Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

|  |  |
| --- | --- |
|  | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

Факультет Информатика и системы управления

Кафедра Компьютерные системы и сети (АК5)

Программа «энциклопедия звездного неба»

Расчетно-пояснительная записка

к курсовой работе

Листов 35

Студент гр. АК5-51 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** В. А. Лантратов

(Подпись, дата) (И.О. Фамилия)

Руководитель курсовой работы,

канд. техн. наук, доцент каф. ИУ-6 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Т.Н. Ничушкина

(Подпись, дата) (И.О. Фамилия)

Москва, 2015

**Реферат**

Записка 34 с., 3 ч., 14 рис., 9 табл., 3 источника, 2 прил.

ЗВЕЗДЫ, СОЗВЕЗДНИЯ, КОСМОС, ПЛАНЕТЫ, ГАЛАКТИКИ

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

Цель работы – проектирование программы, предоставляющей обучающую информацию по космическим объектам и возможность закрепления изученного материала.

Возможности программы:

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

Программой может использоваться любой владелец смартфона под управлением ОС Андроид, версия которой выше версии 2.3.3.

Оглавление

[Введение 4](#_Toc436030675)

[1. Анализ требований и уточнение спецификаций 6](#_Toc436030676)

[1.1. Выбор технологии, языка и среды разработки 6](#_Toc436030677)

[1.2. Построение диаграммы вариантов использования 7](#_Toc436030678)

[1.3. Выбор типа и формы диалога пользовательского интерфейса 10](#_Toc436030679)

[1.4. Выделение объектов предметной области 10](#_Toc436030680)

[1.5 Описание поведения. Системные события 11](#_Toc436030681)

[1.6 Построение диаграмм деятельностей 13](#_Toc436030682)

[2. Проектирование приложения 14](#_Toc436030683)

[2.1 Проектирование пользовательского интерфейса 14](#_Toc436030684)

[2.1.1. Разработка структуры меню. 15](#_Toc436030685)

[2.1.2. Построение графа состояний интерфейса. 15](#_Toc436030686)

[2.1.3 Проектирование игрового экрана 17](#_Toc436030687)

[2.2 Разработка классов и xml-файлов для реализации интерфейса приложения 17](#_Toc436030688)

[2.3 Проектирование структуры приложения 21](#_Toc436030689)

[2.3.1 Уточнение структуры программного обеспечения 21](#_Toc436030690)

[2.3.2 Проектирование классов предметной области 22](#_Toc436030691)

[2.3.3 Проектирование базы данных 24](#_Toc436030692)

[2.3.4 Проектирование классов работы с базой данных 25](#_Toc436030693)

[2.4 Разработка диаграмм компоновки системы 26](#_Toc436030694)

[3. Тестирование программного продукта. 29](#_Toc436030695)

[3.1 Выбор стратегии тестирования 29](#_Toc436030696)

[3.2 Разработка тестов 29](#_Toc436030697)

[3.2.1 Инспекция исходного текста. 29](#_Toc436030698)

[3.2.2 Функционально тестирование. Метод предположение об ошибке 31](#_Toc436030699)

[3.3 Оценочное тестирование. 32](#_Toc436030700)

[Заключение 34](#_Toc436030701)

[Литература 35](#_Toc436030702)

Приложение А. Техническое задание.

Приложение Б. Руководство пользователя.

Приложение В. Правила игры.

# Введение

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 их открытия.

Как известно, 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.

Изучая звездное небо, хочется ориентироваться в созвездиях: знать их названия, расположение и быстро находить их. Для этого надо освоить все созвездия. Однако, это требует немало времени, которого так не хватает в современном мире. В реальной жизни не каждый может позволить себе потратить десяток вечеров для изучения всего материала.

В настоящее время в интернете можно найти много программных продуктов посвященных изучению космоса. Есть и программы ориентированные на изучение звезд, созвездий и планет. Однако, 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. Кроме того, есть интернет ресурс Google Sky, но для использования нужен быстрый интернет.

