МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

КАЛИНИНГРАДСКОЙ ОБЛАСТИ

ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ

«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СТРОИТЕЛЬСТВА»

(ГБУ КО ПОО «КИТиС»)

|  |  |  |
| --- | --- | --- |
| Курсовой проект  допущен к защите  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата)  Зам.директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_\_Павленко Г.Я.  (подпись) |  | Курсовой проект  защищен с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  Руководитель проект  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Пояснительная записка к курсовому проекту

по дисциплине: МДК 01.01 Разработка программных модулей

Тема: Разработка приложения «Таймер с функцией будильника» на смарт -часы

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |
| --- | --- |
| Выполнил: | студент 3 курса,  группы ИСп19-2к  Дегтярев Владимир Сергеевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Руководитель: | Большакова-Стрекалова Анна Викторовна, преподаватель ГБУ КО ПОО КИТиС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Калининград

2022

Содержание

[Введение 3](#_Toc101996388)

[1 Описание предметной области 4](#_Toc101996389)

[1.1 Аналоги разрабатываемого приложения 5](#_Toc101996398)

[1.2 Техническое задание 7](#_Toc101996399)

[1.3 Описание структуры приложения 8](#_Toc101996400)

[2 Описание разработки приложения 9](#_Toc101996401)

[2.1 Обоснование средств разработки 9](#_Toc101996402)

[2.2 Разработка интерфейса 10](#_Toc101996404)

[2.3 Разработка логики работы приложения. Схема взаимодействия компонентов проекта. 11](#_Toc101996409)

[2.4 Описание переменных, компонентов, классов и подпрограмм 12](#_Toc101996410)

[3 Тестирование и установка приложений 20](#_Toc101996413)

[Заключение 21](#_Toc101996414)

[Список используемой литературы 22](#_Toc101996415)

[Приложение А. Листинг рабочего кода 23](#_Toc101996416)

# Введение

Для будущего любого предприятия разработка мобильных или веб-сервисов имеет решающее значение, поскольку создание мобильного приложения для компании - это способ повысить интерес аудитории и продвинуть бизнес.

Большинство смарт-часов, начинали использоваться «в связке» со смартфоном. Однако все меняется очень быстро. В настоящее время доступны несколько автономных умных часов, которым не нужно сопряжение со смартфоном. Последние нововведения в значительной степени улучшили и усовершенствовали.

Требуется разработать программу «Таймер с функцией будильника». Заданы ряд требований по выполнению работы. Программа будет написана на языке Java*.* Java обеспечивает быстроту работы, легкость компиляции, совместимость, интеграционную способность и т.д.

Целью исследования является процесс разработки приложения.

Для достижения поставленной цели необходимо решить ряд задач:

* рассмотреть роль современных приложений в процессе коммуникации с целевой аудиторией;
* исследовать основные организационные, экономические и маркетинговые особенности разработки приложения;
* дать обоснование созданию приложения и изучить этапы его создания

Объектом исследования в данной работе являются приложения для мобильных устройств и смарт-часов.

Предметом исследования является комплекс мероприятий по созданию и продвижению приложения для увеличения количества его потенциальных посетителей.

1 Описание предметной области

Java является основой практически для всех типов сетевых приложений и всеобщим стандартом для разработки и распространения встроенных и мобильных приложений, игр, веб-контента и корпоративного программного обеспечения. В мире насчитывается более 9 миллионов специалистов, разрабатывающих приложения на Java, которая позволяет эффективно разрабатывать, внедрять и использовать превосходные приложения и услуги.

Технология Java протестирована, усовершенствована, расширена и проверена участниками сообщества разработчиков Java, архитекторов и энтузиастов. Java позволяет разрабатывать высокопроизводительные портативные приложения практически на всех компьютерных платформах. Доступность приложений в разнородных средах позволяет компаниям предоставлять более широкий спектр услуг, способствует повышению производительности, уровня взаимодействия и совместной работы конечных пользователей и существенному снижению стоимости совместного владения корпоративными и потребительскими приложениями. Java стала незаменимым инструментом для разработчиков и открыла для них следующие возможности:

* написание программного обеспечения на одной платформе и его запуск практически на любой другой платформе;
* создание программ, работающих в веб-браузере и имеющих доступ к веб-службам;
* разработка приложений на стороне сервера для форумов в Интернете, магазинов, опросов, обработки форм HTML и много другого;
* объединение приложений или служб с использованием языка Java для создания высокоспециализированных приложений или служб;
* создание многофункциональных и эффективных приложений для мобильных телефонов, удаленных процессоров, микроконтроллеров, беспроводных модулей, датчиков, шлюзов, потребительских продуктов и практически любых других категорий электронных устройств.

