На учебной практике вам предстоит решить несколько заданий. Весь неизвестный функционал, который будет использоваться в представленном коде, на учебной практике является материалом для самостоятельного изучения

Задание 2. Секундомер

Вашим вторым заданием будет разработать мобильное приложение "Секундомер"

если для пункта, который следует выполнить нет пояснений как его выполнить, значит вы это уже делали на практических работах

Ход выполнения работы:

1. Создайте новый проект, минимальная версия операционной системы Android 12 (если указать ниже - библиотека не будет работать и эффект не получится).

Подготовительный этап. Создание экрана загрузки

При создании загрузочного экрана будет использоваться API от разработчиков Android Studio SplashScreen Compat API.
https://developer.android.com/develop/ui/views/launch/splash-screen

2. Поместите в проект изображение, которое вы бы хотели использовать в качестве изображения на экране загрузки (можно использовать предложенное или найти свое). Обратите внимание, его размеры должны быть не больше 288*288 px. Изображение может быть как растровым так и векторным.
3. Откройте build.gradle.kts (**Module**) и пропишите следующую зависимость:

```
38  dependencies { this: DependencyHandlerScope
39      implementation("androidx.core:core-splashscreen:1.0.1")
40      implementation(libs.androidx.core.ktx)
41      implementation(libs.androidx.appcompat)
42      implementation(libs.material)
```

4. Синхронизируйте проект
5. Добавьте в проект цвет для фона загрузочного экрана:

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <color name="black">#FF000000</color>
4      <color name="white">#FFFFFFFF</color>
5      <color name="red">#cd373c</color>
6  </resources>
```

6. Удалите ночную тему.

7. Настройте тему вашего приложения, добавив в нее стиль для оформления загрузочного экрана.
8. Первое имя прописываете сами, второе - выбираете из предложенного списка (строка 9).

```xml
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.MyApplication" parent="Theme.Material3.DayNight.NoActionBar">
        <!-- Customize your light theme here. -->
        <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
    </style>

    <style name="Theme.MyApplication" parent="Base.Theme.MyApplication" />
    <style name="Theme.App.SplashScreen" parent="Theme.SplashScreen">
        <item name="windowSplashScreenBackground">@color/red</item>//задаем цвет фона заставки
        <item name="windowSplashScreenAnimatedIcon">@drawable/logo</item>//задаем изображение, которое будет появляться на заставке
        <item name="postSplashScreenTheme">@style/Base.Theme.MyApplication</item>//указываем родительскую тему, у которой нужно вернуться
                                                                                  //как только время отображения заставки закончится см.3 строка
    </style>
</resources>
```

9. В манифесте установите созданный стиль для стартового экрана:

```xml
<activity
    android:theme="@style/Theme.App.SplashScreen"
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

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

10. В файле логики стартового экрана настройте отображение “загрузочного экрана”

activity_main.xml | MainActivity.kt | AndroidManifest.xml | build.gradle.kts (:app) | themes.xml | colors.xml

```kotlin
package com.example.myapplication

import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.splashscreen.SplashScreen.Companion.installSplashScreen
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Thread.sleep( millis: 3000)//время отображения экрана загрузки
        installSplashScreen()//подгружается из библиотеки и реализует отображение экрана загрузки
```

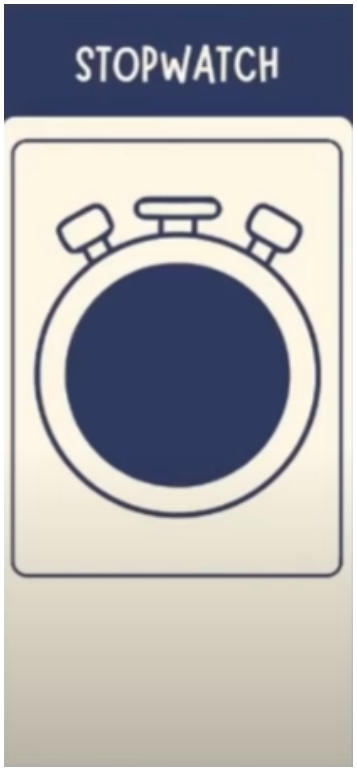
11. Запустите приложение, посмотрите как теперь отображается загрузочный экран
12. Оформите в отчет ответы на следующие вопросы: что такое потоки (класс Thread), для чего они используются? Что делает функция Thread.sleep?

