**Муниципальное автономное общеобразовательное учреждение**

**«САМАРСКИЙ МЕДИКО-ТЕХНИЧЕСКИЙ ЛИЦЕЙ»**

**городского округа Самара**

**Конференция учащихся 8-9 классов**



**Создание будильника с математическами задачами, с использованием языка программирования Java**

|  |  |  |  |
| --- | --- | --- | --- |
| Автор: | Захаров Арсений Алексеевич, | | |
|  | учащийся | | 9 класса |
| Руководитель: | Бурдин А. С., | | |
|  | учитель | информатики | |

**Самара, 2023**

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 3

ГЛАВА 1. 4

§1. Основы JAva 4

§2. принципы ООП. 4

§3. РАЗработка приложений на android studio. 4

ГЛАВА 2. 5

§1 планирование функций будильника. Постройка uml диаграммы (диаграмма классов). Постройка схемы базы данных MYSQL (sequel Secure English query Language). 5

§2. РАзработка android приложения. 5

§3. заключительная часть проекта. создание минимального и понятного для пользователя дизайна. Тестировка и отладка. РАЗВЕРТЫВАНИЕ ПРИЛОЖЕНИЯ и выгрузка его в интернет. 5

ЗАКЛЮЧЕНИЕ 6

СПИСОК ИНФОРМАЦИОННЫХ РЕСУРСОВ 7

ПРИЛОЖЕНИЯ 8

**ВВЕДЕНИЕ**

Для многих школьников России, к достижению 15 - 18 годам, приходится сдавать экзамены, такие как ОГЭ и ЕГЭ. Кроме того, им довольно тяжело вставать рано по утрам. Мой проект направлен на создание инновационного будильника, который требует от пользователей решения ряда математических задач, чтобы отключить будильник. Это не только обеспечит эффективный способ для людей, которым трудно просыпаться вовремя, но и добавит забавный вызов в их утреннюю рутину.

Целью моего проекта является разработка этого продукта на языке программирования Java, что сделает его доступным и простым в использовании для всех типов пользователей.

Мои проектные задачи:

1. Изучение языка программирования Java и ее использование в Android Studio
2. Разработка пользовательского интерфейса и функций будильника.
3. Разработка кода бэкенда, необходимого для функционирования будильника.
4. Внедрение алгоритмов математических задач в код.
5. Написание кода, необходимого для того, чтобы сделать будильник интерактивным и удобным для пользователя.
6. Тестирование и отладка приложения, чтобы убедиться, что оно работает правильно.
7. Развертывание приложения будильника.

Создание будильника с математическими задачами с помощью программирования на Java представляет собой актуальную задачу для многих программистов. Такие приложения могут быть использованы для привлечения внимания к учебе, развития математических навыков и стимулирования самообучения. Для реализации данной идеи потребуется знание Java программирования, а также проектные навыки, чтобы определить задачи, необходимые для реализации этой идеи.

**ГЛАВА 1.**

Теоретическая часть создания будильника с математическими задачами с помощью программирования на Java включает в себя понимание основ языка, таких как структуры данных, классы и объекты, наследование, механизмы потока управления и обработка исключений. Кроме того, важно понять, как реализовать на Java графический интерфейс для взаимодействия с пользователем. После понимания этих понятий можно приступить к написанию кода, использующего математические алгоритмы для создания будильника, который требует от пользователя решить математическую задачу, прежде чем получить доступ.

**§1.Основы Java**

Java - один из самых популярных языков программирования в настоящее время. Он позволяет создавать самые разнообразные приложения и инструменты, которые можно использовать в различных областях. Поэтому особенно важно мне стало понимать основы синтаксиса Java и принципы программирования на этом языке. Синтаксис Java - это набор правил, которые необходимо соблюдать при программировании на языке Java. Эти правила включают в себя лексику и построение программы, в том числе слова, предложения, объявления и определения переменных. Все это должно быть достаточно простым, чтобы другие программисты могли легко использовать ваш код. Принципы программирования на Java также очень важны. Они включают такие понятия, как внутренние и внешние пространства имен, ООП, абстрактные классы и методы, многопоточность, исключения и многое другое. Все это помогаетписать эффективный и безопасный код. В общем, я начал изучать Java с синтаксиса.