Кроме стандартной редакции набора инструментальных средств для разработки программного обеспечения на Java - StandardEdition -, были предложены еще два варианта: MicroEdition (“микроредакция”) для портативных устройств, например для мобильных телефонов, и EnterpriseEdition (редакция для корпоративных приложений).

1.1 Аналоги разрабатываемого приложения

В данном курсовом проекте я сравнил несколько подобных приложений для анализа предстоящих достоинств и недостатков.

Первое приложение «Goodtime» в стиле Pomodoro.



Рисунок 1 – Иконка приложение Goodtime

Принцип Goodtime таков: вы запускаете отсчёт времени (по умолчанию период длится 25 минут) и работаете не отрываясь. Когда заданный период истекает, вы можете вознаградить себя пятиминутным отдыхом.

Таблица 1 – Плюсы и минусы приложения Goodtime

|  |  |
| --- | --- |
| Плюсы | Минусы |
| Интерфейс минималистичный | Ограниченное время таймера (25 минут) |
| Управление жестами | Может сбрасывать некоторые настройки |
| Функция отключение уведомлений |  |

Следующее приложение «Timer Plus». Приложение рассчитано на любителей спорта.

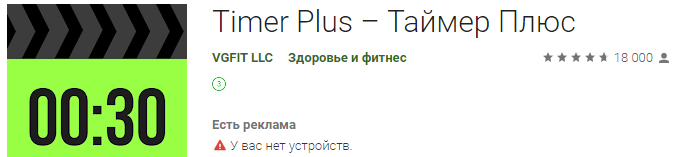


Рисунок 2 – Иконка приложения Timer Plus

Timer Plus позволяет эффективно чередовать тренировки и отдых. В приложении имеется табата-таймер, таймер раундов для кроссфита и секундомер для бегунов. Кроме того, ничего не мешает вам создавать собственные счётчики времени, если вы придерживаетесь нестандартной программы тренировок.

Таблица 2 - Плюсы и минусы приложения Timer Plus

|  |  |
| --- | --- |
| Плюсы | Минусы |
| Очень яркий, контрастный интерфейс | Перестает работать при попытке войти в настройки |
| Присутствуют опции, не позволяя брать в руки смартфон | Не работает в фоновом режиме |
| Предупреждение об окончании очередного раунда | Иногда не запускается на других девайсов |

Далее приложение, нацеленное на повышение продуктивности, «Brain Focus».

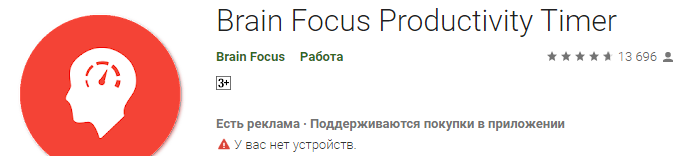


Рисунок 3 – Иконка приложения Brain Focus

Brain Focus помогает эффективно чередовать периоды работы и отдыха, а также ведёт подробную статистику — сколько времени вы потратили на трудовую деятельность, сколько отдыхали и сколько прогуливали.

Таблица 3 - Плюсы и минусы приложения Brain Focus

|  |  |
| --- | --- |
| Плюсы | Минусы |
| Возможность создавать задачи, | Не хватает фиксации конкретного времени |
| Присутствует режим «Не беспокоить» в настройках. | Отсутствие синхронизации между устройствами |
| Блокировка программ, чтоб  возникло соблазна посидеть в социальных сетях | При повороте экрана на смартфоне пропадает текущее время |

1.2 Техническое задание

Программа «Таймер с функцией будильника», которую необходимо написать в данной работе, предназначена для отображения отчёта времени в цифровой.

В программу планируется добавить доступ к системному времени. Так же в дальнейшем будет добавлена возможность выбора звукового сигнала для таймера и будильника.

