!!!!!!!!!!!!!!!!!!!!!!!!

Подписать все картинки и таблицы

Реферат

Записка 34 с., 3 ч., 24 рис., 6 табл., 4 источника, 2 прил.

Результатом выполнения работы является программа «Энциклопедия». Программа предназначена для ознакомления с разнообразными объектами космоса и закрепления полученной информации в игровой форме.

Цель работы – разработка программы предоставляющей информацию по космосу и возможность закрепления материала.

Возможности программы:

* Вывод информации по объектом звездного неба
* Игра вопрос-ответ
* Просмотр статистики пользователя

Программой может использоваться на смартфонах под управлением ОС Андроид, версия которой выше версии 2.3.3.

**Содержание**

Введение 5

1 Анализ требований и уточнение спецификаций 6

1.1 Исследование предметной области 6

1.2 Выбор технологии, языка и среды разработки 7

1.3Построение диаграммы вариантов использования 8

2 Проектирование структуры и компонентов программного продукта 10

2.1 Проектирование интерфейса программного продукта 10

2.1.1 Проектирование диаграммы классов интерфейсной части 10

2.1.2 Проектирование структуры меню 12

2.1.3 Построение графа состояния интерфейса 13

2.1.4 Выбор типа интерфейса и формы диалога 14

2.1.5 Разработка форм ввода-вывода информации 15

2.2 Разработка алгоритмов методов 16

3 Выбор стратегии тестирования и разработка тестов 21

Заключение 24

Литература 25

Приложение А. Техническое задание.

Приложение Б. Руководство пользователя.

Введение

Смотря на звездное небо, хочется ориентироваться в созвездиях, знать названия, знать расположение и быстро находить их. Для этого надо выучить все созвездия, но на это нужно потратить немало времени, которое так не хватает в современном мире. В реальной жизни не каждый может позволить себе потратить десяток вечеров для изучения всего материала.

Разработанное приложение написано под ОС Андроид и позволяет просматривать созвездии, планеты и звезды в любом месте в любое время.

Существующие аналоги, такие как «SkyMap», «StarWalk» или «StarTracker», дают пользователю ограниченный функционал, либо карту неба, либо названия созвездий, либо просто теорию по созвездиям и другим объектам космоса. Еще есть интернет ресурс Google Sky, но для использования нужен хороший интернет.

Разработанное приложение хранит информацию во встроенной в ОС базе данных. Таким образом можно быстро получить доступ к данным по хранящимся там темам (созвездия, планеты и т.д.). А также пользователь имеет возможность проверить или закрепить свои знания по созвездиям и планетам.

В данной работе поставлена задача создать удобное и быстрое приложение, которым можно воспользоваться в любое время при наличии смартфона на базе ОС Андроид.

Многие люди любят проводить свободное время, наблюдая за звездами, складывая из них определенные рисунки, находя созвездия. Еще в древности многие люди видели в небе картинки, которые напоминали людей, животных или какие-либо предметы. Так возникла, например, Большая Медведица, которая на сегодняшний день является наиболее популярным и широко известным созвездием. На сегодняшний день нам известно 88 созвездий, в состав которых входят все звезды, которые мы видим на небе.

Очень часто пользователю интересно название и расположение созвездий и звезд. Надо предоставить возможность пользователю найти созвездие на небе по картинке, словесному описанию положения относительно других объектов неба. Не все приложения могут предоставить полную информацию по этой теме.

Помимо этого у каждого созвездия есть интересная история о том почему они так называются, история открытия.

Все созвездия состоят из звезд, которые получили свои имена еще в древности. Эти звезды светятся по-разному из-за того, что имеют разную структуру и относятся к разным типам. На данный момент существую более 14 типов звезд. О которых пользователю будет интересно знать.

**1. Анализ требований и уточнение спецификаций**

**1.1 Выбор технологии, языка и среды разработки**

Технологией программирования называют совокупность методов и средств, используемых в процессе разработки программного обеспечения. Для разработки программы «Энциклопедия» было принято решение использовать спиральную схему разработки из трех известных схем разработки (а именно схема с промежуточным контролем и каскадная схема разработки), потому что позволяет создавать программу поэтапно и последовательно.

Для написания программы необходимо было выбрать язык программирования и среду разработки. К разрабатываемой программе были сформулировала следующие функциональные требования:

* Поддержка объектного подхода;
* Событийное программирование;
* Модульность, с использованием правил вертикального управления.

На данный момент принято разрабатывать приложения для ОС Андроид на языке Java, так как наиболее удобная среда разработки Android Studio использует язык программирования Java.

Язык Java объектно-ориентированный язык программирования, в соответствии с этой идеологий при разработке интерфейса пользователя будет использован объектный подход в совокупности с технологией событийного программирования.

