName – для хранения ФИО пользователя;

– hashedPassword – для хранения пароля пользователя уже в хешированном виде;

– id – уникальный идентификатор пользователя.

Список методов, реализованных в классе:

– User – конструктор по умолчанию и принимающий параметры для заполнения полей;

– ~User – виртуальный деструктор, чтобы вызывались деструкторы дочерних классов в случае динамического полиморфизма;

– GetID, GetHashedPassword, GetName, GetEntityCount – геттеры полей класса;

– PrintInformation – чисто виртуальный метод для вывода содержимого объекта, переопределен в дочерних классах;

– PrintUserInformation – метод для вывода информации о пользователе (его поля) на экран;

– Searching – метод поиска пользователя на основе совпадения пароля и идентификатора;

– Unload – чисто виртуальный метод для записи полей класса в файл.

Диаграмма класса представлена на рисунке 4.1.

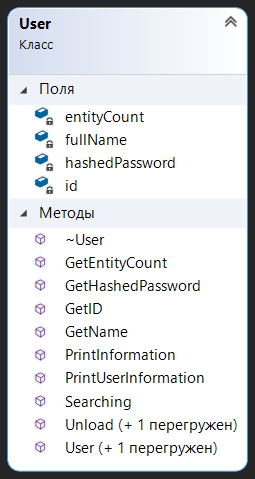


Рисунок 4.1 – Диаграмма класса User.

2 Класс Student предназначен для входа в автоматизированную систему в качестве студента с соответствующим функционалом.

Данный класс содержит следующий набор полей:

– course – курс студента;

– faculty – факультет студента;

– group – группа студента;

– ptrSolvedTestList – указатель на список указателей тестов, решенных студентом;

– ptrSubjectList – смарт-указатель на список предметов студента;

– MenuChoice – перечисление, содержащее константы для выбора в меню.

Для реализации всего функционала были созданы методы, описанные ниже:

– Student – конструктор по умолчанию и принимающий аргументы для заполнения полей класса;

– ~Student – деструктор, в котором очищается динамически выделенная память;

– DeleteSolvedFromFilteredList – фильтрация доступных тестов, уже решенные перемещаются в отдельный список;

– DeleteEditedSolvedTest – метод удаленного теста, если он был изменен преподавателем;

– GetName, GetGroup, GetCourse, GetPtrSolvedTestList, GetPtrSubjectList – геттеры полей классса

– Menu – метод, вызываемый после входа в систему и предоставляющий интерфейс для взаимодействия с системой;

– PrintAllSolvedTest – вывод на экран решенных тестов;

– PrintAvailableAndNoSolvedTest – вывод на экран фильтрованных тестов (доступных и еще не решенных);

– PrintInformation – вывод на экран личной информации, переопределенный метод;

– Unload – запись в файл информации о студенте, переопределенный метод.

Диаграмма класса представлена на рисунке 4.2.

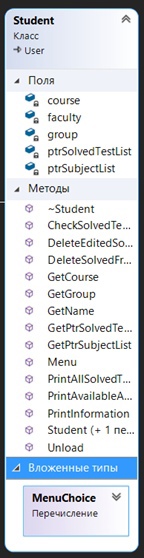


Рисунок 4.2 – Диаграмма класса Student

3 Класс Teacher предназначен для входа в автоматизированную систему в качестве студента с соответствующим функционалом.

Данный класс содержит поля:

– ptrGroupList – смарт-указатель на список групп преподавателя;

– subject – предмет, который он преподает;

– MenuChoice – перечисление для удобного отображения интерфейса

– QuestionMenu – перечисление для удобного отображения интерфейса изменения вопроса

Для реализации функционала были разработаны методы:

– Teacher – конструктор по умолчанию и принимающий аргументы для заполнения полей класса;

– ~Teacher – деструктор;

– CreateQuestion – метод для создания нового вопроса в тесте;

– CreateTest – метод для создание нового теста;

– GetHashedPassword, GetSubject, GetPtrGroupList – геттеры полей;

– Menu - метод, вызываемый после входа в систему и предоставляющий интерфейс для взаимодействия с системой;

– PrintAvailableTest – вывод доступных для взаимодействия тестов;

– PrintInformation – переопределяемый метод для вывода личной информации;

– PrintOwnStudents – вывод студентов, в группах которых ведет преподаватель;

– SearchAvailableTest – поиск доступного теста в отфильтрованном списке;

– Unload – переопределяемый метод для записи в файл.