Все функции установки будильника и таймера достаточно просты и удобны. Часы должны иметь удобную форму и красивый фон. При необходимости их можно переместить по экрану. Они всегда будут на рабочем столе, и пользователь всегда будет знать время.

1.3 Описание структуры приложения

Ниже представлена блок-схема алгоритма.

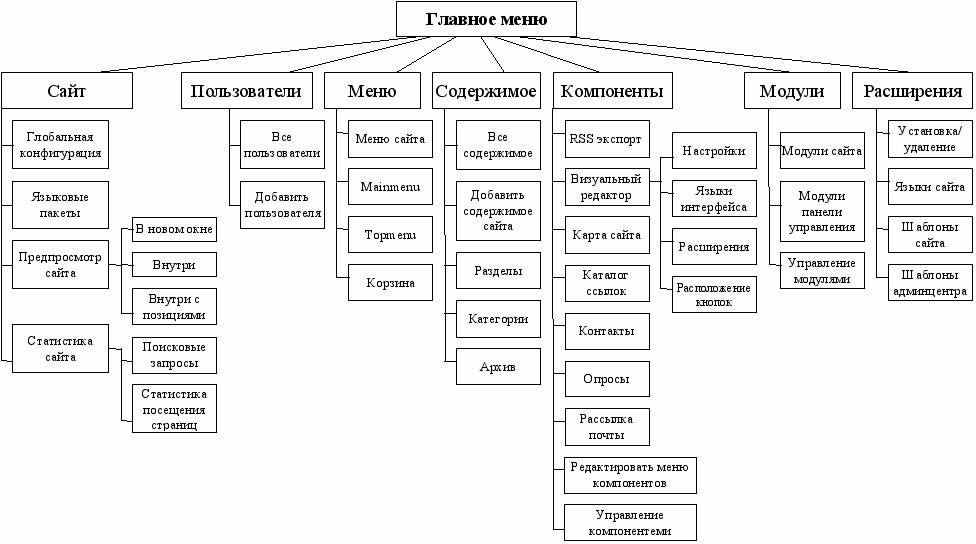


Рисунок 4 - Блок-схема алгоритма работы органайзера

Блок-схема объясняет полный алгоритм работы программы. При запуске программы появляется диалоговое окно, в котором нам предлагается выбрать мелодию, которая будет воспроизводиться в будильнике и таймере. Одновременно с запуском программы запускается секундомер, который покажет, как долго мы используем эту программу. Параметры задаются вручную. Чтобы подтвердить запуск, нажимаем кнопку "Установить". Также существует опция сброса.

2 Описание разработки приложения

В данной программе реализовано отображение времени в цифровой форме, имеющим точность 10 знаков, а также присутствует дизайнерская проработка.

Рассмотрим более детально реализованные в программе функции. Главная функция программы - отображение времени. Цифровая форма более удобна, так как отображается точное время в формате чч - мм - сс.

Для получения времени используется системная функция Time, а для отображения ее на экране функция Timetostr.

При перемещении часов изменяется задний фон, а также цвет отображения времени.

Возможность установить будильник на нужное время реализована с помощью стандартного компонента изменения даты и времени.

Возможность активации будильника представлена в цифровой форме. Используются кнопки типа Button, для включения будильника и установки времени, для изменения времени звонка.

Каждый элемент задания (часы, таймер) располагаются на отдельной форме. Звуковой сигнал воспроизводится с общего музыкального проигрывателя.

2.1 Обоснование средств разработки

Как правило, современные средства создания Java-приложений поддерживают разработку приложений для нескольких различных платформ, а зачастую и сами существуют в версиях для различных платформ. Обычно в их число входят Windows, Linux, Solaris, а иногда и другие операционные системы.

Среда Java - это сложный механизм, обеспечивающий высокоэффективную работу программиста. Визуально она реализуется несколькими одновременно раскрытыми на экране окнами. Окна могут перемещаться по экрану, частично или полностью перекрывая друг друга, что обычно вызывает у пользователя, привыкшего к относительной «строгости» среды текстового процессора Word или табличного процессора Excel, ощущение некоторого дискомфорта.