В качестве среды разработки использовал Android Studio. Android Studio много функциональная среда предназначенная для разработки мобильных приложений на базе ОС Андроид. Среда разработки имеет удобный инструмент для работы с SDK и предоставляет быстрый доступ к созданию и управлению AVD, что ускоряет тестирование и проверку работоспособности приложения на разных этапах разработки.

**1.3 Построение диаграммы вариантов использования**

В соответствии с ТЗ были выделены варианты использования. Варианты использования представлены в таблицах 1-6.

Таблица 1

|  |  |
| --- | --- |
| Название варианта | **Просмотр данных об объектах** |
| Цель | Получение сведений по объектам звездного неба |
| Действующие лица | Пользователь |
| Краткое описание | Пользователь выбирает режим «Энциклопеди», затем выбирает категорию, и выбирает объект. Выбрав объект пользователю предоставляется информация по объекту. |
| Тип варианта | Основной |

Таблица 2

|  |  |
| --- | --- |
| Название варианта | **Просмотр данных об категории.** |
| Цель | Получение сведений по объектам звездного неба |
| Действующие лица | Пользователь |
| Краткое описание | Пользователь выбирает режим «Энциклопеди», затем выбирает категорию. Выбрав категорию пользователю предоставляется информация по категории. |
| Тип варианта | Основной |

Таблица 3

|  |  |
| --- | --- |
| Название варианта | **Игра** |
| Цель | Прохождение игры |
| Действующие лица | Пользователь |
| Краткое описание | Пользователь выбирает режим «Игра», затем нажимает на кнопку «Начать игру», выбирает тему игры, отвечает на 10 вопросов, выбирая ответы на вопросы из представленных вариантов ответов. |
| Тип варианта | Основной |

Таблица 4

|  |  |
| --- | --- |
| Название варианта | **Добавление статистики** |
| Цель | Добавить статистики после прохождения игры пользователем. |
| Действующие лица | Система |
| Краткое описание | После прохождение игры пользователь попадает на экран результатов. Программа сохраняет результат в файл |
| Тип варианта | Дополнительный |

Таблица 5

|  |  |
| --- | --- |
| Название варианта | **Просмотр статистики** |
| Цель | Посмотреть общую статистику по играм |
| Действующие лица | Пользователь |
| Краткое описание | Пользователь выбирает режим «Игра», нажимает на кнопку «Статистика». На экране появляется статистика |
| Тип варианта | Основной |

Таблица 6

|  |  |
| --- | --- |
| Название варианта | **Удаление статистики** |
| Цель | Удалить статистику |
| Действующие лица | Пользователь |
| Краткое описание | Пользователь выбирает режим «Игра», нажимает на кнопку «Статистика», и нажимает на кнопку «Удалить статистику», в появившемся диалоговом окне пользователь нажимает на кнопку «Да». Статистика обнуляться и записывается в файл |
| Тип варианта | Дополнительный |

Приведем варианты использования «Просмотр данных об объектах и «Игра» подробно, таблицах 7-11:

Вариант использования «**Просмотр данных об объектах**»

Таблица 7

|  |  |
| --- | --- |
| Действия пользователя | Отклик системы |
| 1. Пользователь выбирает режим «Энциклопедия» | 1. Система выводит на экран список категорий |
| 1. Пользователь выбирает категорию | 1. Система выводи на экран списк объектов, относящихся к выбранной категории. |
| 1. Пользователь выбирает объект | 1. Система вывод на экран информацию по этому объекту. |

Вариант использование «**Игра**»

Таблица 8

|  |  |
| --- | --- |
| Действия пользователя | Отклик системы |
| 1. Пользователь выбирает режим «Игра» | 1. Система выводит экран с игровым меню. |
| 1. Пользователь выбирает один из вариантов. 2. Кнопка «Начать игру» см. вариант использования Игровой процесс. 3. Кнопка «Статистика» см. вариант использования Статистика. 4. Кнопка «Назад» | 4. Если пользователь выбрал кнопку Назад, то он возвращается на предыдущий экран. |
|  |  |
|  |  |

Вариант использования «**Игровой процесс**»

Таблица 9

|  |  |
| --- | --- |
| Действия пользователя | Отклик системы |
| 1 |  |
| 1. Пользователь выбирает тему | 1. Система случайным образом выбирает 10 вопросов и выводит на экран первый вопрос. |
| 1. Пользователь отвечает поочерёдно на 10 вопросов | 1. Система вывод на экран результаты ответов и сохраняет их в файл. |

Объединив таблицы можно построить общую диаграмму вариантов использования, это пожжет наглядно представить ожидаемое поведение системы и уточнить варианты использования данной программы.



