**Лабораторная работа №2. Жизненный цикл. Сохранение состояния.**

**Custom View**

**Жизненный цикл активности**

В процессе работы каждая активность может проходить через несколько этапов жизненного цикла: создание (create), запуск (start), приостановка(pause), возобновление (resume), остановка (stop) и разрушение (destroy).

Создайте новый проект. Перегрузите методы (OnCreate, OnStart, OnResume, OnPause, OnStop, OnDestroy, OnSaveInstanceState) у MainActivity. Для этого введите в классе ключевое слово override и нажмите пробел – среда покажет список методов, которые можно переопределить в данном классе:



Добавьте логирование этих событий следующим образом:



Среда по умолчанию не знает о классе Log, поэтому в список директив using нужно включить пространство имен Android.Util:



Запустите приложение. Чтобы увидеть результат логирования, нужно открыть окно Вид → Другие окна → Android Device Logging... По умолчанию лог может содержать много строк, чтобы увидеть нужные, следует отфильтровать их по тегу, указанному в первом параметре команды логирования:



**Сохранение состояния программы**

Перейдите в вёрстку MainActivity (activity\_main.xml). Добавьте под кнопкой элемент ScrollView с вертикальным LinearLayout (id=container) внутри.

В методе OnCreate у MainActivity добавим обработку нажатия на кнопку:



В методе OnCreate у кнопки подключается обработчик нажатия Button\_Click. При нажатии кнопки создаётся новое текстовое поле, в него помещается очередное значение счётчика, и оно добавляется в контейнер. Однако, если повернуть устройство из вертикальной ориентации в горизонтальную или наоборот, то текст пропадает.

Чтобы исправить этот недочёт, добавим код сохранения переменной viewsCount в методе OnSaveInstanceState:



А в методе OnCreate восстановим состояние этой переменной:



}

Запустите приложение. Теперь при повороте экрана все добавленные TextView будут сохраняться. Тем не менее, позиция ScrollView сохраняться не будет.

**Custom View**

Для объектно-ориентированного программирования более логичным может быть подход, при котором сохранение и восстановление данных поручается самим объектам. Для этого можно создать дочерний класс от View, и инкапсулировать в него весь необходимый функционал. Такие классы с дополненным функционалом называются Custom View.

Создайте новый класс с названием ViewsContainer, унаследуйте его от LinearLayout. Этот класс будет включать в себя все необходимые инструменты для добавления текстовых полей, а также сохранения и восстановление их при смене ориентации экрана устройства.

Добавьте в класс метод IncrementViews, аналогичный методу AddTextView из MainActivity:



Создайте также новый класс SavedState, унаследованный от BaseSavedState, который будет использоваться для хранения состояния данных:



}

Переопределите методы OnRestoreState и OnSaveState у класса ViewsContainer следующим образом:



Перейдите в текстовое представление activity\_main.xml и замените внешний элемент LinearLayout на ViewsContainer. В MainActivity уберите всё связанное с сохранением состояния и метод AddTextView. Должно получится следующее:



Запустите приложение, убедитесь, что состояние сохраняется при повороте экрана.

**Задание**

Реализуйте описанный Custom View. Добавьте на активность результат в разных состояниях. При повороте экрана состояния должны сохраняться.



При тапе по любой точке View, CheckBox должен переключаться. Если картинки нет, текст должен сдвигаться влево. Если подзаголовка нет, заголовок должен быть отцентрирован по вертикали. При увеличении высоты View, CheckBox должен оставаться на месте.

**Оформление отчета**

Отчет должен состоять из поэтапного описания выполняемых действий. Отдельно представляется листинг и скрины, работающей программы. Отчет должен содержать титульный лист. Текст отчета должен быть оформлен по требованиям, предъявляемым к курсовым и ВКР. Каждый отчет защищается студентом индивидуально.