Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***ИУК «Информатика и управление»***

КАФЕДРА ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

ЛАБОРАТОРНАЯ РАБОТА №8

«Сервисы»

ДИСЦИПЛИНА: «Разработка мобильного ПО»

Выполнил: студент гр. ИУК4-62Б _____ (Карельский М.К.)
(Подпись)

Проверил: _____ (Шаматрин А.Г.)
(Подпись)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2023

Цель: формирование практических навыков создания различных android-служб.

Задачи:

1. Научиться создавать различные службы для мобильного устройства.
2. Уметь понимать схемы взаимодействия службы с другими элементами платформы Android.
3. Разработать эффективные приложения с учетом аппаратных ограничений мобильных устройств.

Вариант 4

Создать службу контроля зарядки аккумуляторной батареи. Приложение должно иметь возможность установки предельно допустимой скорости разрядки аккумуляторной батареи, получения текущего состояния заряда. Скорость разрядки измеряется в единицах процент/минута. При превышении предельно допустимой скорости разрядки аккумуляторной батареи служба должна посылать уведомление. Также необходимо предусмотреть возможность включения и отключения службы.

Листинг:

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission
android:name="android.permission.BATTERY_PROPERTY_ENERGY_COUNTER" />
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.LW328"
        tools:targetApi="31">
        <service
            android:name=".BatteryService"
            android:enabled="true"
            android:exported="true"></service>

        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

```
</manifest>
```

BatteryService.kt

```
package com.blackline.lw3_2_8

import android.app.Notification
import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.Service
import android.content.Context
import android.content.Intent
import android.graphics.BitmapFactory
import android.graphics.Color
import android.os.BatteryManager
import android.os.Handler
import android.os.IBinder
import android.util.Log
import android.widget.Toast

class BatteryService : Service() {
    private var handler: Handler = Handler()
    private var previousCharge = 0.0
    private var speed = 0.0
    private var maxCapacity = 0.0
    private var isNotified = false
    private lateinit var batteryManager : BatteryManager

    lateinit var notificationManager: NotificationManager
    lateinit var notificationChannel: NotificationChannel
    lateinit var builder: Notification.Builder
    private val channelId = "i.apps.notifications"
    private val description = "Test notification"

    private val chargeRunnable = object : Runnable {
        override fun run() {
            val batteryLevelMilliAmpere =
batteryManager.getIntProperty(BatteryManager.BATTERY_PROPERTY_CHARGE_COUNTER) /
1000.0
            val dischargeSpeed = (previousCharge - batteryLevelMilliAmpere) * 60
/ 3 / maxCapacity * 100
            previousCharge = batteryLevelMilliAmpere

            if (dischargeSpeed != 0.0) {
                if (dischargeSpeed > speed && !isNotified) {
                    isNotified = true
                    notificationManager.notify(1234, builder.build())
                }

                Log.d("ChargeSpeed", "Скорость разрядки %/min: $dischargeSpeed")
            }

            handler.postDelayed(this, 3000)
        }
    }

    override fun onBind(intent: Intent): IBinder? {
        return null
    }

    override fun onCreate() {
        super.onCreate()

        notificationManager = getSystemService(Context.NOTIFICATION_SERVICE) as
```

```
NotificationManager

        notificationChannel = NotificationChannel(channelId, description,
NotificationManager.IMPORTANCE_HIGH)
        notificationChannel.enableLights(true)
        notificationChannel.lightColor = Color.GREEN
        notificationChannel.enableVibration(false)
        notificationManager.createNotificationChannel(notificationChannel)

        builder = Notification.Builder(this, channelId)
            .setSmallIcon(R.drawable.ic_launcher_foreground)
            .setLargeIcon(BitmapFactory.decodeResource(this.resources,
R.drawable.ic_launcher_foreground))
            .setContentTitle("The phone discharges")
            .setContentText("Your discharge limits has been exceeded")

        batteryManager = getSystemService(Context.BATTERY_SERVICE) as
BatteryManager

        val mPowerProfile: Any
        var batteryCapacity = 0.0
        val POWER_PROFILE_CLASS = "com.android.internal.os.PowerProfile"
        try {
            mPowerProfile = Class.forName(POWER_PROFILE_CLASS)
                .getConstructor(Context::class.java)
                .newInstance(this.applicationContext)
            batteryCapacity = Class
                .forName(POWER_PROFILE_CLASS)
                .getMethod("getBatteryCapacity")
                .invoke(mPowerProfile) as Double
        } catch (e: Exception) {
            e.printStackTrace()
        }

        maxCapacity = batteryCapacity
    }

    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int
{
        speed = intent?.extras!!.getString("speed")!!.toDouble()

        val batteryLevelMilliAmpere =
batteryManager.getIntProperty(BatteryManager.BATTERY_PROPERTY_CHARGE_COUNTER) /
1000.0
        previousCharge = batteryLevelMilliAmpere
        chargeRunnable.run()

        Toast.makeText(this, "Battery service has been started",
Toast.LENGTH_SHORT).show()
        return super.onStartCommand(intent, flags, startId)
    }

    override fun onDestroy() {
        super.onDestroy()

        handler.removeCallbacks(chargeRunnable)
        Toast.makeText(this, "Battery service has been stopped",
Toast.LENGTH_SHORT).show()
    }
}
```