Рис 2. Диаграмма вариантов использование.

**1.2 Выделение объектов из предметной области**

Анализ ТЗ и предметной области позволил выделить следующие объекты:

* Созвездие – объект, описывающий созвездие реального мира
* Планета – объекты, описывающий планеты как солнечной системы, так и находящиеся за пределами(экзопланеты), также карликовые планеты.
* Объект категория – объект, описывающий категории, в которых нет объектов.
* Объект вопрос – объект, описывающий вопрос, содержит вопрос в виде картинки и 4 варианта ответа.

Перечисленные объекты будут хранится в базе дынных, структура которой будет уточв разделе разработкибд.

На картинке опечатка(База данных)



Рис1. Диаграмма концептуальной модели предметной области

**1.4 Описание поведения. Системные события**

Для уточнения поведения варианта использования «**Просмотр данных об объектах**» проводим анализ предметной области и получаем, что пользователь должен инициализировать 3 события. Диаграмма последовательностей системы представлена на рисунках 3



Рис 3. Диаграмма последовательностей системы варианта использования «**Просмотр данных об объектах**»

Анализ варианта использования «**Игровой процесс**» позволяет выделить 4 события, которые пользователь должен инициализировать. Диаграмма последовательностей системы варианта использования «**Игровой процесс**» представлена на рисунке 4.



Рис 4. Диаграмма последовательностей системы варианта использования «**Игровой процесс**»

На основе двух предшествующих диаграмм последовательностей системы был сформирован список системных операций, приведённый на рисунке 5.



Рис 5. Системные операции

Для уточнения системных операций следует выполнить подробное описание систем опер. Для этого необходим вып анализ предметную область и ТЗ. Результат анализа представлен в таблицах 12 – 13.

Таблица 11

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | ***Выбрать режим()*** |
| Обязанности | *Вывести экран, соответствующий выбранному режиму.* |
| Тип | *Системная* |
| Ссылки | *Вариант использования Просмотр данных об объектах* |
| Примечания | *—* |
| Исключения | *—* |
| Вывод | *—* |
| Предусловия | *Предполагается на выбор два режима* |
| Постусловия | *Пользователь попадает на экран, выбранного режима* |

Таблица 12

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | ***Выбор тему()*** |
| Обязанности | *Вывести экран с вопросами, по выбранной теме.* |
| Тип | *Системная* |
| Ссылки | *Вариант использования Игровой процесс* |
| Примечания | *Предусмотреть выборку из базы данных 10 случайных вопросов по теме* |
| Исключения | 1. *Если в базе данных по выбранной теме нет 10 вопросов, то вывести все имеющиеся* 2. *Если в базе дынных нет вопросов по выбранной теме, то сообщить пользователю об этом* |
| Вывод | *—* |
| Предусловия | *Предлагается выбрать тему для игры* |
| Постусловия | *Пользователь попадает на экран, где ему надо ответить на 10 вопросов* |

Таблица 13

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | ***Выбор ответ()*** |
| Обязанности | *Получить ответ пользователя* |
| Тип | *Системная* |
| Ссылки | *Вариант использования Игровой процесс* |
| Примечания | *Предусмотреть смену вопроса после выбора ответа* |
| Исключения | *—* |
| Вывод | *—* |
| Предусловия | *Пользователю предлагается вопрос и 4 варианта ответа* |
| Постусловия | *Смена вопроса на следующий вопрос.* |

**1.6 Построение диаграмм деятельностей**

Для уточнения поведения вариант использования «**Просмотр данных об объектах»** проведём анализ предметной области и последовательности системы варианта использования «**Просмотр данных об объектах**»(Рис.3). Получаем, что системные операции «Выбрать режим, Выбрать категорию, Выбрать объект» будут вызываться последовательно. Покажем это на диаграмме деятельностей на рисунке 6.



Рис 6. Диаграмма деятельностей варианта использования

«**Просмотр данных об объектах**»

Для уточнения поведения вариант использования «**Игровой процесс»** проведём анализ предметной области и последовательности системы варианта использования «**Игровой процесс**». Получаем, что системные операции «Выбор режима «Игра и Выбор темы» будут вызываться последовательно, а системная операция «Ответ на вопрос» будет вызываться, пока пользователь не ответит на все вопросы. Покажем это на диаграмме деятельностей на рисунке 7.