Создание приложения “Секундомер”. Верстка

13. Подкорректируйте тему приложения (продолжаем работу над созданным ранее проектом):

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.MyApplication" parent="Theme.Material3.DayNight.NoActionBar">
        <!-- Customize your light theme here. -->
    <item name="colorPrimary">@color/red</item>
        <item name="colorPrimaryVariant">@color/red</item>
        <item name="android:statusBarColor">@color/red</item> //если версия Игуана, действия этой строки видно не будет
    </style>

```

14. Самостоятельно подключите биндинги в вашем приложении (см.прошлую практическую работы).
15. Синхронизируйте проект.
16. Используя любой графический редактор, отрисуйте экран с секундомером, это изображение будет использоваться в качестве фона (размер холста задайте пропорциональным или равным размеру экрана телефона, на котором отображается ваше приложение). Сохраните в растровом формате. За основу можно взять такой пример:



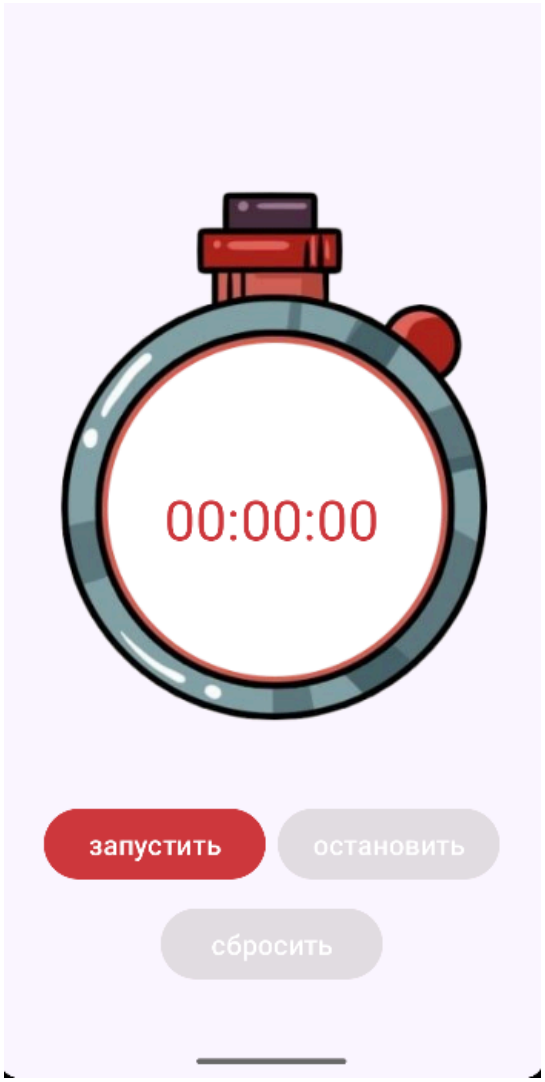
17. Скопируйте нарисованный секундомер в соответствующую папку проекта.
18. Выполните верстку экрана с секундомером (файл разметки Activity Main). Помимо фонового изображения добавьте три кнопки и элемент для отображения текста внутри секундомера (дизайн контролируйте при запуске приложения, в режиме Design UI может отображаться некорректно):

```
<Button
    android:id="@+id/startBtn"
```

```
<Button
    android:id="@+id/stopBtn"
    android:enabled="false"
```

- enabled = false кнопка изначально будет отключена

```
<Button
    android:id="@+id/resetBtn"
    android:enabled="false"
```



Создание приложения “Секундомер”. Настройка логики

19.Подключите использование биндингов на активити:

```
class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Thread.sleep( millis: 3000)//время отображения экрана загруз
        installSplashScreen()//подгружается из библиотеки и реали
        enableEdgeToEdge()
        binding=ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
```

20. Настройте логику

MainActivity.kt

```
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.splashscreen.SplashScreen.Companion.installSplashScreen
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import com.example.myapplication.databinding.ActivityMainBinding
import kotlinx.coroutines.Runnable

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    private var isRunning = false //переменная используется для отслеживания запущен таймер в данный момент или нет
    private var timerSeconds = 0 //количество секунд, которые отсчитывает таймер
    private val handler = Handler(Looper.getMainLooper())//
    private val runnable = object : Runnable {//объект наследуется от интерфейса Runnable
        override fun run() {//как только активируется функция run начнется отсчет секунд
            timerSeconds++
            val hours : Int  = timerSeconds/3600
            val minutes : Int  = (timerSeconds%3600)/60
            val seconds : Int  = timerSeconds%60
            val time : String  = String.format("%02d:%02d:%02d", hours, minutes, seconds)//форматированный вывод значений переменных
            binding.textView.text=time
        handler.postDelayed( r: this,  delayMillis: 1000)//.postDelayed позволяет запланировать повторный запуск Runnable после задержки в 1 секунду
        }

