Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт лабораторной работе №2**

Дисциплина: Проектирование мобильных приложений

Тема: Activity Lifecycle. Alternative resources.

Выполнил студент гр. 3530901/90201 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Швецов Д.С.

(подпись)

Принял \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кузнецов А. Н.

(подпись)

“ ” 2022 г.

Санкт-Петербург

2022

Цели:

* Познакомиться с жизненным циклом Activity
* Изучить основные возможности и свойства alternative resources.

**Выполнение 1 цели**

**Задание 1.**

Ознакомление с жизненным циклом Activity, только нужно сделать это на нетривиальном примере.

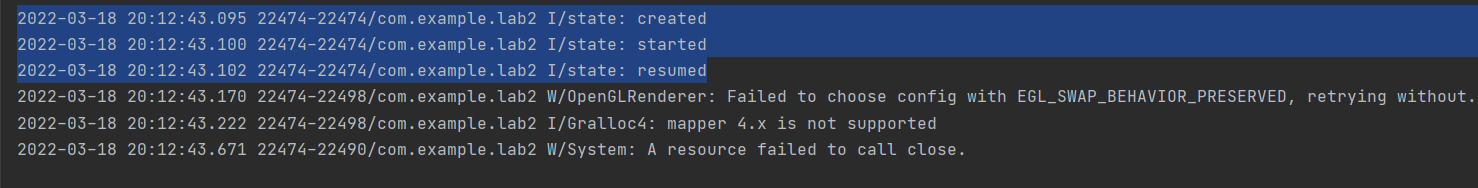
1. Попробуем завершить приложение с помощью сигнала STOP, который реализован как force-stop

Команда:



И к удивлению, приложение закрылось, но никаких выводов в лог не было, действительно, нестандартная ситуация.

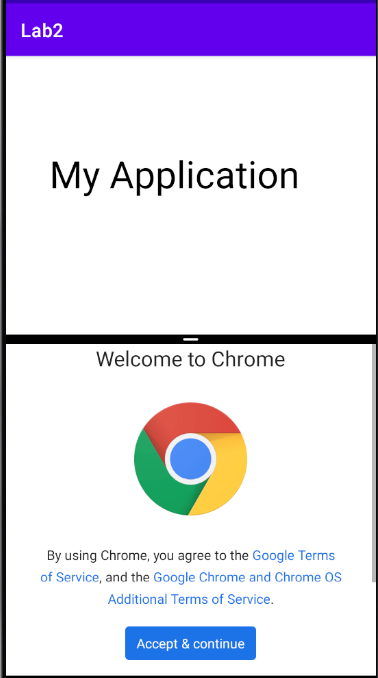
Вывод лога:



Видим, что приложение было в состоянии “resumed” и после просто завершилось.

1. Попробуем открыть приложение в мультиоконном режиме.

Мультиоконный режим:



Переход состояний:

Изображение выглядит как текст

Автоматически созданное описание

Видим, что первые 3 состояния как и в 1 случае, далее мы вывели приложение в background, поэтому оно перешло в состояние paused, stopped, created, started, resumed, и когда мы начали делить экран оно перешло в paused, когда открыли chrome перешло в resumed.

1. Попробуем позвонить на устройство.

Вид звонка:

Изображение выглядит как текст

Автоматически созданное описание

Изменения состояний

Изображение выглядит как текст

Автоматически созданное описание

Видим, что ни в какое состояние после создания приложение не перешло.

**Выполнение 2 цели**

**Задание 2.**

Alternative Resources для варианта 21(6).

Требуется придумать ситуацию, где применим альтернативный ресурс screen aspect. Вообще screen aspect отвечает за соотношение сторон, всего существует 2 значения: long(широкий) и not long.

Пример использования: допустим, что мы написали наш UI для приложения, которое работало на телефонах с экранами типа WQVGA, то есть широкими, и если мы бы не написали реализацию для not long, то на телефонах с экранами типа QVGA( не широким) наш UI сжался бы и некоторые кнопки возможно перекрывали друг друга или что-то подобное, к примеру, в сервисе заказа такси не смогли бы нормально выбрать способ оплаты проезда.