Рис 7. Диаграмма деятельностей варианта использования «**Игровой процесс**»

**1. 7 Выбор типа и формы диалога пользовательского интерфейса**

Различают четыре типа пользовательских интерфейсов:

* *примитивные* - реализуют единственный сценарий работы, например, ввод данных - обработка - вывод результатов;
* *меню* - реализуют множество сценариев работы, операции которых организованы в иерархические структуры;
* *со свободной навигацией* - реализуют множество сценариев, операции которых не привязаны к уровням иерархии, и предполагают определение множества возможных операций на конкретном шаге работы; интерфейсы данной формы в основном используют Windows-приложения;
* *прямого манипулирования* - реализуют множество сценариев, представленных в операциях над объектами, основные операции инициируются перемещением пиктограмм объектов мышью, данная форма реализована в интерфейсе самой операционной системы Windows альтернативно интерфейсу со свободной навигацией.

На основе анализа диаграммы вариантов использования программы, был выбран интерфейс типа меню. Такая организация даст возможность пользователю быстро освоиться в программе. Перемещаться по меню пользователь будет с помощью сенсорного экрана.

Для удобства пользователя было решено выбрать табличную форму диалога. Табличная форма диалога удобна для пользователя тем, что выбрать всегда проще, чем вспомнить в отличии от директивной формы, в которой предполагается вводит команды для пользования приложением.

**1.8 Разработка структуры меню.**

Для обеспечения быстрой и удобной навигации пользователя по программе, навигацию целесообразно реализовать в виде иерархического меню. Анализ предметной области и диаграммы вариантов использования позволил разработать меню, структура которой представлена на рис 8.

****

Рис 8. Структура меню.

**1.9 Построение графа состояний интерфейса.**

На основе анализа диаграммы вариантов использования и структуры меню был построен граф состояний интерфейса. Таким графом называют ориентированный взвешенный граф, каждая вершина которого сопоставлена конкретная картинка на экране или определенное состояние диалога, характеризующееся набором доступных пользователю действий. Дуги, исходящие из вершин, показывают возможные изменения состояний при выполнении пользователем указанных действий. В качестве весов дуг указывают условия переходов из состояния в состояние и операции, выполняемые во время перехода. Граф состояний интерфейса представлен на рисунке 9.



Рис 9 . Граф состояний интерфейса.

**2. Проектирование программного обеспечения**

**2.1 уточнение структуры программного обеспечения**

Анализ предметной области, концептуальной модели, вариантов использования позволяет и системных событий позволил выделить следующие группы классов:

* Пользовательский интерфейс – классы, реализующие объекты интерфейса с пользователем
* Библиотека интерфейсных компонентов – классы, реализующие интерфейсные компоненты кнопки, текстовые поля и т.п.
* Объекты базы данных – классы, реализующие объекты предметной области.
* Интерфейсы базы данных – классы, реализующие интерфейс с базой данных
* База данных – класс, реализующий взаимодействие с базой дынных.
* Базовые структуры данных – классы, реализующие внутренние структуры данных, такие как списки, деревья, множества.
* Обработка ошибок – классы исключений, реализующие обработку нештатных ситуаций

Эти пакеты и их зависимости представлены на рисунке 10.



Рис 10. Диаграмма пакетов.

**2.2 Уточнение объектов и проектирование классов**

В ходе анализа концептуальной модели предметной области и варианта использования, а также в ходе уточнения атрибутов, были выделены следующие классы предметной области и определены отношения между ними.

Все объекты предметной области имеют общие поля:

* name – имя объекта
* intId – идентификатор объекта

Для этого я выделил объект MyObject, который будет родителем для всех объектов предметной области.

Класс ViewObject реализует класс «Категория». Этот класс наследуется от класс MyObject и расширяет его своими полями. Класс ViewObject нужен для работы с категориям, не имеющих объектов. И помимо полей базового класса, класс должен иметь поле для хранения информации описания категории, которую реализует объект этого класса.

Поля класса ViewObject:

* Text – текст описание для объекта категории

Классы PlanetObject, ConstellationObject и QuestionObject являются классами, реализующими объекты неба, и имею общее поле название картинки, и поэтому для них я выделил класс родитель SkyObject, в которой вынесено общее поле. Класс наследуется от класса MyObject.

Поля класса SkyObject:

* img – название картинки объекта неба

Класс PlanetObject реализует объекты планеты. Класс наследуется от класс SkyObject и расширяет его необходимыми полями достаточными для описания планеты.

Поля класса PlanetObject:

* mass – масса планеты
* radius – средний радиус планеты
* day – продолжительность дня
* year – продолжительность года
* radiusSun – средний радиус орбиты обращение вокруг солнца
* info – информация по планете, например история открытия, химический состав поверхности и т.п.

Класс ConstellationObject – это класс реализующий объекты созвездия. Наследуется от класса SkyObject.