Диаграмма класса представлена на рисунке 4.3.

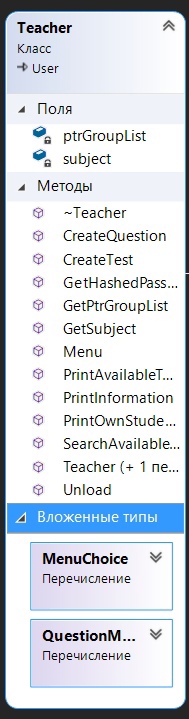


Рисунок 4.3 – Диаграмма класса Teacher

4 Класс DataBase предназначен для хранения всех существующих тестов, студентов и преподавателей с помощью контейнеров STL. Также он предоставляет возможности регистрации и входа для пользователей, фильтрацию тестов на основе личной информации пользователя.

Поля класса:

– listOfStudents – список для хранения указателей на студентов;

– listOfTeachers – список для хранения указателей на преподавателей;

– listOfTests – список для хранения указателей на тесты;

– MenuChoice – перечисление для удобного отображения интерфейса;

– UserType – перечисление для отображения типа пользователя.

Методы класса, реализованные в данном курсовом проекте:

– DataBase – конструктор по умолчанию и принимающий в качестве параметра путь файла, из которого считывается вся необходимая информация;

– ~DataBase – деструктор, очищающий память, выделенную под объекты классов тестов, преподавателей и студентов;

– AuthorizationMenu – метод для авторизации, выбора типа пользователя;

– LoadStudentsFilter – предоставление отфильтрованного списка студентов для преподавателя;

– LoadTestWithFilter – предоставление отфильтрованного списка тестов для преподавателя и студента;

– Login – логин студента или преподавателя;

– PrintAllStudents, PrintAllTeachers, PrintAllTests – вывод на экран всех загруженных студентов, преподавателей и тестов (для отладки);

– Registration – регистрация студента или преподавателя;

– Unload – запись всей считанной ранее из файлов информации обратно в те же файлы.

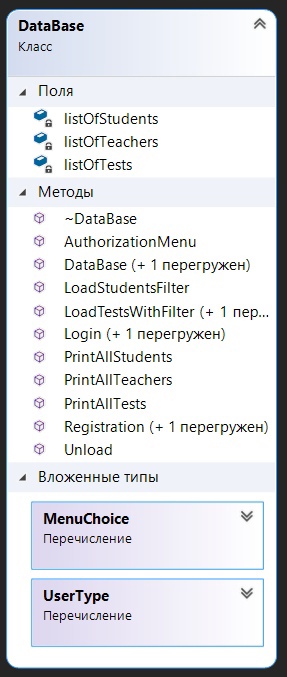
Диаграмма класса представлена на рисунке 4.4.

Рисунок 4.4 – Диаграмма класса DataBase

5 Класс Test предназначен для взаимодействия с тестами, решения, просмотра, создания. Имеет набор полей:

– course – курс, для которого предназначен тест

– entityCount – статическая переменная, обозначающая количество объектов класса, служит для создания уникальных идентификаторов;

– maxPointsPerTest – максимальное количество баллов за тест;

– numberOfQuestions – количество вопросов в тесте;

– ptrQuestionList – указатель на список вопросов;

– shortDescription – краткое описание теста;

– subject – предметная область теста;

– uniqueID – идентификатор теста;

Класс содержит следующие методы:

– Test – конструктор по умолчанию, конструктор, принимающие все необходимые параметры для создания нового объекта, конструктор, принимающий путь файла для считывания теста;

– ~Test – деструктор, который удаляет динамические выделенную память под поля теста;

– DownloadFromFile – чтение данных теста из файла;

– GetCourse, GetEntity, GetSubject, GetID, GetMaxPointsPerTest, GetNumbeOfQuestions – геттеры полей класса;

– PrintTest – вывод теста на экран с вопросами;

– PrintTestBriefly – вывод на экран только необходимой информации о тесте;

– SetMaxPointsPerTest – сеттер для максимального значения баллов;

– Solving – решение теста, путем ввода ответов и создания объекта класса SolvedTest;

– UnloadTest – запись теста в файл.