    }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Thread.sleep( millis: 3000)//время отображения экрана загрузки
```

***Handler** может использоваться для планирования выполнения кода в некоторый момент в будущем. Также класс может использоваться для передачи кода, который должен выполняться в другом программном потоке*

Looper - это что-то вроде фонового потока, который работает постоянно, не завершаясь. Чтобы передать в этот фоновый поток выполнение какой-либо операции, используется Handler.

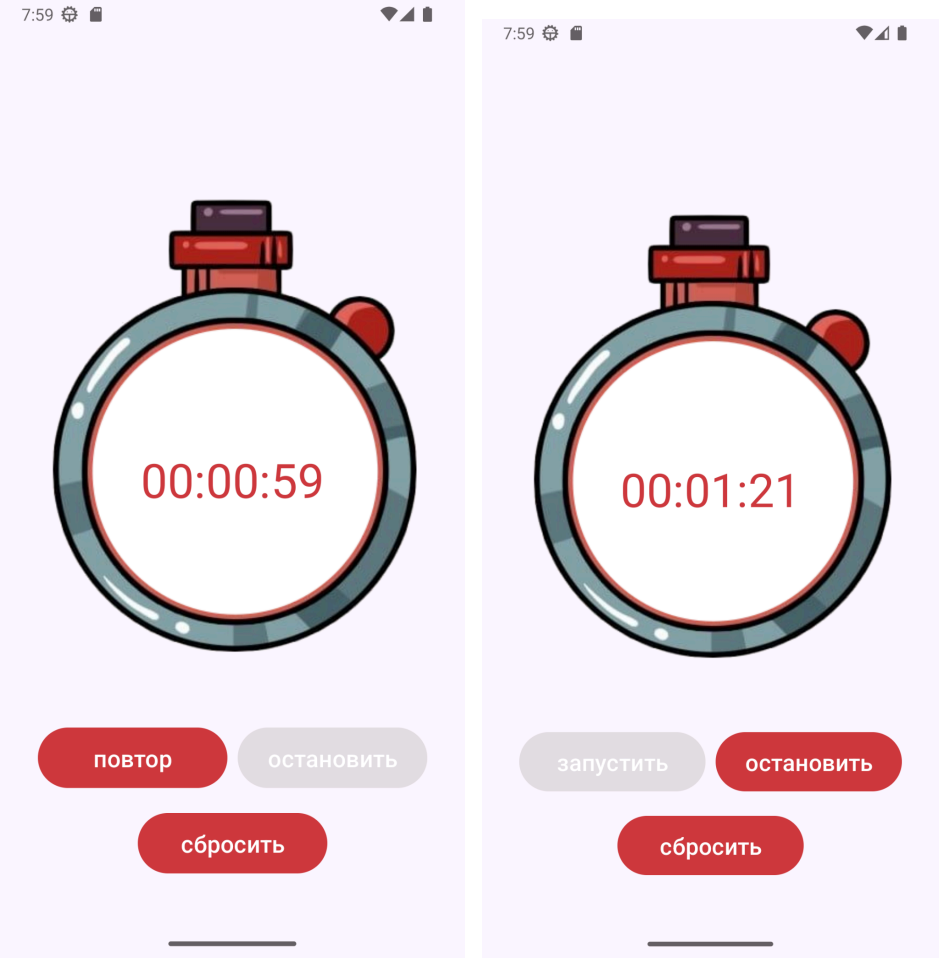
*Интерфейс **Runnable** — это задача, которую выполняет поток, то есть код. Интерфейс содержит основной метод run() — в нём и находится точка входа и логика исполняемого потока.*

MainActivity.kt

```
        )//.postDelayed позволяет запланировать повторный запуск Runnable после задержки в 1 секунду
    }

}

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    Thread.sleep( millis: 3000)//время отображения экрана загрузки
    installSplashScreen()//подгружается из библиотеки и реализует отображение экрана загрузки
    enableEdgeToEdge()
    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
        val systemBars : Insets  = insets.getInsets(WindowInsetsCompat.Type.systemBars())
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
        insets
    }

}

private fun startTimer() {//запускаем таймер
    if (!isRunning) {
        handler.postDelayed(runnable,  delayMillis: 1000)//запуск таймера с задержкой 1000 милисекунд
        // после каждой 2 секунды он будет увеличиваться
        isRunning = true
        binding.startBtn.isEnabled = false
        binding.stopBtn.isEnabled = true
        binding.resetBtn.isEnabled = true
    }
}

private fun stopTimer() {//запускаем таймер
    if (isRunning) {
        handler.removeCallbacks(runnable)//используется для остановки выполнения функции run
        isRunning = false
        binding.startBtn.isEnabled = true
        binding.startBtn.text =
            "возобновить"//если пользователь нажмет на кнопку стоп, название кнопки старт поменяется
        binding.stopBtn.isEnabled = false
        binding.resetBtn.isEnabled = true
    }
}

private fun resetTimer() {//сброс
    stopTimer()
    timerSeconds = 0
    binding.textView.text = "00:00:00"
    binding.startBtn.isEnabled = true
    binding.stopBtn.isEnabled = false
    binding.startBtn.text = "запустить"
}
}
```

21. Самостоятельно, с помощью слушателя нажатий SetOnClickListener, назначьте на каждую кнопку соответствующее ей действие.

22.Запустите приложение, проверьте как работает секундомер.



23.Поэкспериментируйте с временем задержки (100, 10)