MainActivity.kt

```
package com.blackline.lw3_2_8
```

```
import android.content.Context
import android.content.Intent
import android.os.BatteryManager
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.Handler
import android.view.View
import android.widget.EditText
import android.widget.TextView
import java.text.DecimalFormat

class MainActivity : AppCompatActivity() {
    private var handler: Handler = Handler()
    private var previousCharge = 0.0
    private var maxCapacity = 0.0
    private lateinit var batteryManager : BatteryManager
    private lateinit var chargeTextView : TextView

    private val chargeRunnable = object : Runnable {
        override fun run() {
            val batteryLevelMilliAmpere =
batteryManager.getIntProperty(BatteryManager.BATTERY_PROPERTY_CHARGE_COUNTER) /
1000.0
            val dischargeSpeed = (previousCharge - batteryLevelMilliAmpere) * 60
/ 3 / maxCapacity * 100
            previousCharge = batteryLevelMilliAmpere

            chargeTextView.text =
"${DecimalFormat("#.###").format(dischargeSpeed)} %/min"

            handler.postDelayed(this, 3000)
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity main)

        chargeTextView = findViewById<TextView>(R.id.chargeTextView)

        batteryManager = getSystemService(Context.BATTERY_SERVICE) as
BatteryManager

        val mPowerProfile: Any
        var batteryCapacity = 0.0
        val POWER_PROFILE_CLASS = "com.android.internal.os.PowerProfile"
        try {
            mPowerProfile = Class.forName(POWER_PROFILE_CLASS)
                .getConstructor(Context::class.java)
                .newInstance(this.applicationContext)
            batteryCapacity = Class
                .forName(POWER_PROFILE_CLASS)
                .getMethod("getBatteryCapacity")
                .invoke(mPowerProfile) as Double
        } catch (e: Exception) {
            e.printStackTrace()
        }

        maxCapacity = batteryCapacity

        val batteryLevelMilliAmpere =
batteryManager.getIntProperty(BatteryManager.BATTERY_PROPERTY_CHARGE_COUNTER) /
1000.0
        previousCharge = batteryLevelMilliAmpere
        chargeRunnable.run()
    }
```

```
    fun onStartButtonClick(view: View) {
        val speedTextView =
findViewById<EditText>(R.id.speedEditTextNumberDecimal)
        val speed = speedTextView.text.toString()
        if (speed != "")
            startService(Intent(this@MainActivity,
BatteryService::class.java).apply {
                putExtra(
                    "speed",
                    speed
                )
            })
    }

    fun onStopButtonClick(view: View) {
        stopService(Intent(this@MainActivity, BatteryService::class.java))
    }

    override fun onDestroy() {
        super.onDestroy()

        handler.removeCallbacks(chargeRunnable)
    }
}
```

activity_main.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/startButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:onClick="onStartButtonClick" />

    <Button
        android:id="@+id/stopButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Stop"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/startButton"
        android:onClick="onStopButtonClick" />

    <EditText
        android:id="@+id/speedEditTextNumberDecimal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Discharge speed"
        android:inputType="numberSigned|numberDecimal"
```

```
            android:textAlignment="center"
            android:textSize="20sp"
            app:layout_constraintBottom_toTopOf="@+id/startButton"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent" />

