МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»



**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

по выполнению лабораторных работ

по дисциплине «Программирование мобильных устройств»

для студентов направлений 09.03.03 «Прикладная информатика» и 09.03.02 «Информационные системы и технологии».

Ставрополь

2021

# **Лабораторная работа 6. Таймеры и секундомеры. Логирование.**

**Цель работы:** Научиться использовать отладочную информацию

**Формируемые компетенции:** ПК-7

**Теоретическая часть:**

Важным инструментом разработчика является отладка приложения. Пока Вы не создали приложение в идеальном виде, оно может иногда «вылетать», «обзываться» или вообще даже не запускаться. Как уже опытные программисты Вы должны знать, что такое средства отладки (точки останова, шаги с обходом и тд). При работе с AndroidStudio есть несколько нюансов. Самый главный – приложение в режиме отладки нужно запускать сразу. Что это значит: в Visual Studio Вы можете поставить точку останова когда приспичит, и она сразу же остановит приложение, когда до неё дойдёт очередь. В Android Studio же Вы должны сразу запускать приложение в режиме отладки, чтобы можно было расставлять точки останова и работать с ними. Для того чтобы запустить приложение в этом режиме, необходимо найти кнопку Debug:



После этого можно смело расставлять точки останова в любом месте приложения.

Тем не менее, точки останова не всегда удобно использовать. Бывают ситуации, когда нужные Вам данные «спрятаны» внутри объекта, который в свою очередь тоже является частью другого объекта. В таком случае, проще всего использовать Логи (журнал). В них Вы можете записывать всё, что Вам нужно: от обычной информации о переменных и объектах до отображения ошибок.

Естественно, данные Логи не предназначены для «простых смертных», поэтому, добраться к ним может только программист.

Вообще, Логи делятся на 5 основных типов:

* Info – простое информационное сообщение;
* Warning – предупреждение – ещё не ошибка, но уже стоит действовать аккуратно
* Error – ошибка – что-то в процессе выполнения пошло не так, поэтому, нужно предупредить самого себя;
* Debug – отладка – сообщения, связанные с отладкой приложения;
* Verbose – Огромный лог для подробного описания того, что произошло.

Для того чтобы записать что-то в журнал, нужно использовать соответствующий метод. В Android Studio для каждого из перечисленных типов Логов есть свой метод, который называется первой буквой определяющего слова: Например, Log.i() – для информационных сообщений, Log.e() для ошибок и тд.

Для каждого из методов есть 2 варианта, но в обоих Вы можете использовать теги для их быстрого поиска в журнале:

* С ошибкой (то есть, как только запишется Ваше сообщение в Лог, то приложение выдаст ошибку);
* Без ошибки.



Первым параметром идёт тэг. Он нужен для того, чтобы быстро найти нужное сообщение в журнале. Второй – непосредственно текст сообщения. Здесь можно передать какое-нибудь строковое значение переменной.

Чтобы посмотреть журнал во время работы Вашего приложения, можно найти панель, которая называется Logcat. Чтобы её открыть, нужно выбрать View→ToolWindows→Logcat. Использовать данные Логи Вы сможете при выполнении следующих заданий.

Порой возникает необходимость выполнить какое-то действие спустя некоторое время. Или же наоборот начать отслеживание выполнения действия, чтобы потом посчитать, сколько времени ушло на его выполнение. Разберёмся по порядку.

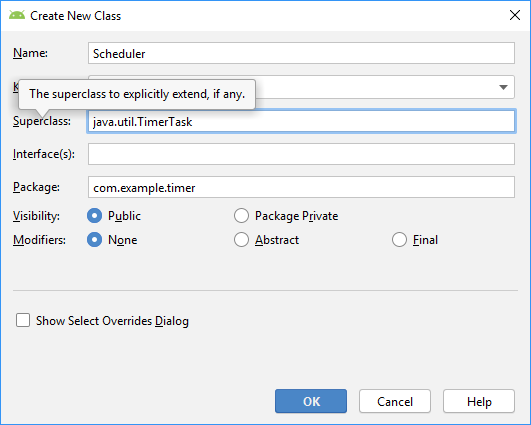
# **Таймер**

Таймеры состоят из двух частей. Первое – то, что должно выполниться. То есть, действия, которые должны совершиться по истечении какого-то времени. Второе – это непосредственно время, спустя которое должно выполниться действие.

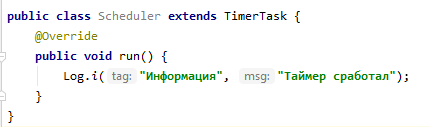
TimerTask – относится к первой части программирования таймера. Именно этот класс отвечает за выполнение действия. Timer – вторая часть. По странному стечению обстоятельств, таймер является частью самого себя.