Основные конструкции:

1. Циклы(for и while)

Чтобы создать цикл в программе Java, можно использовать конструкцию цикла, такую как цикл for, цикл while. Эти циклы позволяют выполнять код несколько раз подряд и могут быть использованы для создания циклов в логике вашей программы. Например, следующий код создает цикл, который распечатывает числа от 1 до 10:

for (int i = 1; i <= 10; i++) {

System.out.println(i);

}

В этом примере цикл будет выполняться 10 раз, и каждый раз будет выводиться текущее значение переменной счетчика (i). Это создает цикл, потому что цикл всегда будет начинаться с начала после достижения конца.

В Java цикл while - это оператор потока управления, который позволяет повторять выполнение кода на основе заданного булева условия. Цикл будет выполняться до тех пор, пока условие не станет ложным. Вот пример простого цикла while в Java:

while (condition) {

// Код для выполнения

}

В этом примере условие оценивается перед каждой итерацией цикла. Если условие оценивается как true, выполняется код внутри цикла. Когда условие становится ложным, цикл завершается, и программа продолжает выполнение кода, следующего за циклом while.

2. Условия(if, switch-case)

В Java существует несколько типов условных конструкций, которые можно использовать для управления ходом программы. К ним относятся оператор if, оператор if-else, оператор switch, вложенные операторы if.

Оператор if используется для проверки истинности условия и последующего выполнения определенного набора инструкций. Например, следующий код выполнит код внутри оператора if, только если условие (x == 5) истинно:

if (x == 5) {

// Code to be executed

}

Оператор if-else похож на оператор if, но он позволяет указать различные наборы инструкций для выполнения в зависимости от того, истинно или ложно условие. Например, следующий код выполнит код внутри оператора if, если условие (x == 5) истинно, или код внутри оператора else, если оно ложно:

if (x == 5) {

// Код, который будет выполнен, если x == 5

} else {

// Код, который будет выполнен, если x != 5

}

Оператор switch - это более лаконичный способ написания нескольких операторов if-else, который полезен для проверки нескольких условий одновременно. Например, следующий код будет выполнять различные наборы инструкций в зависимости от значения переменной x:

switch (x) {

case 1:

// Код, который будет выполнен, если x == 1

break;

case 2:

// Код будет выполнен, если x == 2

break;

default:

// Код, который будет выполнен, если x не 1 или 2

break;

}

Вложенные операторы if позволяют проверять несколько условий одновременно путем вложения нескольких операторов if друг в друга. Например, в следующем коде перед выполнением кода внутри оператора if проверяется, равна ли переменная x 1, а переменная y равна 2:

if (x == 1) {

if (y == 2) {

// Код, который будет выполнен, если x == 1 и y == 2

}

}

3. Массивы и списки

В Java массив и список ArrayList являются хорошо известными структурами данных. Массив - это базовая функциональность, предоставляемая Java, тогда как ArrayList - это класс Java Collection Framework, реализованный как изменяемый по размеру массив. Массивы хранят данные линейно и могут быть доступны по номеру индекса, в то время как ArrayList хранит объекты в динамическом списке и не имеет фиксированного размера. С точки зрения производительности, операции с массивами обычно быстрее, чем операции со списками, но для операций, занимающих несколько наносекунд, разница несущественна. Кроме того, операции над массивами ограничены операциями набора, в то время как интерфейс списка предоставляет дополнительные операции, такие как добавление, удаление и поиск.