2.2 Разработка интерфейса

Основные составные части Java

* Дизайнер форм.
* Окно редактора исходного текста.
* Палитра компонентов.
* Инспектор объектов.

Дизайнер форм в Java прост в использовании, поэтому создать визуальный интерфейс очень просто. Дизайнер форм первоначально состоит из одного пустого окна, которое заполняется разными объектами, расположенными на Палитре компонентов. Дизайнер форм изображен на рисунке 5.

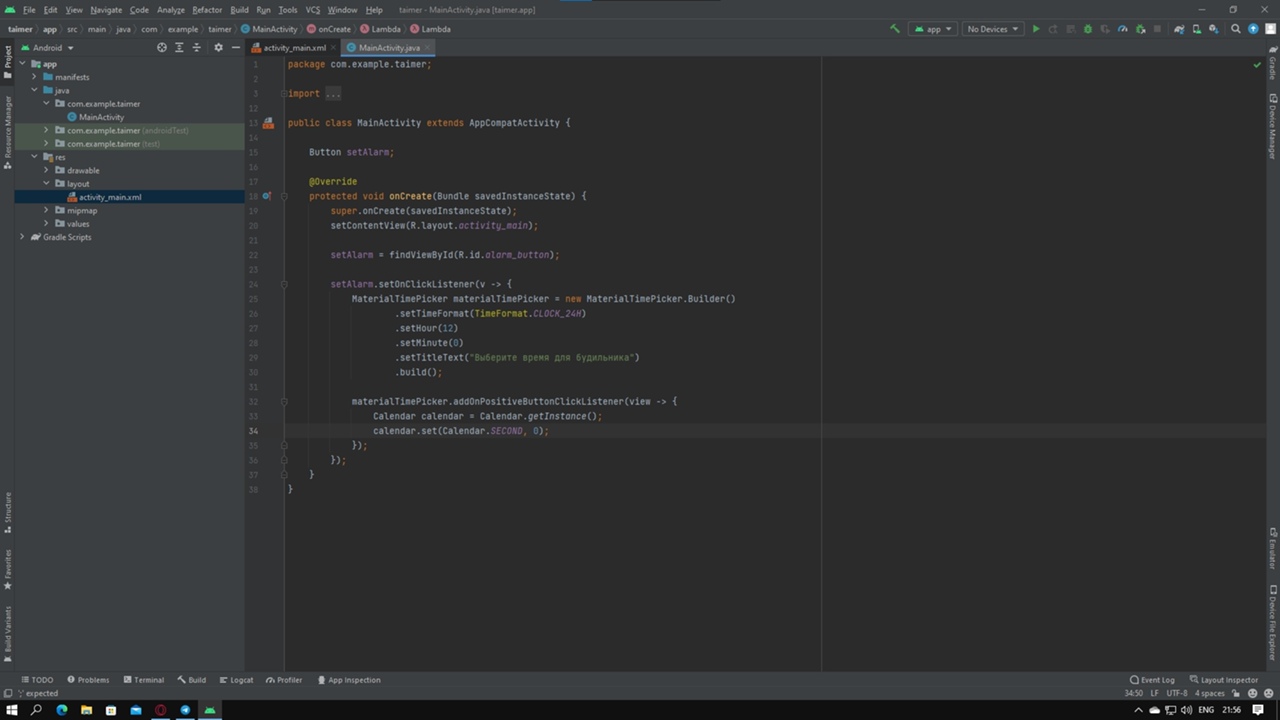


Рисунок 5 - Дизайнер форм в Java

Редактор кода - это, область в которой можно прописывать и редактировать код программы.

Основные процедуры, используемые для реализации часов:

procedure FormCreate(Sender: TObject) - процедура создания формы. Timer1Timer(Sender: TObject) - процедура работы таймера, который получает время. procedure N1Click(Sender: TObject) - процедура вызова таймера.C1Click(Sender: TObject) - процедура вызова секундомера.N2Click(Sender: TObject) - процедура выхода из программы.N3Click(Sender: TObject) - процедура вызова будильника.N4Click(Sender: TObject) - процедура вызова перемещения часов.Button1Click(Sender: TObject) - процедура установления.Button2Click(Sender: TObject) - процедура сброса.Button3Click(Sender: TObject) - процедура выхода из окна.