Как показывает опыт, информация хорошо запоминается во время диалога. Поэтому была поставлена задача, разработать собственное удобное и быстрое приложение, которое не только будет давать информацию, но и позволит закрепить полученную информацию в игровой форме.

Разработанное приложение должно быть реализовано под ОС Андроид, так как это наиболее распространенная ОС, и позволит просматривать информацию о созвездиях, планетах и звездах в любом месте в любое время.

Разработанное приложение будет хранить информацию во встроенной в ОС базе данных. Таким образом, можно быстро получить доступ к данным по хранящимся там темам (созвездия, планеты и т.д.). Кроме того, пользователь будет иметь возможность провеять или закреплять свои знания по созвездиям и планетам.

# Анализ требований и уточнение спецификаций

Для разработки эффективного и технологичного продукта в первую очередь должны быть приняты принципиальные решения, которые во много определяют процесс проектирования. К основным принципиальным решениям относят:

* Выбор архитектуры программного продукта
* Выбор типа пользовательского интерфейса
* Выбор подхода к разработке
* Выбор языка и среды программирования

## 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 сформулированы следующие функциональные требования:

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

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

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

## Построение диаграммы вариантов использования

Для выполнения проектирования необходимо выявить внешних пользователей разрабатываемого программного обеспечения и перечень отдельных аспектов его поведения в процессе взаимодействия с конкретным пользователем. Внешним пользователем будет человек, установивший разрабатываемый программный продукт. В процессе анализа ТЗ и предметной области были выделены следующие варианты использования:

* Основные: «Просмотр данных об объектах», «Просмотр данных о категории», «Просмотр статистики», «Игра», «Игровой процесс»
* Дополнительные: «Добавление статистики», «Удаление статистики»

Опишем варианты использования «Просмотр данных об объекте» , «Просмотр данных о категории» в краткой форме. Описание представлено в таблицах 1 – 2.

Таблица 1. Вариант использования «Просмотр данных об объекте»

|  |  |
| --- | --- |
| Название варианта | **Просмотр данных об объекте** |
| 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. Вариант использования «Просмотр данных о категории»

|  |  |
| --- | --- |
| Название варианта | **Просмотр данных о 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 |

Ниже приведены варианты использования «Игра» и «Игровой процесс» подробно в таблицах 3 – 4:

Таблица 3. Вариант использование «**Игра**»

|  |  |
| --- | --- |
| Действия пользователя | Отклик системы |
| 1. Пользователь выбирает режим «Игра» | 1. Система выводит экран с игровым меню. |
| 1. Пользователь выбирает один из пунктов игрового меню. 2. Кнопка «Начать игру» см. вариант использования Игровой процесс. 3. Кнопка «Статистика» см. вариант использования «Добавление статистика» и «Удаление статистики». 4. Кнопка «Назад» | 4. Система переходит в соответствующий режим |
| 1. Пользователь нажимает кнопку «Назад» | 1. Система возвращает предыдущую форму |
|  |  |

Таблица 4. Вариант использования «**Игровой процесс**»

|  |  |
| --- | --- |
| Действия пользователя | Отклик системы |
| 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 1-7 можно построить общую диаграмму вариантов использования, что поможет наглядно представить ожидаемое поведение системы и уточнить варианты использования данной программы. Общая диаграмма вариантов использования представлена на рисунке 1.



Рисунок 1 – Диаграмма вариантов использование.

## 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 и функций ТЗ, был выбран интерфейс типа меню. Такая организация даст возможность пользователю быстро освоиться в программе, так как 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 отличие от директивной формы, в которой предполагается ввод команды для пользования приложением.