Поля класса ConstellationObject:

* textWhereFrom – поля хранит информацию как, где и когда можно найти созвездие на небе.
* textInf – поле хранит информацию когда было открыто созвездие и историю созвездия из римской или греческой мифологии.

Класс QuestionObject – класс реализует объекты-вопросы, используются в игровом процессе. Наследуется от класс SkyObject.

Поля класс QuestionObject:

* answers – список ответов
* countAns – количество ответов
* rightAbswer – правильный ответ

Все класс имеют методы Seter’s для установки значений полей и методы Geter’s для получения данных полей.

Класс QuestionObject, помимо выше описанных методов, имеет дополнительные методы:

* isRightAnswer() – метод возвращает значение логического типа, истина, если пользователь ответил на вопрос правильно и ложь, если пользователь ответил не правильно на вопрос.
* mixAnswers() – метод перемешивания списка ответов.

На рисунке 11 представлена подробная диаграмма классов предметной области.

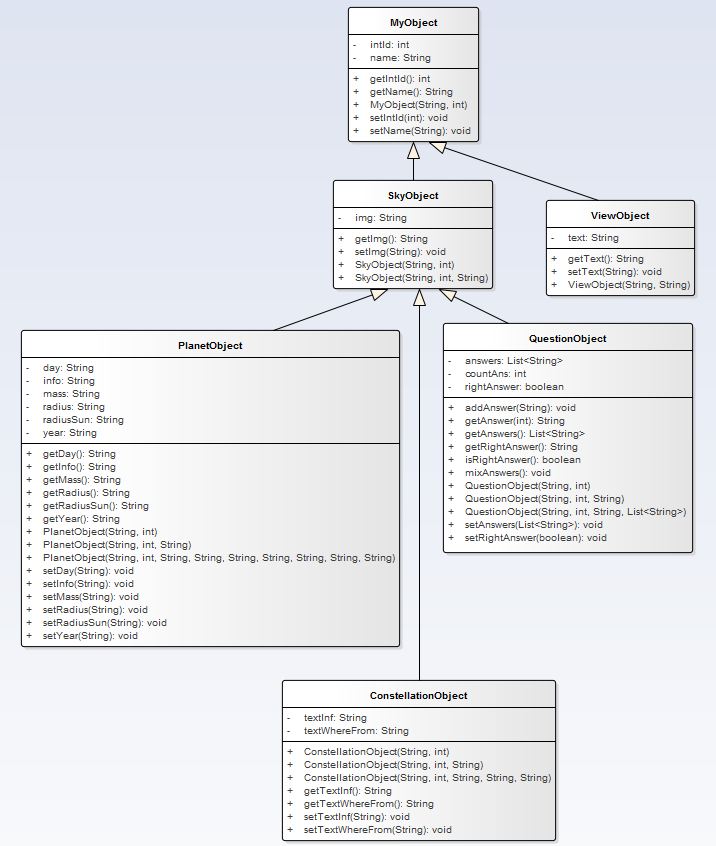


Рис 11. Диаграмма классов.

**2.3 Проектирование базы данных**

Данные по созвездиям, планетам, объекты звездного неба и список тем надо хранить в телефоне. Хранить данные в файлах не целесообразно, так как в процессе работы приложения надо будет производить выборку данных. Если данные хранятся в файлах, то выборку данных производить очень затратно. Поэтому было принято решения спроектировать базу данных.

В базе надо хранить данные по объектам:

* Созвездия(ConstellationObject)
* Планеты(PlanetObject)
* Объекты неба(SkyObject)
* Категории(ViewObject)
* Темы

Для каждого пункта создадим таблицы в базе данных:

* Constellation –хранится информация по созвездиям
* Planet – хранится информация по планетам
* Sky\_objects – хранятся объекты звездного неба
* View\_obj – хранится информация по категориям
* Themes – хранятся темы для игры.

Структура таблиц см в таблицах 14 – 17.

Таблица constellation

Таблица 14

|  |  |
| --- | --- |
| Id | Идентификатор |
| Title | Название созвездия |
| Int\_id | Уникальный идентификатор в приложении |
| Img | Название картинки |
| Text\_where\_from | Информация, где можно найти созвездие на небе |
| Text\_inf | Дополнительная информация |

Таблица planet

|  |  |
| --- | --- |
| Id | Идентификатор |
| Title | Название планеты |
| Int\_id | Уникальный идентификатор в приложении |
| Img | Название картинки |
| Mass | Масса планеты |
| Radius | Средний радиус планеты |
| Day | Продолжительность дня |
| Year | Продолжительность года |
| Radius\_sun | Радиус орбиты |
| Info | Дополнительная информация |

Таблица sky\_objects

Таблица 15

|  |  |
| --- | --- |
| Id | Идентификатор |
| Title | Название |
| Int\_id | Уникальный идентификатор в приложении |
| Img | Название картинки |

Таблица view\_obj

Таблица 16

|  |  |
| --- | --- |
| Id | Идентификатор |
| Title | Название |
| Text | Информация по объектам |

Таблица Themes

Таблица 17

|  |  |
| --- | --- |
| Id | Идентификатор |
| Title | Название |
| Int\_id | Уникальный идентификатор в приложении |

На рисунке 12 представлена спроектированная база данных.