2.3 Разработка логики работы приложения. Схема взаимодействия компонентов проекта

Назначение программы - отображение времени в цифровой форме с возможностью вызова секундомера.

В рамках курсового проекта я создал макет пользовательского интерфейса приложения, теперь мне необходимо разработать схему окон приложения и служб данного приложения, для представления о выполняемых приложением функциях и происходящих при этом процессах.

Для работы программы мне необходимо: процессор с частотой 100 МГц и не менее 400МГц

* операционная система Windows
* манипулятор мышь и клавиатура
* объем дискового пространства 5 Мб
* оперативная память 64 Мб и более.

Для мобильного приложения активности (деятельности) определяют функциональные возможности любого приложения на платформе Android.

Существует четыре типа компонентов, применяемых в среде Android:

* Activity - деятельность;
* Service - служба;
* Content Provider – контент-провайдер;
* Broadcast Receiver – приёмник широковещательных намерений (асинхронных сообщений, активирующих деятельности, службы и приемники широковещательных намерений.

Намерение – это объект класса Intent, представляющий собой содержание сообщения.

2.4 Описание переменных, компонентов, классов и подпрограмм

Один из используемых компонентов в коде данного разрабатываемого приложения является Intent

Intent - это механизм для описания одной операции - выбрать фотографию, отправить письмо, сделать звонок, запустить браузер и перейти по указанному адресу. В Android-приложениях многие операции работают через намерения.

Но это не единственный вариант использования намерения. Также можно использовать для объявления о запуске активности или сервиса, направленных на выполнение каких-либо действий (как правило, речь о работе с определенной частью данных) или для передачи уведомлений о том, что произошло некое событие (или действие).

Намерения могут применяться для трансляции сообщений по системе. Любое приложение способно зарегистрировать широковещательный приёмник и отслеживать эти намерения с возможностью на них реагировать. Это позволяет создавать приложения, использующие событийную модель, в основе которой лежат внутренние, системные или сторонние события, передаваемые внешними программами.

Явные намерения:

Intent intent = new Intent(HelloWorld.this, AboutActivity.class);

startActivity(intent);

New - создание объекта который присваивается определённой переменной

Пример:

Second second = new Second();

В правой части с помощью ключевого слова new создается объект от класса Second. Созданный объект присваивается переменной second, чей тип указывается перед именем переменной. В случае наличия наследственных связей тип переменной может не совпадать с типом создаваемого объекта.

Class - шаблон или описание объекта,  а объект представляет экземпляр этого класса.

Класс определяется с помощью ключевого слова сlass:

class Person{

}

В данном случае класс называется Person. После названия класса идут фигурные скобки, между которыми помещается тело класса - то есть его поля и методы.

Для хранения состояния объекта в классе применяются поля или переменные класса. Для определения поведения объекта в классе применяются методы. Например, класс Person, который представляет человека, мог бы иметь следующее определение:

class Person{

    String name;       // имя

    int age;           // возраст

    void displayInfo(){

        System.out.printf("Name: %s \tAge: %d\n", name, age);

    }

}

В классе Person определены два поля: name представляет имя человека, а age - его возраст. И также определен метод displayInfo, который ничего не возвращает и просто выводит эти данные на консоль.

Private - закрытый класс или член класса, противоположность модификатору public. Закрытый класс или член класса доступен только из кода в том же классе.

Protected - такой класс или член класса доступен из любого места в текущем классе или пакете, или в производных классах, даже если они находятся в других пакетах/

Модификатор по умолчанию - отсутствие модификатора у поля или метода класса предполагает применение к нему модификатора по умолчанию. Такие поля или методы видны всем классам в текущем пакете.