## 1.4. Выделение объектов предметной области

Анализ ТЗ и предметной области позволил выделить следующие понятия-сущности:

*Созвездие, планета,* *туманность, категория, чёрная дыра, статистика, астероиды, кометы галактики, вопрос, ответ, масса планеты, радиус планеты, длинна дня, описание созвездия, описание планеты и т.д.*

На роль объектов претендуют понятия Созвездие, Планета, Вопрос. Такие понятия как масса, длинна дня, описание планеты, радиус можно рассматривать как характеристики планеты т. е. атрибуты объектов Планета. А понятия описание созвездия можно рассматривать как атрибут объектов Созвездия. Понятия туманность, чёрная дыра, статистика, астероиды, кометы галактики однотипны, и их можно объединить в одно понятие категория, которое может быть описано объектом. Понятие вопрос также может быть объявлен объектом, а понятие ответ является атрибутом этого объекта.

В результате проведенного анализа, в качестве объектов предметной области были выделены:

* 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.

Все перечисленные объекты, кроме объекта вопрос, будут храниться в базе данных, структура которой будет уточняться в разделе разработки базы данных. Объект вопрос не хранится в базе так, как он является составным, и включает объект созвездие или планета и 4 варианта ответа.



Рисунок 2 – Диаграмма концептуальной модели предметной области

## 1.5 Описание поведения. Системные события

Для описания особенностей поведения разрабатываемого программного продукта, т. е. описание возможных действий системы, целесообразно использовать: диаграммы последовательностей системы, системные операции.

Диаграмма последовательностей системы позволяет для определённого сценария варианта использования показать генерируемые действующими лицами события и их порядок.

Анализ варианта использования «**Игровой процесс**» позволяет выделить 4 события, которые пользователь должен инициализировать. Диаграмма последовательностей системы варианта использования «**Игровой процесс**» представлена на рисунке 3.



Рисунок 3 – Диаграмма последовательностей системы варианта использования «**Игровой процесс**»

На основе предшествующий диаграммы последовательностей системы был сформирован список системных операций, приведённый на рисунке 4.



Рисунок 4 – Системные операции

Для уточнения системных операций следует выполнить подробное описание системных операций. Для этого необходимо провести анализ предметной области и ТЗ. В таблице 5 представлена уточненная системная операция «Выбрать ответ()».

Таблица 5. Уточненная системная операция «Выбрать ответ()»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | ***Выбрать 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 «**Игровой процесс»** необходимо провести анализ предметной области и последовательности системы варианта использования «**Игровой процесс**».

В результате анализа выяснилось, 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. Диаграмме деятельностей варианта использования «**Игровой процесс**» приведена на рисунке 5.



Рис 5 – Диаграмма деятельностей варианта использования «**Игровой процесс**»

# 2. Проектирование приложения

Разрабатываемый программный продукт будет интерактивным приложением, следовательно, целесообразно начать разработку с разработки интерфейса.

## 2.1 Проектирование пользовательского интерфейса

Так как был выбран интерфейс типа меню и табличная форма диалога, то стоит начать разработку интерфейса с разработки меню.

### 2.1.1. Разработка структуры меню.

Для обеспечения быстрой и удобной навигации пользователя по программе, перемещение по функциям целесообразно реализовать в виде иерархического меню. Анализ предметной области и диаграммы вариантов использования позволил разработать иерархию меню, структура которой представлена на рисунке 6.

****

Рисунок 6 – Структура меню.

### 2.1.2. 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 7.



Рисунок 7 – Граф состояний интерфейса.

Следующий этап проектирования интерфейса – это разработка форм ввода-вывода для каждого из состояний графа диалога.

### 2.1.3 Проектирование игрового экрана

Для реализации игрового режима необходимо разработать структуру экранной формы. Необходимо предусмотреть компоненты, реализующие диалог с пользователем.

На игровой форме пользователь должен имеет возможность ответить на вопрос, выбрав один из 4 вариантов ответа. Вопрос было решено представить в виде картинки, на котором изображен объект звездного неба. Таким образом, на экране должно быть расположено 4 кнопки. Одна кнопка с правильным ответом, и три кнопки с ложными ответами. Также следует предусмотреть на экране строку, фиксирующую количество заданных вопросов.