Рис 12.

**2.3 Проектирование классов работы с базой данных**

В операционной системе Андроид есть возможность работы с базой данных SQLite. Работа с базой данных осуществляется с помощью класса SQLiteOpenHelper.

Для работы с базой реализуем два класса: DBHelper и SkyDataBase.

Класс DBHelper наследуется от стандартного класса SQLiteOpenHelper и переопределяет два метода (onCreate() и onUpgrade()). В этом классе будут поля определяющие структуру базы данных(названия таблиц, колонок, sql команды для создания таблиц)

Класс SkyDataBaseImpl – это класс, которые реализует основную работу с базой данных.

Для класса SkyDataBaseImpl реализован интерфейс SkyDataBase.

На рисунке 13 представлена диаграмма описанных выше классов.

****

Рис 13.

Класс DBHelper содержит все необходимую информацию о структуре база дынных и имеет следующие методы:

|  |  |
| --- | --- |
| DBHelper() | Конструктор |
| onCreate() | Создание структуры БД |
| onUpgrade() | Изменение существующей структуры БД |
| onUpdata() | Обновление БД |
| onCreateTableThemes() | Создание таблицы theme |
| onCreateTableScore() | Создание таблицы score |
| onCreateTableSkyObjects() | Создание таблицы sky\_objects |
| onCreateTableConstelltions() | Создание таблицы constelltion |
| onCreateTablePlanets() | Создание таблицы planet |
| onCreateViewObject() | Создание таблицы view\_objects |

Класс SkyDataBaseImpl содерзит следующие методы:

|  |  |
| --- | --- |
| SkyDataBase() | Конструктор класса |
| onUpdataDB() | Обновление БД |
| getListSkyObjects() | Получение списка объектов неба |
| getListConstellationSimply() | Получение упрощенного списка созвездий |
| getConstellationById() | Получение созвездия по имени |
| getConstellationIdList() | Получение списка id созвездий |
| getListPlanetsSimply() | Получение упрощенного списка планет |
| getPlanetById() | Получение планеты по id |
| getPlanetIdList() | Получение списка id планет |
| getThemes() | Получение списка тем |
| getQuestions() | Получение списка вопросов |
| getQuestionsOfConstellation() | Получение списка вопросов созвездий |
| getQuestionOfPlanet() | Получение списка вопросов планет |
| addScore() | Добавление статистики |
| getScore() | Получение статистики |
| deleteStatistic() | Удаление статистики |
| getViewObjectByName() | Получение списка объектов звездного неба |
| logCursor() | Вывод результатов запросов в лог-файл |

**2.4 Разработка диаграмм компоновки системы**

На рисунке 14 представлена структурная схема компоновки разработанной системы.



Рис 14.

1 – файлы, все классы проекта, кроме классов реализующие интерфейс.

Перечислим их: ConstellationObject.java, MyObject.java, PlanetObject.java, QuestionObject.java, SkyObject.java, ViewObject.java, DBHelper.java, SkyDataBase.java, SkyDataBaseImpl.java

2 – скомпилированные класс описанные в пункте 1.

3 – классы проекта, реализующие интерфейс:

MainActivity.java, HelpActivity.java, ConstellationViewActivity.java, EncyclopediaActivity.java, ObjectsListActivity.java, PlanetViewActivity.java, ViewActivity.java, ChoiceGameActivity.java, GameActivity.java, GameMenuActivity.java, GameOverActivity.java, GameOverShowQuestionActivity.java, StatisticsActivity.java

4 – скомпилированные класс описанные в пункте 3.

5 – xml файлы задающие структуру экрана.

Перечислим их: activity\_choice\_game.xml, activity\_constellation\_view.xml, activity\_game.xml, activity\_game\_menu.xml, activity\_game\_over.xml, activity\_help.xml, activity\_list\_1.xml, activity\_main.xml, activity\_planet\_view.xml, activity\_settings.xml, activity\_statistics.xml, activity\_view.xml, button\_back\_item.xml, button\_item.xml, list\_item\_result.xml, list\_item\_text.xml, list\_item\_text\_with\_img.xml, tab\_1\_header\_1.xml, tab\_1\_inf\_1.xml, tab\_1\_inf\_2.xml, tab\_2\_header\_1.xml, tab\_2\_story\_1.xml, tab\_2\_story\_2.xml.