Рассмотрим модификаторы доступа на примере следующей программы:

public class Program{

    public static void main(String[] args) {

        Person kate = new Person("Kate", 32, "Baker Street", "+12334567");

        kate.displayName();    // норм, метод public

        kate.displayAge();     // норм, метод имеет модификатор по умолчанию

        kate.displayPhone();   // норм, метод protected

        //kate.displayAddress(); // ! Ошибка, метод private

        System.out.println(kate.name);     // норм, модификатор по умолчанию

        System.out.println(kate.address);  // норм, модификатор public

        System.out.println(kate.age);      // норм, модификатор protected

        //System.out.println(kate.phone);  // ! Ошибка, модификатор private

    }

}

class Person{

    String name;

    protected int age;

    public String address;

    private String phone;

    public Person(String name, int age, String address, String phone){

        this.name = name;

        this.age = age;

        this.address = address;

        this.phone = phone;

    }

    public void displayName(){

        System.out.printf("Name: %s \n", name);

    }

    void displayAge(){

        System.out.printf("Age: %d \n", age);

    }

    private void displayAddress(){

        System.out.printf("Address: %s \n", address);

    }

    protected void displayPhone(){

        System.out.printf("Phone: %s \n", phone);

    }}

В данном случае оба класса расположены в одном пакете - пакете по умолчанию, поэтому в классе Program мы можем использовать все методы и переменные класса Person, которые имеют модификатор по умолчанию, public и protected. А поля и методы с модификатором private в классе Program не будут доступны.

Если бы класс Program располагался бы в другом пакете, то ему были бы доступны только поля и методы с модификатором public.

Модификатор доступа должен предшествовать остальной части определения переменной или метода.

Return - возращает методам некоторые их значения

После оператора return указывается возвращаемое значение, которое является результатом метода. Это может быть литеральное значение, значение переменной или какого-то сложного выражения.

Пример:

public class Program{

    public static void main (String args[]){

        int x = sum(1, 2, 3);

        int y = sum(1, 4, 9);

        System.out.println(x); // 6

        System.out.println(y); // 14

    }

    static int sum(int a, int b, int c){

        return a + b + c;

    }

}

В методе в качестве типа возвращаемого значения вместо void используется любой другой тип. В данном случае метод sum возвращает значение типа int, поэтому этот тип указывается перед названием метода. Причем если в качестве возвращаемого типа для метода определен любой другой, отличный от void, то метод обязательно должен использовать оператор return для возвращения значения.

При этом возвращаемое значение всегда должно иметь тот же тип, что значится в определении функции. И если функция возвращает значение типа int, то после оператора return стоит целочисленное значение, которое является объектом типа int. Как в данном случае это сумма значений параметров метода.

Extends - это ключевое слово, предназначенное для расширения реализации какого-то существующего класса. Создается новый класс на основе существующего, и этот новый класс расширяет (extends) возможности старого.

Код кнопок в программе:

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <Button  
 android:id="@+id/alarm\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Установить будильник"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

Так же в программе было создано звуковое и текстовое оповещение при срабатывании будильника:

Код текстового оповещения:

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Alarm!!!"  
 android:textSize="48sp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

Код звукого оповещения:

Ringtone ringtone;  
  
 @Override  
 protected void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 setContentView(R.layout.*activity\_alarm*);  
  
 Uri notificationUri = RingtoneManager.*getDefaultUri*(RingtoneManager.*TYPE\_ALARM*);  
 ringtone = RingtoneManager.*getRingtone*(this, notificationUri);  
 if (ringtone == null) {  
 notificationUri = RingtoneManager.*getDefaultUri*(RingtoneManager.*TYPE\_RINGTONE*);  
 ringtone = RingtoneManager.*getRingtone*(this, notificationUri);  
 }  
 if (ringtone != null) {  
 ringtone.play();  
 }  
 }  
  
 @Override  
 protected void onDestroy() {  
 if (ringtone != null && ringtone.isPlaying()) {  
 ringtone.stop();  
 }  
 super.onDestroy();  
 }  
}

Так же в моем коде используется:

SimpleDateFormat - позволяет Вам запускать, выбирая любые определяемые пользователем образцы для разового датой форматирования. Однако, Вы поощряетесь создать разовое датой средство форматирования с также getTimeInstance, getDateInstance, или getDateTimeInstance в DateFormat. Каждый из этих методов класса может возвратить средство форматирования даты/времени, инициализированное с образцом формата по умолчанию. Можно изменить образец формата, используя applyPattern методы как требующийся. Для получения дополнительной информации об использовании этих методов.