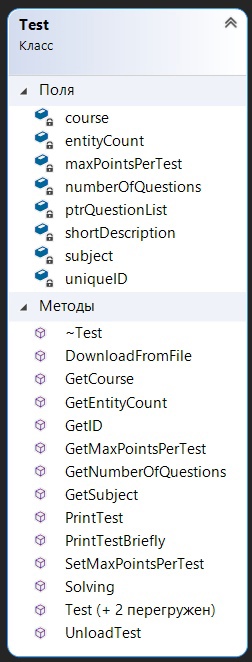
Диаграмма класса представлена на рисунке 4.5.

Рисунок 4.5 – Диаграмма класса Test

6 Класс Question предназначен для работы с минимальной структурной единицей любого теста – вопроса. Включает в себя поля:

– answerOptions – указатель на массив строк вариантов ответа на вопрос;

– correctAnswerOption – правильный вариант ответа;

– numberOfAnswers – количество вариантов ответа;

– pointsPerQuestion – количество баллов за вопрос;

– question – сам вопрос;

Класс также содержит методы:

– Question – конструктор по умолчанию, а также конструктор, принимающий в качестве параметров все необходимые для создания вопроса поля;

– ~Question – деструктор, очищающий динамически выделенную память под поля класса;

– GetCorrectAnswerOption, GetNumberOfAnswers, GetPoints – геттеры полей класса;

– PrintCorrectAnswer – вывод на экран правильного ответ;

– PrintQuestion – вывод на экран вопроса с вариантами ответа;

– UnloadQuestion – запись информации о вопросе в файл.

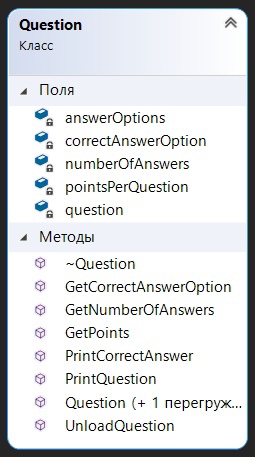
Диаграмма класса представлена на рисунке 4.6.

Рисунок 4.6 – Диаграмма класса Question

7 Класс SolvedTest предназначен для хранения ответов студента на вопросы, количество полученных баллов, информации о решенном тесте. Имеет следующие поля:

– answers – указатель на массив ответов студента;

– maxPoints – максимальное количество баллов за тест;

– receivedPoints – полученные баллы за тест;

– shorDescription – краткое описание решенного теста;

– subject – предметная область решенного теста;

– uniqueID – уникальный идентификатор теста;

Для реализации функционала были написаны методы:

– SolvedTest – конструктор, принимающий в качестве параметров все необходимые переменные для создания объекта класса;

– ~SolvedTest - деструктор, очищающий динамически выделенную память под поля класса;

– GetID, GetPercent, GetSubject – геттеры полей класса.

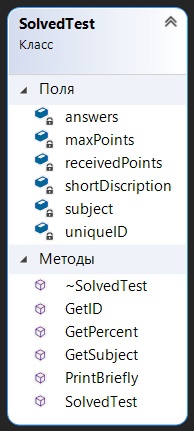
Диаграмма класса представлена на рисунке 4.7.

Рисунок 4.7 – Диаграмма класса Question

5 РАЗРАБОТКА И ОПИСАНИЕ ДИАГРАММЫ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ ПРИЛОЖЕНИЯ

Диаграмма вариантов использования — диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.