        <TextView
            android:id="@+id/chargeTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="0 %/min"
            android:textSize="50sp"
            app:layout_constraintBottom_toTopOf="@+id/speedEditTextNumberDecimal"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Результат:

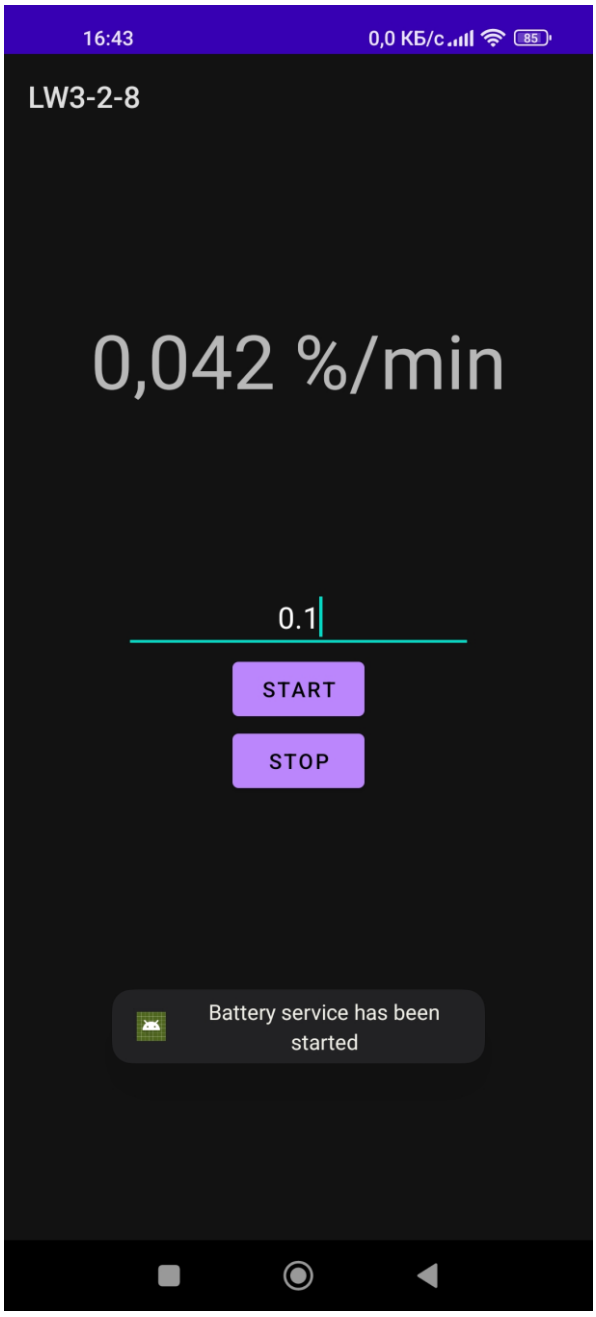

Рис. 1. Запуск сервиса

Рис. 2. Остановка сервиса

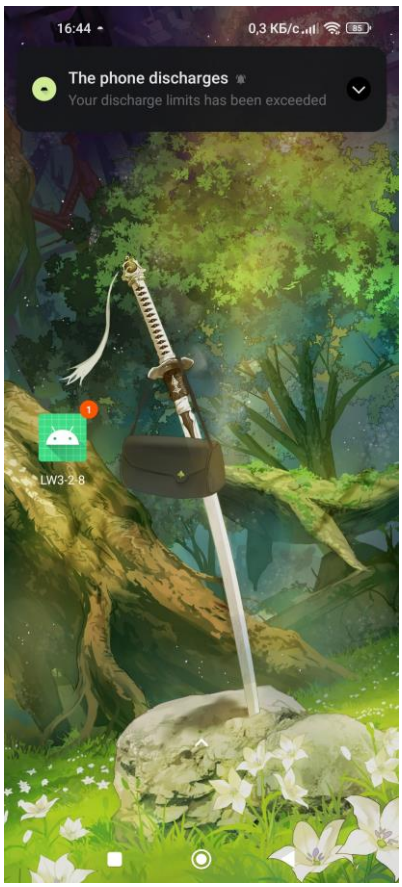

Рис. 3. Уведомление о превышении скорости разрядки

Вывод: в ходе выполнения лабораторной работы были получены практические навыки создания различных android-служб.