Рассмотрим простой пример:

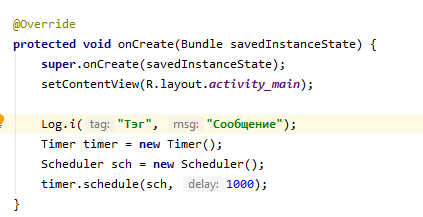
1. Создаём новое приложение.
2. В папке Java создаём новый класс:



1. Добавляем внутри нашего класса переопределённый метод run.
2. Метод должен содержать следующий код:



1. В MainActivity создаём Экземпляр класса Timer, который будет работать по нашему планировщику.



1. Запустим приложение и убедимся, что код выполнился спустя какое-то время. Откройте вкладку logcat в приложении и убедитесь, что там есть запись, которую мы добавили.



# **Секундомер.**

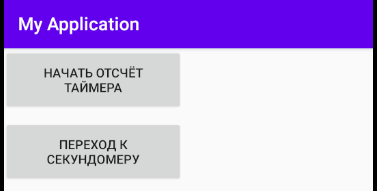
Для реализации секундомера можно использовать два способа:

* Встроенный класс Chronometer;
* Timer.

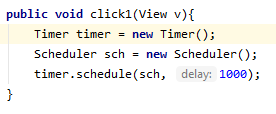
Самый простой способ – использовать Chronometer. Зачем изобретать велосипед, если он уже есть. Рассмотрим следующий пример: Нам нужно написать простой секундомер с кнопками «Запуск», «Стоп» и «Очистить». Каждую секунду отображать в TextView текущее значение секундомера.

1. Создадим второе окно в нашем приложении и добавим две кнопки на главной странице:

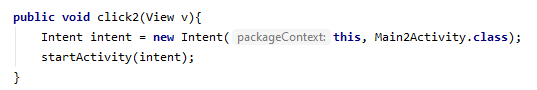
* Начать отсчёт таймера.
* Переход к секундомеру.



1. Переместим код таймера в обработчик нажатия по первой кнопке.



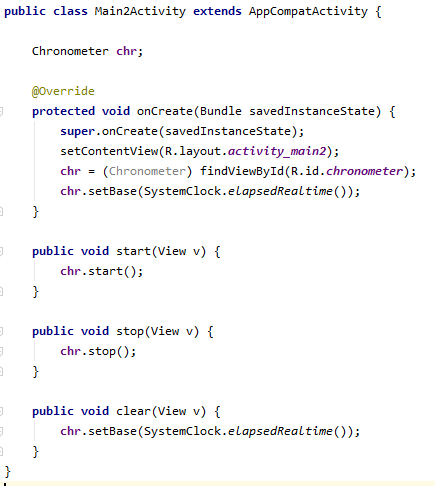
1. Создадим ещё один обработчик для перехода на другое окно с секундомером.



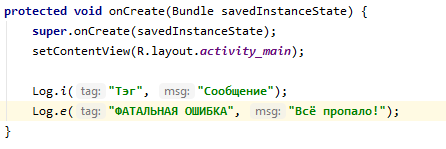
1. Создадим следующую разметку на странице секундомера.

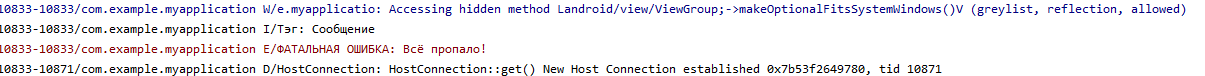


1. И напишем обработчики всех кликов:



Во время работы Вашего приложения оно может непроизвольно «вылетать». Это происходит из-за критической ошибки. Чтобы узнать, что за ошибка привела к этому, нужно снова обратиться к Logcat. Давайте, создадим такую ошибку и попробуем её найти в журнале:





Критические ошибки обычно хранят информацию обо всех файлах, которые её вызвали, поэтому, найти их в журнале достаточно просто: будет много красного текста. Чтобы правильно использовать логирование ошибок, необходимо помещать их в конструкцию try catch.

# **Задание**

С помощью таймера и логов решить задачу в соответствии с вариантом:

1. С шагом 3 посчитать сумму всех простых чисел в диапазоне от 0 до 100. Каждый шаг должен выполняться в новом «тике». Вывести результат в журнале.
2. Каждую секунду в течение минуты отображать в логах следующий элемент из последовательности Фибоначчи, начиная с 1.
3. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):
4. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):
5. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):
6. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):
7. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):
8. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):
9. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):
10. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):
11. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):
12. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):
13. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):
14. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):
15. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):
16. После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

**Содержание отчета:** отчет по лабораторной работе должен быть выполнен в редакторе MSWord и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчета приводятся выводы о проделанной работе.

В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

**Контрольные вопросы:**

1. Таймер.
2. Chronometer.
3. Виды и описание логов.

**Список литературы, рекомендуемый к использованию по данной теме:**

1. Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. Основы интернет-технологий: учебное пособие / Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. – М.: Архангельск: ИПЦ САФУ, 2013 – 366 с.; То же [Электронный ресурс]. - URL:http://biblioclub.ru/index.php?page=book\_red&id=436379&sr=1
2. Соколова В. В. Разработка мобильных приложений: учебное пособие/ Соколова В. В. Томск: Издательство Томского политехнического университета, 2015 – 176с. То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book\_red&id=442808&sr=1