На рисунке 8 представлен макет игрового экрана.



Рисунок 8 – Вид макета игрового экрана

## 2.2 Разработка классов и xml-файлов для реализации интерфейса приложения

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

В соответствии с идеологией программирования для ОС Андроид, структура задается в файле с расширением xml. Ниже приведена упрощённая структура файла игрового экрана activity\_game.xml:

<LinearLayout>

<LinearLayout>

<TextView/>

<TextView/>

</LinearLayout>

<LinearLayout>

<ImageView/>

</LinearLayout>

<TableLayout>

<TableRow>

<Button/>

<Button/>

</TableRow>

<TableRow>

<Button/>

<Button/>

</TableRow>

</TableLayout>

</LinearLayout>

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 9 представлена диаграмма класса игрового интерфейса(GameActivity).



Рис 9 – Диаграмма класса GameActivity.

Как видно из диаграммы класса для реализации надо разработать методы, отвечающие за логику программы.

Наиболее интересным является метод onClick(), который ниже описан более подробно:

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

Схема алгоритма метода onClick() приведена на рисунке 10.



Рисунок 10 – Схема алгоритма onClick()

Остальные классы интерфейса были разработаны аналогичным образом.

## 2.3 Проектирование структуры приложения

Интерфейс определил основную структуру, но помимо интерфейса в предметной области были выделены объекты предметной области, а кроме того, приложение должно иметь возможность обращаться к базе данных.

### 2.3.1 Уточнение структуры программного обеспечения

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

* Пользовательский интерфейс – классы, реализующие объекты интерфейса.
* Библиотека интерфейсных компонентов – классы, реализующие интерфейсные компоненты кнопки, текстовые поля и т.п.
* Объекты предметной области – 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 группы классов можно объединить в пакеты. Полученные пакеты и их зависимости представлены на рисунке 11.



Рис 11 – Диаграмма пакетов.

### 2.3.2 Проектирование классов предметной области

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



Рис 12 - Диаграмма классов.

Все объекты предметной области имеют общие поля:

* name – имя объекта
* intId – идентификатор объекта

Поэтому технологично выделить все эти поля в отдельный класс MyObject, который будет общим для всех классов, реализующих объекты предметной области.

Класс ViewObject реализует класс «Категория». Этот класс наследуется от класса MyObject и расширяет его возможности путём добавления новых полей. Класс ViewObject нужен для работы с категориям, не имеющими 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 имеют 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 класса SkyObject и расширяет его необходимыми полями, необходимыми 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 классы 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.

### 2.3.3 Проектирование базы данных

Как было решено выше, данные по созвездиям, планетам, объекты звездного неба и список тем будут храниться в телефоне. Большие объемы можно хранить в файлах. Как показал анализ, хранить данные в файлах не целесообразно, так как в процессе работы приложения надо будет выполнять выборку данных. Если данные хранятся в файлах, то процесс выборки данных становится трудоемким. Поэтому было принято решения спроектировать базу данных.

Выбор данных из базы проводится быстрее, чем из файла. СУБД предоставляет удобный инструмент выборки данных и не требует дополнительных затрат времени на разработку алгоритма выборки.

В базе надо хранить данные по объектам:

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

Для каждого объекта следует разработать 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.

На рисунке 13 представлена даталогическая модель данных.



Рисунок 13 – Даталогическая модель данных

### 2.3.4 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 необходимо разработать два дополнительных класса: DBHelper и SkyDataBaseImpl.

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, который выполняет основную работу с базой данных.

Для класса SkyDataBaseImpl реализован интерфейс SkyDataBase.

На рисунке 14 представлена диаграмма описанных выше классов.

****

Рисунок 14 – Диаграмма классов реализующие работу с БД.