При моделировании системы с помощью диаграммы прецедентов системный аналитик стремится:

– чётко отделить систему от её окружения;

– определить действующих лиц (акторов), их взаимодействие с системой и ожидаемую функциональность системы;

– определить в глоссарии предметной области понятия, относящиеся к детальному описанию функциональности системы (то есть прецедентов).

Работа над диаграммой может начаться с текстового описания, полученного при работе с заказчиком. При этом нефункциональные требования (например, конкретный язык или система программирования) при составлении модели прецедентов опускаются.

Диаграмма вариантов использования для автоматизированной системы тестирования по различным темам представлена в приложении В на рисунке В1.

В данной диаграмме присутствует два актора – студент и преподаватель. Они имеют как одинаковые, так и различные прецеденты. Как студент, так и преподаватель имеют возможности:

– зарегистрироваться, если до этого они ни разу не входили в систему;

– войти в систему, если уже прошли регистрацию и их данные имеются в системе, для этого нужно ввести пароль и логин;

– просмотреть свою личную информацию, логин, пароль, ФИО, а также информацию, соответствующую типу актора.

Для студента в основном все прецеденты представлены взаимодействием со списками доступных и уже решенных тестов. Эти списки выдает база данных сразу после входа в систему пользователем. Критерий фильтрации тестов строится на основе личной информации пользователя. Среди доступных тестов студент может выбрать тест для решения, либо просмотреть весь список кратко. Среди решенных тестов студент может просмотреть один тест полностью, то есть увидеть вопрос, свои варианты ответа и количество баллов, которое он набрал за вопрос. Также он может просмотреть весь список решенных тестов кратко. Оба списка доступны для сортировки по определенным критериям (доступные – по предмету, решенные – по предмету и проценту правильных ответов).

Для преподавателя все прецеденты представлены взаимодействием со списками доступных тестов и студентов, в группах которых он преподает. Фильтрация происходит также на основе личных данных пользователя. С доступными тестами преподаватель может делать следующие действия:

– создать новый тест, который добавляется в список доступных;

– удалить тест, удаление идет также и из списка;

– редактировать конкретный тест, его краткое описание, либо какой-то один вопрос. В вопросе он может изменить сам вопрос, варианты ответа, правильный вариант ответа, количество баллов за вопрос;

– просмотреть весь список тестов кратко, в табличном виде либо выбрать один тест для детального просмотра.

В списке студентов преподаватель может выбрать одного студента и просмотреть его решенные тесты, их название и количество баллов, которые студент набрал. Также есть возможность просмотра всех студентов преподавателя в табличном виде, то есть кратко.

6 СХЕМА АЛГОРИТМА РАБОТЫ ВСЕЙ ПРОГРАММЫ И АЛГОРИТМА РАБОТЫ ДВУХ И БОЛЕЕ ОСНОВНЫХ МЕТОДОВ

В приложении Г на рисунке Г1 приведена схема алгоритма работы всей программы. Перед предоставлением доступа пользователю к интерфейсу программы производится загрузка всей необходимой информации путем чтения из файлов. Схема алгоритма чтения из файла представлена на рисунке Г2 приложения Г.

На этом этапе производится тщательная проверка сперва на открытие файлов с данным, а затем на считанные данные. Если было вызвано исключение, то оно будет обработано в конструкторе класса DataBase и приведет к тому, что будет выведено на экран сообщение об ошибке, некоторые объекты созданы не будут (если исключение возникло в конструкторе). Однако программа продолжит корректно работать и предложит пользователю сначала выбрать пройти регистрацию или ввести свои данные для входа.

После того, как пользователь смог зайти в систему, класс DataBase на основе личных данных пользователя производит фильтрацию списков и передает их в пользование последнего с помощью вызова метода Menu и передачи в качестве аргументов отфильтрованных списков.