6 – картинки проекта

7 – файлы задающие стиль для компонентов экрана

8 – файлы ресурсы.

color.xml – содержит цвета-константы, используемые в проекте

dimens.xml – содержит размеры-константы

strings.xml – строковые константы

style.xml – стили

**3 Проектирование пользовательского интерфейса**

**3.1 Проектирование игрового экрана.**

На игровом экране пользователь имеет возможность ответить на вопрос, выбрав один из 4 вариантов ответа. Вопросы представлены в виде картинок, на котором изображен объект звездного неба. Так на экране расположено 4 кнопки. Одно кнопка с правильным ответом, и три кнопки с ложными ответами. Еще на экране есть строка с количеством пройденных вопросов.

Строка, пройденных вопросов, реализована с помощью класса TextView, кнопки с помощью класса Button, а изображение с помощью класса ImageView.

Структура экрана задается в xml файле. Здесь приведу упрощённую структуру файла activity\_game.xml:

<LinearLayout>

<LinearLayout>

<TextView/>

<TextView/>

</LinearLayout>

<LinearLayout>

<ImageView/>

</LinearLayout>

<TableLayout>

<TableRow>

<Button/>

<Button/>

</TableRow>

<TableRow>

<Button/>

<Button/>

</TableRow>

</TableLayout>

</LinearLayout>

На рисунке 15 представлен интерфейс игрового экрана.



Рис 15. Интерфейс игрового экрана

**3.2 Проектирование классов для реализации интерфейса приложения**

Анализ показал, что класс игрового интерфейса содержит 4 кнопки(Button), класс картинки(ImageView) и текстовое поле(TextView). Картинка является вопросом, пользователь должен определить, что нарисовано на картинке. Кнопки – это варианты ответа, нажав ан одну из них, пользователь делает выбор. После того как пользователь ответил на вопрос, отображается следующий вопрос или, если вопрос был последним, переходит на экран результатов.

Класс игрового интерфейса называется GameActivity. Он наследуется от класса AppCompatActivity. Для того чтобы интерфейс мог реагировать на нажатие на кнопки класс должен реализовать интерфейс OnClickListerner.

На рисунке 16 представлена диаграмма класса игрового интерфейса(GameActivity).

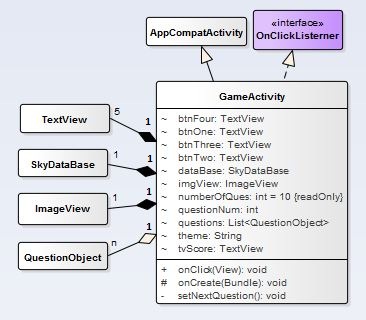


Рис 16. Диаграмма класса GameActivity.

Остальные класс интерфейса разработаны аналогичным образом.

Опишем методы onClick() и setNextQuestion() подробно:

При нажатии кнопки на экране, вызывает обработчик события зажатия onClick(). В обработчике фиксируется игранный ответ и, если вопрос не последний, то вызывается метод setNextQuestion(), которые выводит на экран следующий вопрос. Если вопрос последний, то открывается экран результатов.

Покажем алгоритмы методов onClick() и setNextQuestion() на рисунках 17 - 18.



Рис. 17. Схема алгоритма onClick()



Рис 18. Схема алгоритма setNextQuestion().

**4. Тестирование программного продукта.**

Тестирование – это процесс выполнения программы, целью которого является выявление ошибок.

При тестировании данной программы в основном использовался метод черного ящика, оценочное тестирование и инспекция исходного текста.

Главной задачей, которой я руководствовалась, создавая данную программу, являлось удобство и понятность пользователю, который будет работать с этой системой, так как она должна выполнять все предписанные функции правильно.

**4.1 Инспекция исходного текста.**

Предлагается ответить на список вопросов для выявления исторически сложившихся общих ошибок программирования.

**1. Контроль обращений к данным**

* Все ли переменные инициализированы? - Все переменные в программе инициализированы.
* Не превышены ли максимальные (или реальные) размеры массивов и строк? – Нет.
* Не перепутаны ли строки со столбцами при работе с матрицами? – Нет работы с матрицами.
* Присутствуют ли переменные со сходными именами? – Нет, таких переменных нет.
* Используются ли файлы? Если да, то при вводе из файла проверяется ли завершение файла? – Да используются файлы формата txt. В случае, если файл отсутствую, он создается. При ввод из файла проверять на конец файл не требуется. В файле хранится пара ключ – значение, и если в файле нет такой пары, то туда записывается ключ со значением 0.
* Соответствуют ли типы записываемых и читаемых значений? – Типы читаемых и записанных значений совпадают.
* Использованы ли нетипизированные переменные, открытые массивы, динамическая память? – Да, особенность языка Java в том, что все переменные используют динамическую память. Нетипизированные переменные не используются. В качестве открытого массива используется класс ArrayList, контроль переполнения осуществляется классом.