## 2.4 Разработка диаграмм компоновки системы

В соответствии с идеологией разработки для ОС Андроид, для каждого класса создается отдельный модуль. Модули всех классов в процессе компоновки объединяются в единый исполняемый файл приложения. Диаграмма компоновки разработанного приложения приведена на рисунке 14. Скомпилированные классы интерфейсов (компонент 4) зависят от классов интерфейсов (компонент 3) и от скомпилированных классов определяющих логику интерфейса (компонент 2). Классы интерфейсов зависят от классов определяющих логику работы интерфейса (компонент 1) , от xml файлов определяющих структуру экрана(компонент 5) и от изображений(компонент 6). Xml файлы, определяющие структуру экрана, зависят от изображений, xml файлов, задающие стили элементам экрана (компонент 7) и от файлов ресурсов(компонент 8).

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

Перечень классов: ConstellationObject.java, MyObject.java, PlanetObject.java, QuestionObject.java, SkyObject.java, ViewObject.java, DBHelper.java, SkyDataBase.java, SkyDataBaseImpl.java

2 – скомпилированные классы, 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 классы описанные в пункте 3.

5 – xml файлы задающие структуру экрана.

Перечень файлов: 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



Рисунок 15 – Диаграмма компоновки.

# 3. 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.

## 3.1 Выбор стратегии тестирования

Для тестирования разработанной программы было выбрано две стратегии тестирования, а именно:

1. Ручной контроль программного обеспечения метод инспекция исходного текста.
2. Функциональное тестирование метод предположение об ошибке

Ручной контроль был выбран, так как доказано, что с помощью него можно находить от 30 до 70% ошибок логического проектирования и кодирования. Был выбран метод инспекции исходного кода, поскольку это позволит выявлять ошибки логического проектирования и самые распространенные ошибки кодирования на всех этапах разработки.

Кроме того было выбрано функциональное тестирования методом предположения об ошибки, так как этот метод позволит выявить возможные ошибки в наиболее сложных моментах программы.

## 3.2 Разработка тестов

### 3.2.1 Инспекция исходного текста.

В соответствие с этой стратегией, инспектирующему 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? – Да
* 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 вводе из файла проверять файл на достижение конца 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.

В процессе инспекции исходного текста были обнаружены и устранены ошибки, а именно в функции onClick() происходило перекрытие переменной intent.

### 3.2.2 Функционально тестирование. Метод предположение об ошибке

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 9.

Таблица 6. Результаты тестирования методом черного ящика.

|  |  |  |
| --- | --- | --- |
| 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 происходило из-за того, 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 надо перед сравнением заменить знак переноса на следующую строку знаком пробела.

В ходе тестирования были выявлены и устранены ошибки программирования.

## 3.3 Оценочное тестирование.

Целью оценочного тестирования является тестирование программы на соответствие основным требованиям.

Из предлагаемых были выбраны следующие виды оценочного тестирования, так как эти вилы тестирования позволяют оценить на сколько пользователю будет комфортно пользоваться приложением.

* 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.

# Заключение

В результате выполнения курсовой работы была разработана программа «Энциклопедия звездного неба». Программа позволяет получить информацию по объектам звездного неба и закрепить полученную информацию в игровой форме.

Ограниченное время на разработку приложения не позволило реализовать больше тем для игры и обновление базы через интернет.

В перспективе планируется сделать:

* Больше тем для игрового режима
* Обновление базы через интернет
* Улучшить дизайн приложения

# Литература

1. Иванова Г.С. Технология программирования. М. : Изд-во КНОРУС, 2013.

2. Иванова Г.С., Ничушкина Т. Н., Пугачев Е.К., Самарев Р.С. Методические указания по выполнению курсовой работы по дисциплине «ТРПС», 2013.

3. Иванова Г.С. Основы программирования: Учебник для втузов. – 4-е изд. М.: Изд-во МГТУ им. Н.Э. Баумана, 2007.

4. Иванова Г.С., Ничушкина Т.Н. Объектно – ориентированное программирование: учебник. М.: Из-во МГТУ им. Н. Э. Баумана, 2014. 455 с.