4. Подпрограммы(функции, процедуры)

В Java функции и процедуры называются подпрограммами или методами. Метод - это блок кода, который связан с классом или объектом и используется для выполнения определенной задачи. Методы определяются с помощью возвращаемого типа, имени и параметров. Например, вот простой метод, который принимает два параметра, умножает их вместе и возвращает результат:

public int multiply(int x, int y) {

return x \* y;

}

Методы также могут не принимать никаких параметров и не возвращать никакого значения, или же они могут возвращать объект или массив. Методы также могут быть определены с ключевым словом void, которое сообщает компилятору, что метод не возвращает значение.

Java также поддерживает концепцию рекурсивных методов, которые представляют собой методы, вызывающие сами себя. Рекурсия часто используется для решения сложных задач, которые можно разбить на более мелкие.

**§2.Принципы ООП**

Четыре основных принципа объектно-ориентированного программирования (ООП) - это инкапсуляция, абстракция, наследование и полиморфизм.

Инкапсуляция - это процесс объединения данных и функций в единый блок, или класс. Это полезно для хранения связанного кода и данных в одном месте.

Абстракция - это процесс сокрытия внутренних деталей класса от внешнего мира. Это позволяет сделать код более гибким и простым в обслуживании.

Наследование - это процесс создания нового класса на основе существующего. Новый класс наследует все функции существующего класса, а также может добавлять свои собственные.

Полиморфизм - это способность класса принимать различные формы. Это позволяет одинаково обращаться с объектами разных классов.

Агрегация и композиция - это два типа ассоциативных отношений, которые существуют между двумя классами. При агрегации один класс (родительский) имеет ссылку на другой класс (дочерний), но дочерний может существовать независимо от родительского. Агрегация также известна как отношение "has-a". В композиции родитель и ребенок имеют более сильные отношения, и ребенок не существует независимо от родителя. Композиция также известна как отношение "part-of". И агрегация, и композиция позволяют дочернему объекту наследовать свойства от родителя и получать доступ к его методам. При композиции методы родителя могут быть переопределены в дочернем классе, а родительский объект может быть уничтожен без ущерба для дочернего объекта. Агрегация и композиция являются важными понятиями в объектно-ориентированном программировании и используются для представления отношений между объектами.



**§3. Разработка приложений в Android Studio**

Android Studio - это мощная среда разработки для создания приложений на платформе Android. Она предоставляет интуитивно понятный пользовательский интерфейс и комплексные инструменты, позволяющие разработчикам быстро создавать, тестировать, отлаживать и внедрять свои приложения. Благодаря богатому набору функций, таких как перетаскивание элементов дизайна, поддержка виртуальных устройств для тестирования нескольких версий ОС Android и устройств с различными размерами экрана, интегрированные системы контроля версий, такие как Git и Subversion, которые помогают эффективно управлять исходным кодом, разработка приложений в Android Studio превращается в легкое дело.

Приложения Android имеют жизненный цикл, который определяет, как они создаются, запускаются, возобновляются, приостанавливаются, останавливаются и уничтожаются. Жизненный цикл активности состоит из семи методов обратного вызова, которые выполняются в следующем порядке: onCreate(), onStart(), onResume(), onPause(), onStop(), onRestart() и onDestroy(). Метод onCreate() является обязательным и вызывается при первом запуске приложения. Метод onStart() вызывается, когда активность становится видимой, но еще не находится на переднем плане. Метод onResume() вызывается, когда активность находится на переднем плане и пользователь может взаимодействовать с ней. Метод onPause() вызывается, когда активность больше не находится на переднем плане и пользователь больше не взаимодействует с ней. Метод onStop() вызывается, когда активность больше не видна. Метод onRestart() вызывается, когда активность была остановлена и перезапускается. Наконец, метод onDestroy() вызывается, когда активность уничтожается и больше не используется. Понимая жизненный цикл активности, разработчики могут лучше управлять своими приложениями и обеспечивать их бесперебойную работу.