После выполнения всех желаемых действий пользователь выходит из программы, но она работу на этом не прекращает. Перед выходом необходимо записать все данные обратно в файлы. Схема алгоритма записи данных обратно в файлы представлен на рисунке Г3 приложения Г. Сперва идет открытие файла. Затем создается итератор, указывающий на начало списка. Увеличивая в цикле значение итератора на единицу, алгоритм проходит весь список, пока не будет достигнут элемент, следующий за концом списка. В теле сперва создается путь файла, в который впоследствии будет записан объекта, а затем вызывается метод класса, отвечающий за запись объекта в файл. В качестве параметра он принимает строку, которая является раннее созданным путем файла.

7 ОПИСАНИЕ АЛГОРИТМА ЗАПУСКА ПРИЛОЖЕНИЯ, ЕГО ИСПОЛЬЗОВАНИЯ, РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ, ТЕСТИРОВАНИЯ ОБРАБОТКИ ОШИБОК

Заключение

Список использованных источников

Приложения (листинг программы с комментариями, диаграмма классов, диаграммы вариантов использования, схемы алгоритмов).

ПРИЛОЖЕНИЕ А

ПРИЛОЖЕНИЕ Б

(обязательное)

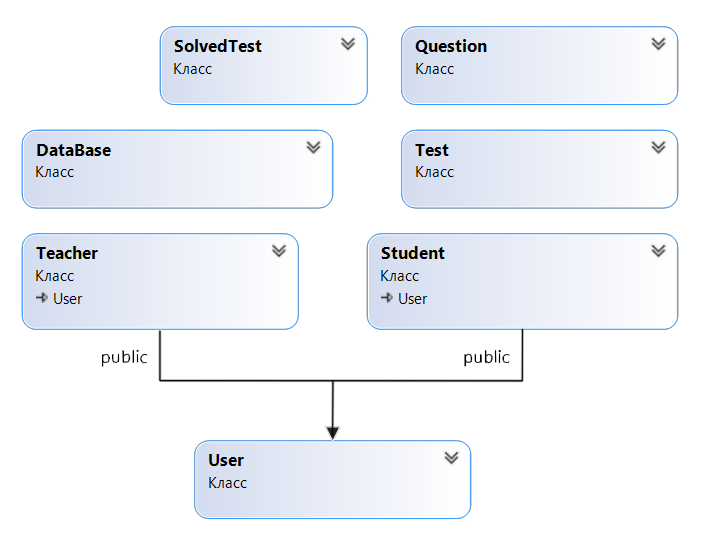
Диаграмма классов

Рисунок Б1 – диаграмма классов автоматизированной системы тестирования по различным темам

ПРИЛОЖЕНИЕ В

(обязательное)  
Диаграмма вариантов использования

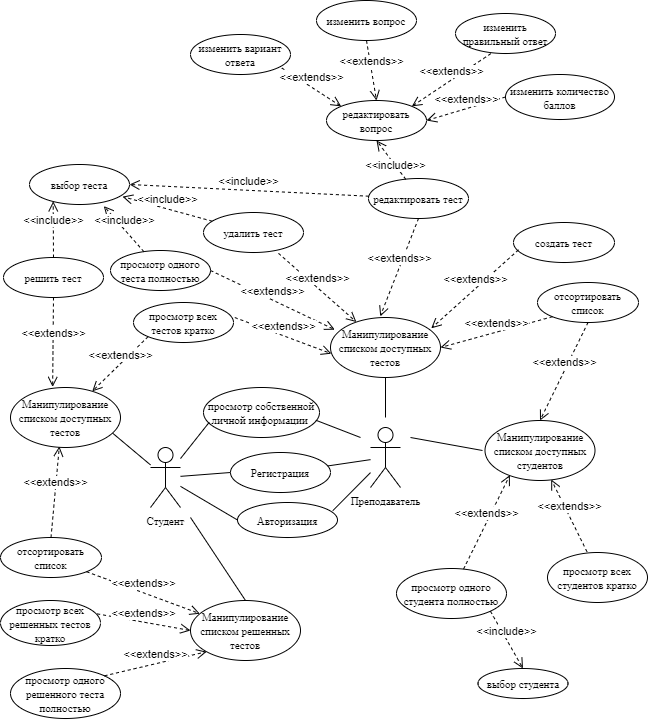


Рисунок В1 – диаграмма классов автоматизированной системы тестирования по различным темам