Так же в моей программе были импортированы следующие модули:

import android.app.AlarmManager;  
import android.app.PendingIntent;  
import android.content.Context;  
import android.content.Intent;  
import android.os.Bundle;  
import android.widget.Button;  
import android.widget.Toast;

AlarmManager - всегда регистрируется с намерением. Когда срабатывает сигнал тревоги, намерение, зарегистрированное в AlarmManager, транслируется системой автоматически. Это намерение запускает целевое приложение, если оно не запущено. Рекомендуется использовать AlarmManager, когда вы хотите, чтобы код вашего приложения запускался в определенное время, даже если ваше приложение в данный момент не запущено. Для других операций синхронизации следует использовать обработчик, потому что он прост в использовании. Обработчик рассматривается в другом уроке.

PendingIntent – Объект PendingIntent оборачивает функциональность объекта Intent, позволяя вашему приложению указать, что другое приложение должно сделать от вашего имени в ответ на будущее действие. Например, обернутое намерение может быть вызвано при срабатывании будильника или когда пользователь нажимает на уведомление.

Context - это базовый абстрактный класс, реализация которого обеспечивается системой Android. Этот класс имеет методы для доступа к специфичным для конкретного приложения ресурсам и классам и служит для выполнения операций на уровне приложения, таких, как запуск активностей, отправка широковещательных сообщений, получение намерений и прочее.

Bundle - по сути представляет собой враппер (оболочку) над коллекцией ArrayMap для создания более комфортного в работе контейнера для элементов разных типов, в котором можно разместить любые Parcelabe-объекты и примитивные типы. Класс является потокобезопасным и может использоваться для передачи значений между разными потоками. Доступ к элементам этой коллекции осуществляется, как и в ArrayMap, по парам ключ-значение. Дополнительно класс предоставляет методы по размещению и извлечению типизированных данных и некоторые другие возможности.

Так же в программе используется подпрограмма(метод)

OnCreate:

protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);

OnCreate задаёт начальную установку параметров при инициализации активности.

3 Тестирование и установка приложений

При тестировании возникало не мало ошибок как в самом коде, так же и в приложении такие как:

1. Не рабочее текстовое оповещение о срабатывании будильника
2. Ошибки при введении новых переменных
3. Ошибки при импорте

Но они все были успешно устранены и приложении в полностью рабочем состоянии

Далее приложение было выгружено на HONOR 8A на базе android 10, где оно присутствует до сих пор.

Так же на настоящем телефоне приложение работало, как и планировалось.

# Заключение

В этом курсовом проекте я проанализировал популярные программы и создал свое альтернативное приложение, которое наверняка найдёт спрос среди определённых кругов лиц.

Так же создавая данную программу, я использовал множество разных классов методов и компонентов, создав данную программу я изучил и узнал довольно много нового о создании программ и о языке java в целом, это очень ценный опыт для студента.

# Список используемой литературы

1. Блоха Д. Эффективная Java / Джошуа Блох – Второе издание 2019 – 431 с. [4]
2. Дерендяев П., Java-разработчик с нуля/ П. Дерендяев – 2021 г [7]
3. Дробышевский С.В., Функция для анализа тега MP3 ID3v2/ С.В. Дробышевский – 2022 г [9]
4. Кей С. Хорстманн Java. Библиотека профессионала / Кей С. Хорстманн – Том 1 Основы 2019 – 864 с. [13]
5. Климов А., Методы активности Android/ А. Климов – 2022 г [16]
6. Прокофьев С., Activity и жизненный цикл приложения/ С. Прокофьев - 2012-2022 г [18]
7. Рокатанский М., Всё о PendingIntents/ М. Рокатанский – 2006 г [20]
8. Рубин Э., Что такое Android? Руководство для начинающих/ Эндрю Рубин – 2019 г [23]
9. Русские блоги, Понимание и применение контекста в разработке под Android/ - 2020-2022 г [25]
10. Седжвик Р. Алгоритмы на Java / Роберт Седжвик, Кевин Уэйн – 4-е издание 2013 – 894 с [25]

# Приложение А. Листинг рабочего кода

AlarmActivity.java

package com.arhiser.alarmc;  
  
import android.media.Ringtone;  
import android.media.RingtoneManager;  
import android.net.Uri;  
import android.os.Bundle;  
  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
  
public class AlarmActivity extends AppCompatActivity {  
  
 Ringtone ringtone;  
  