**ГЛАВА 2.**

**§1. Планирование функций будильника. Постройка UML диаграммы (диаграмма классов). Постройка схемы Базы Данных MySQL (SEQUEL - Secure English QUEry Language).**

При написании будильника для приложения Android мне было важно спланировать технические аспекты проекта. Это включает UML диаграмму классов, дерево активностей, схема база данных, кнопки, настройки и другие функции. Для платформы Android я как следует спланировал, как использовать Android SDK, а также какие API я буду использовать для доступа к системным функциям, таким как будильники и уведомления. Другие технические соображения включают выбор подходящего языка программирования для разработки, а также выбор архитектуры и дизайна программного обеспечения. Наконец, мне необходимо было запланировать тестирование и отладку будильника, чтобы убедиться, что он работает так, как ожидается.

Из функций будильника, я выделил две самых важных: звонить в определенное время и запускать активность с примерами, чтобы потом отрубить будильник.

Для начала я решил нарисовать UML диаграмму для запланированных мною классов.



UML диаграмма запланированного проекта

Дальше я приступил к разработки схемы базы данных и вот, что получилось:



На этом мой этап планирования окончился и я перешел к разработке моего продукта - будильника.

**§2. Разработка Android приложения.**

Для начала я создал MainActivity в AndroidStudio. Первым интересующим меня вопросом, как я стал бы реализовывать будильник, пока что без математических примеров.

Первое, что я начал делать - это искать дизайн для будильника. Я решил использовать Material.io.

Material.io - это интерактивный сайт, поддерживаемый компанией Google, который предоставляет рекомендации по дизайну, инструменты и ресурсы, помогающие разработчикам создавать последовательный и привлекательный пользовательский опыт в своих цифровых продуктах. Сайт включает библиотеку элементов дизайна, таких как цвета, типографика, иконки и компоненты, а также визуальные ресурсы, такие как галереи вдохновения и учебники. Он также предлагает набор инструментов для создания прототипов, создания и тестирования продуктов, а также платформу для совместной работы с командами. Material.io предназначен для создания интуитивно понятных, доступных и отзывчивых пользовательских интерфейсов.

Для эксплуатации я выбрал MaterialTimePicker и объявил его в лямбда-функции (анонимной функции), которая вызывается после нажатия на кнопку.

Чтобы установить время, в которое мой будильник будет звонить я воспользовался классом Calendar.

Класс Calendar в Java - это абстрактный класс, который предоставляет методы для преобразования даты между определенным моментом времени и набором календарных полей, таких как YEAR, MONTH, DAYOFMONTH, HOUR и так далее. Он также предоставляет дополнительные поля и методы для реализации конкретной календарной системы вне пакета. Класс Calendar включает методы для преобразования объекта Date в набор полей календаря и наоборот, а также методы для выполнения арифметики дат. Он также содержит ряд вспомогательных методов для сравнения дат, вычисления количества дней в месяце и других задач.

Чтобы будильник, звонил, я использовал AlarmManager. Класс AlarmManager в Java является частью системной службы Android, которая позволяет планировать и выполнять задачи через заданные интервалы времени. Он предоставляет API для установки сигналов тревоги, которые могут пробудить устройство от сна и запустить широковещательный приемник или службу в определенное время. Будильники основаны на истекшем времени и могут быть настроены на повторение через определенный интервал. AlarmManager также можно использовать для планирования одноразовых или периодических задач, которые должны выполняться в определенное время. Кроме того, AlarmManager можно использовать для планирования любого типа ожидающих намерений, включая широковещательные намерения, намерения активности и намерения службы.

А также прописал, всплытие принятия разрешение на показ "Поверх других окон" и сделал PendingIntent, для того, чтобы, когда пользователь вырубил приложение моего будильника. Его поток продолжал работать без засыпаний, как и вся система.

Я создал вторую активность: AlarmActivity. Я в нем сделал так, чтобы звонил звонок, через ringtone. И написал с небольшим, как по мне хардкодингом логику для получения примера с ответом и его сравнением в данной активности. Там я выполнил рандомный выбор примера из бд, при помощи индексов, которые сравнивались с настройками. Сами же настройки я хранил в SharedPreferences в формате json.