**2. Контроль вычислений**

* Правильно ли записаны выражения (порядок следования операторов)? – Да выражения записаны правильно.
* Корректно ли выполнены вычисления с переменными различных типов? – Нет вычислений с переменными различных типов.
* Возможно ли переполнение разрядной сетки или ситуация машинного нуля? – Нет.
* Присутствуют ли сравнения переменных различных типов? – Нет.

**3. Контроль передачи управления**

* Будут ли корректно завершены циклы? - Да.
* Будет ли завершена программа? – Да.
* Существуют ли циклы, которые не будут выполняться из-за нарушения условия входа? – Да.
* Существуют ли поисковые циклы? Корректно ли отрабатываются ситуации «элемент найден» и «элемент не найден»? – Поисковых циклов нет.

**4. Контроль межмодульных интерфейсов**

* Соответствуют ли списки параметров и аргументов по порядку, типу, единицам измерения? – Да
* Не изменяет ли подпрограмма аргументов, которые не должны изменяться? – Таких аргументов нет.
* Не происходит ли нарушения области действия глобальных и локальных переменных с одинаковыми именами? – Нарушений не обнаружено.

**4.2 Тестирование методом черного ящика.**

Узким моментом в программе является игровой процесс. Возможные ошибки: вывод вопросов не по теме, вывод вопросов, количество которые не равно 10, не правильное фиксирование ответа пользователя.

Результат тестирования черным ящиком представлен в таблице 20.

Таблица 20.

|  |  |  |
| --- | --- | --- |
| Действия | Ожидаемый результат | Реакция системы |
| Пользователь выбирает тему игры | Пользователь переходит на экран первого вопроса на выбранной теме. | Пользователь переходит на экран первого вопроса на выбранной теме. |
| После выбора темы пользователю предлагается ответить на 10 вопросов | Пользователю предоставлено 10 вопросов | Пользователь может ответь на 10 вопросов |
| Пользователь отвечает на все вопросы правильно. | На экране результатов все ответы выделены зелёным | Не все вопросы выделены зелёным цветом |

При тестировании было выявлено, что при правильном ответе на все вопросы система не все ответы фиксирует правильно. После проверки кода выяснилось, что ошибка возникает в сравнении ответа пользователя с правильным ответом. Это происходит из-за, что при выводе на экран вариантов ответа, в ответе состоящем из 2 слов пробел заменятся на знак переноса на следующую строку(“\n”). Таким образом ошибки возникаю только при вопросе, ответ на который состоит из 2 слов. Для того чтобы исправить это надо перед сравнением заменить знак переноса на следующую строку знаком пробела.

**4.3 Оценочное тестирование.**

Цель оценочного тестирования является тестирование программы на соответствие основным требованиям.

Оценочное тестирование:

* Тестирование удобства использования.

Разработанная программа соответствует содержанию технического задания. Все предполагаемые функции были реализованы.

* Тестирование на предельных объемах

При большом объеме данных в базе программа корректно отрабатывает все

функции. Замечено, что при большом объеме данных отбор 10 вопросов для теста проходит быстрее.

* Тестирование на предельных нагрузках

Программы верно отображает информацию, поступившую в течении короткого времени, выводит ее быстро и без искажений.

* Тестирование удобства эксплуатации

Программа была предложена 3 пользователям. Они пользовались программой в течении 15 мин, были выделены следующие замечания:

1. При просмотре объекта звездного неба не заметны вкладки
2. После просмотра результатов игры пользователь попадал на экран с вопросом

Исправление замечаний:

1. Изменён цвет выделенной вкладки с лилового на бледно жёлтый
2. Теперь открытие экрана с вопросами производится с флагом FLAG\_ACTIVITY\_NO\_HISTORY, что позволяет не сохранять экран в стеке экранов.

* Тестирование конфигураций оборудования

Программа была проверена на смартфонах со следующими конфигурациями:

1. Qualcomm MSM 8255, 1000 МГц, 768 Мб
2. Qualcomm Snapdragon 400 MSM8626, 1200 МГц , 1 Гб
3. MediaTek MT6735, 1300 МГц , 2 Гб

На всех устройствах программа отработала без сбоев.

* Тестирование удобства установки

Для установки приложения требует, только запусти установочный файл с расширением apk.