 @Override  
 protected void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 setContentView(R.layout.*activity\_alarm*);  
  
 Uri notificationUri = RingtoneManager.*getDefaultUri*(RingtoneManager.*TYPE\_ALARM*);  
 ringtone = RingtoneManager.*getRingtone*(this, notificationUri);  
 if (ringtone == null) {  
 notificationUri = RingtoneManager.*getDefaultUri*(RingtoneManager.*TYPE\_RINGTONE*);  
 ringtone = RingtoneManager.*getRingtone*(this, notificationUri);  
 }  
 if (ringtone != null) {  
 ringtone.play();  
 }  
 }  
  
 @Override  
 protected void onDestroy() {  
 if (ringtone != null && ringtone.isPlaying()) {  
 ringtone.stop();  
 }  
 super.onDestroy();  
 }

MainActivity.java

package com.arhiser.alarmc;  
  
import android.app.AlarmManager;  
import android.app.PendingIntent;  
import android.content.Context;  
import android.content.Intent;  
import android.os.Bundle;  
import android.widget.Button;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import com.google.android.material.timepicker.MaterialTimePicker;  
import com.google.android.material.timepicker.TimeFormat;  
  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Locale;  
  
public class MainActivity extends AppCompatActivity {  
  
 Button setAlarm;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 SimpleDateFormat sdf = new SimpleDateFormat("HH:mm", Locale.*getDefault*());  
  
 setAlarm = findViewById(R.id.*alarm\_button*);  
  
 setAlarm.setOnClickListener(v -> {  
 MaterialTimePicker materialTimePicker = new MaterialTimePicker.Builder()  
 .setTimeFormat(TimeFormat.*CLOCK\_24H*)  
 .setHour(12)  
 .setMinute(0)  
 .setTitleText("Выберите время для будильника")  
 .build();  
  
 materialTimePicker.addOnPositiveButtonClickListener(view -> {  
 Calendar calendar = Calendar.*getInstance*();  
 calendar.set(Calendar.*SECOND*, 0);  
 calendar.set(Calendar.*MILLISECOND*, 0);  
 calendar.set(Calendar.*MINUTE*, materialTimePicker.getMinute());  
 calendar.set(Calendar.*HOUR\_OF\_DAY*, materialTimePicker.getHour());  
  
 AlarmManager alarmManager = (AlarmManager) getSystemService(Context.*ALARM\_SERVICE*);  
  
 AlarmManager.AlarmClockInfo alarmClockInfo = new AlarmManager.AlarmClockInfo(calendar.getTimeInMillis(), getAlarmInfoPendingIntent());  
  
 alarmManager.setAlarmClock(alarmClockInfo, getAlarmActionPendingIntent());  
 Toast.*makeText*(this, "Будильник установлен на " + sdf.format(calendar.getTime()), Toast.*LENGTH\_SHORT*).show();  
 });  
  
 materialTimePicker.show(getSupportFragmentManager(), "tag\_picker");  
 });  
  
   
 }  
 private PendingIntent getAlarmInfoPendingIntent() {  
 Intent alarmInfoIntent = new Intent(this, MainActivity.class);  
 alarmInfoIntent.setFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP* | Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  
 return PendingIntent.*getActivity*(this, 0, alarmInfoIntent, PendingIntent.*FLAG\_UPDATE\_CURRENT*);  
 }  
  
 private PendingIntent getAlarmActionPendingIntent() {  
 Intent intent = new Intent(this, AlarmActivity.class);  
 intent.setFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP* | Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  
 return PendingIntent.*getActivity*(this, 1, intent, PendingIntent.*FLAG\_UPDATE\_CURRENT*);  
 }  
}

activityalarm.xml

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <TextView  
 android:layout\_width="wrap\_content"

установленная широта оповещения  
 android:layout\_height="wrap\_content"

установленная высота оповещения  
 android:text="Alarm!!!"

текст установленный на оповещение  
 android:textSize="48sp"

размер текста на оповещение  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />

activitymain.xml

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <Button  
 android:id="@+id/alarm\_button"

присваивание id к кнопке   
 android:layout\_width="wrap\_content"

установленная широта  
 android:layout\_height="wrap\_content"

установленная высота  
 android:text="Установить будильник"

текст установленынй на кнопке  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>