**Задание 3.**

Таблица 1. Конфигурации в задании.

|  |  |
| --- | --- |
| Конфигурация устройства | Конфигурация ресурсов |
| |  | | --- | | LOCALE\_LANG: fr | |  |  |  | | --- | | LOCALE\_REGION: rFR | |  |  |  | | --- | | SCREEN\_SIZE: xlarge | |  |  |  | | --- | | SCREEN\_ASPECT: long | |  |  |  | | --- | | ROUND\_SCREEN: notround | |  |  |  | | --- | | ORIENTATION: land | |  |  |  | | --- | | UI\_MODE: appliance | |  |  |  | | --- | | NIGHT\_MODE: notnight | |  |  |  | | --- | | PIXEL\_DENSITY: xhdpi | |  |  |  | | --- | | TOUCH: notouch | |  |  |  | | --- | | PRIMARY\_INPUT: qwerty | |  |  |  | | --- | | NAV\_KEYS: nonav | |  |   PLATFORM\_VER: v25 | |  | | --- | | (default) | |  |  |  | | --- | | rFR-finger | |  |  |  | | --- | | round-port-nonav-v25 | |  |  |  | | --- | | small-port-night-hdpi-v25 | |  |  |  | | --- | | port-notouch | |  |  |  | | --- | | en-rCA-normal-night-notouch-nokeys | |  |  |  | | --- | | long-round-port | |  |  |  | | --- | | rUS | |  |  |  | | --- | | television-finger-v25 | |  |  |  | | --- | | large-night-nokeys-trackball | |  |   normal-land-watch-nonav-v25 |

1. Шаг

Уберем конфигурации ресурсов, которые противоречат конф. ус-ва.

|  |  |
| --- | --- |
| Название | То, что должно быть |
| default | Подходит |
| |  | | --- | | rFR-finger | | |  | | --- | | TOUCH: notouch | |  | |
| round-port-nonav-v25 | |  | | --- | | ROUND\_SCREEN: notround | |
| small-port-night-hdpi-v25 | |  | | --- | | SCREEN\_SIZE: xlarge | |
| port-notouch | |  | | --- | | ORIENTATION: land | |
| |  | | --- | | en-rCA-normal-night-notouch-nokeys | | |  | | --- | | LOCALE\_LANG: fr | |
| |  | | --- | | long-round-port | | |  | | --- | | ROUND\_SCREEN: notround | |
| |  | | --- | | rUS | | |  | | --- | | LOCALE\_REGION: rFR | |
| |  | | --- | | television-finger-v25 | | |  | | --- | | UI\_MODE: appliance | |
| |  | | --- | | large-night-nokeys-trackball | | |  | | --- | | SCREEN\_SIZE: xlarge | |
| normal-land-watch-nonav-v25 | |  | | --- | | SCREEN\_SIZE: xlarge | |

Итого мы видим, что все ресурсы, кроме default не подходят, поэтому мы оставим конфигурацию default.

**Задание 4.**

Начнем с нахождения ошибок

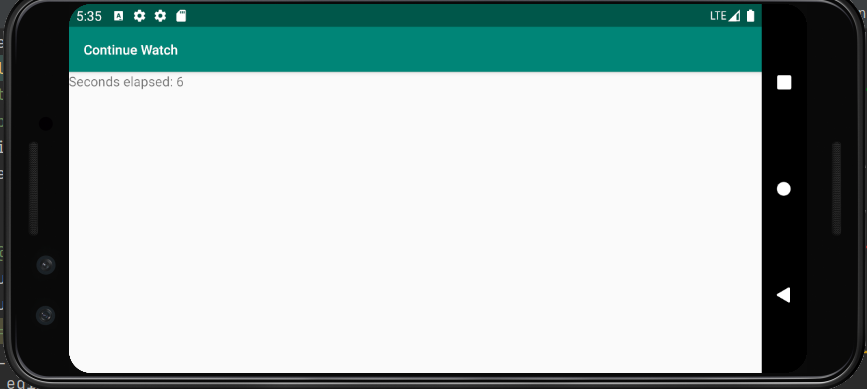
Ошибка №1

В activity\_main.xml (land) text\_view не привязан ни к чему.