ПРИЛОЖЕНИЕ Г

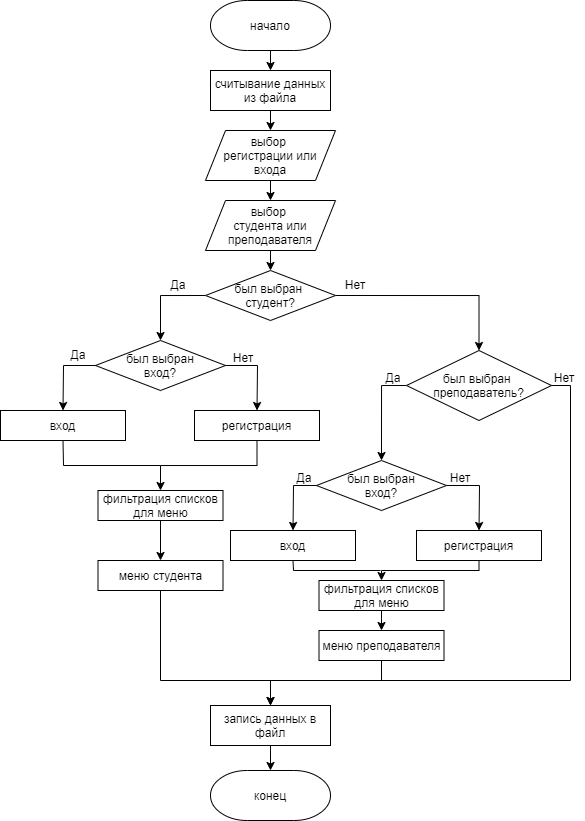
(обязательное)  
Схема алгоритмов

Рисунок Г1 – схема алгоритма работы всей программы

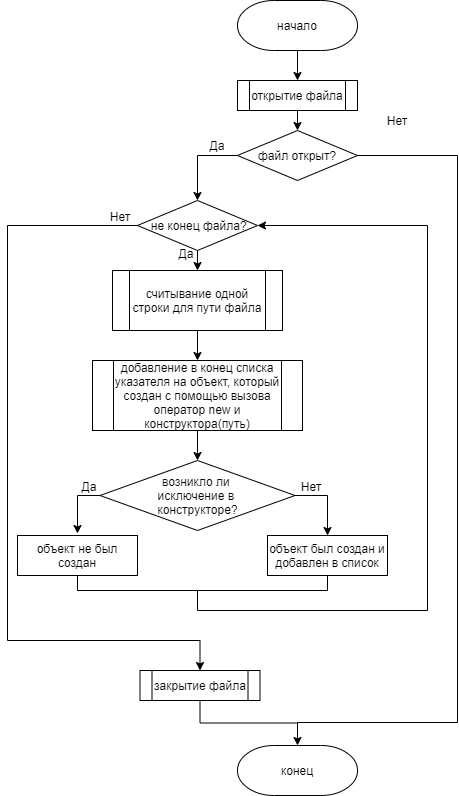
Продолжение приложения Г

Рисунок Г2 – схема алгоритма чтения из файла

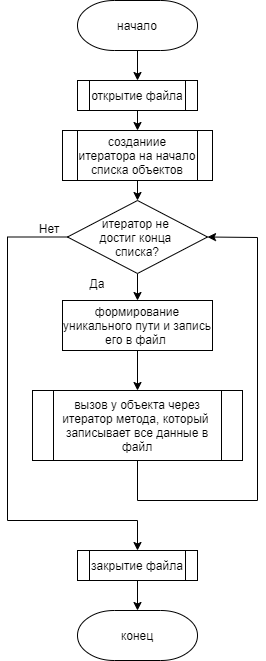
Продолжение приложения Г

Рисунок Г3 – схема алгоритма записи в файл