Теперь я создал активность SettingsAlarmClock. В нем я реализовал, отключение и включение примеров, по CheckBox. Потом их состояние я сохранил в SharedPreferences, по нажатии на кнопку "Save".

**§3. Заключительная часть проекта. Создание минимального и понятного для пользователя дизайна. Тестировка и отладка. Развертывание приложения и выгрузка его в интернет.**

Любое мобильное приложение не может быть без названия и иконки. Мое приложение не стало исключением. Я решил его назвать ***"MAlarm"***. А иконку я сделал при помощи Искуственного Интелекта DALL·E 2 от компании Илона Маска OpenAI (позже на основе его API Сбербанк создаст свою нейросеть RuDalle).

Вот, иконка сгенерированная нейросетью.



Дальше в файлах XML, я сделал небольшой дизайн. Выровнял кнопки, лэйбелы. Также написал дополнительных пояснений для удобной эксплуатации, моего приложения.

Следующая часть, была подготовка к развертке приложения. Я сделал бета-версии для моих друзей. Они мне писали баги, которые находили, и я проводил отладку. В итоге баги я полностью пофиксил. Я создал apk файл с итоговым приложением. И выложил его на Mega.

Вот ссылка на финальное приложение: <https://clck.ru/33dg6S>.

**ВЫВОД**

Во время создания будильник с математическими задачами с помощью языка программирования Java, я пытался добиться, того чтобы программа была организована, эффективна и проста для понимания. Кроме того, программа использовала существующие встроенные библиотеки, такие как Calendar и AlarmManager, чтобы упростить процесс кодирования. Также я позаботился об создании аварийных сигналов, которые были правильно запланированы и выполнялись вовремя, и использовали соответствующие отложенные намерения для различных типов задач. Наконец, я и мои друзья тщательно протестировали программу, чтобы я смог убедиться, что пользовательский дизайн удовлетворителен, и сигналы тревоги работают правильно.

**ЗАКЛЮЧЕНИЕ**

Создание будильника с математическими задачами с использованием языка программирования Java стал отличным способоб сделать пробуждение по утрам более приятным и интересным. С помощью этого типа будильника пользователи могут выбирать различные уровни сложности при установке будильника, что делает его подходящим для всех возрастов. Кроме того, этот тип будильника предлагает уникальный поворот к традиционным методам, включив в свой дизайн математику. Это побуждает людей мыслить критически, пока они еще не проснулись!

Java стал отличным выбором для создания такого приложения благодаря своей надежности и гибкости как языка программирования. Он позволил мне быстро создавать программное обеспечение со сложной функциональностью без ущерба для производительности и стабильности. Кроме того, Java предоставляет множество библиотек, которые позволяли мне получить доступ к дополнительным функциям, таким как элементы графического интерфейса, которые в противном случае были бы сложными или трудоемкими задачами, если бы выполнялись вручную с нуля в самой кодовой базе.

В целом, создание будильника на основе алгебры с использованием Java дало много преимуществ по сравнению разработки данного приложения на других языках. Он помог реализовать, многие сложные задачи, несколькими строчками кода. У меня также был некий опыт с объектно-ориентированными языками, такими как java, поэтому разработка такого приложения не составила мне особых сложностей в разработке.

**СПИСОК ИНФОРМАЦИОННЫХ РЕСУРСОВ**

* Thinking in Java, 4th Edition by Bruce Eckel
* Java in a Nutshell, 7th Edition by Benjamin J Evans
* Core Java Volume I—Fundamentals, 11th Edition by Cay S. Horstmann
* Java: The Complete Reference, 10th Edition by Herbert Schildt
* Java Programming on Android: An Introduction to Building Apps with Android Studio by John Horton
* <https://habr.com/>
* <https://github.com/MCtop4ik>

**ПРИЛОЖЕНИЯ**

**Ссылка на мой github:** [***https://github.com/MCtop4ik/Alarmc***](https://github.com/MCtop4ik/Alarmc)