Изображение выглядит как текст

Автоматически созданное описание

Из-за этого получается, что при повороте текстовое поле съезжает.



Исправление:

Изображение выглядит как текст

Автоматически созданное описание

Ошибка 2: при входе приложения в background счет продолжается, хотя по документации должен останавливаться.

Ошибка 3: при закрытии приложения поток, производящий счет не заканчивает работу, а просто теряется.

Ошибки 2 и 3 мы можем решить вместе: в методе onPaused мы будем завершать выполнение потока, а в методе onResumed стартовать новый поток, в таком случае старый поток не будет работать где-то после потери ссылки на него и приложение не будет считать в Background.

Вариант решения:

Изображение выглядит как текст

Автоматически созданное описание

Ошибка 4: при повороте экрана счет сбрасывается.

Эту ошибку мы можем решим 2 способами:

Мы создадим 2 класса, с разными видами решений, чтобы переключиться между ними, нужно выбрать нужный класс в файле AndroidManifest.

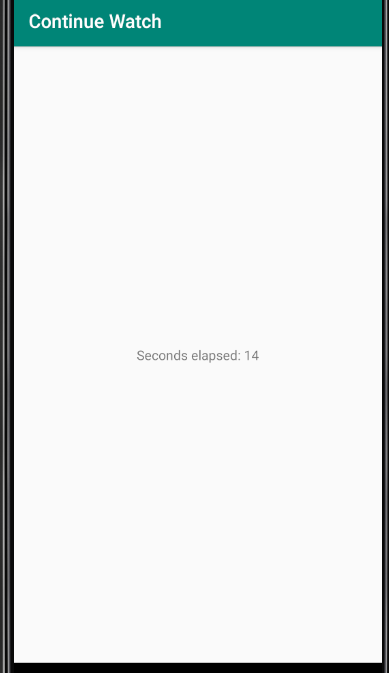
1. onSaveInstanceState / onRestoreInstanceState

Изображение выглядит как текст

Автоматически созданное описание

Здесь все понятно, мы сохраняем в Bundle, который является ассоциативной памятью, переменную secondsElapsed, и при создании приложения вызывается метод onRestoreInstanceState, который восстанавливает значение переменной.

Покажем правильный вариант работы приложения.

  Изображение выглядит как текст

Автоматически созданное описание

Мы видим, что приложение считало секунды, далее ушло в background, далее вышло из него и мы повернули экран, посмотрим на логи.

Изображение выглядит как текст

Автоматически созданное описание

Из логов мы видим, что поскольку мы 2 раза вошли в состояние paused, то и 2 раза мы уничтожали и создавали новый поток.

1. SharedPreferences

Изображение выглядит как текст

Автоматически созданное описание

C помощью sharedPreferences в onStop() мы обращаемся к некому хранилище( с именем, которое лежит под ресурсом R.string.storage), далее мы наполняем это хранилище, а в onStart() мы опять обращаемся к хранилищу и получаем требуемую переменную.

**Отличия в решениях:**

Главное отличие в работах решений является то, что при перезапуске приложения с sharedPreferences секунды, посчитанные ранее сохраняются, когда в варианте с onSaveInstanceState счет начинается с 0.

**Вывод:**

Мною были успешно выполнены все необходимые задания, также для их выполнения пришлось почитать немалое кол-во документации и сайтов, поэтому я считаю, что я изучил жизненный цикл андроид приложений и изучил основные возможности сохранения состояния приложения и alteranative resources.