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 2. Разработка дизайна страницы с помощью языка XML. Менеджеры размещения.**

**Цель работы:** Изучение XML разметки

**Формируемые компетенции:** ПК-7

**Теоретическая часть:**

На прошлой лабораторной работе мы рассмотрели объектную структуру главной страницы. Мы научились добавлять на страницу графические примитивы и даже управлять ими. Таким же образом возможно добавлять на страницу элементы управления: кнопки, галочки, текстовые блоки и т.д.

Очевидно, что проектировать дизайн приложения таким образом не самый лучший вариант. Поэтому используется язык разметки XML, который позволяет в гораздо более удобном виде управлять внешним видом страницы.

Очень важно понять, что язык XML не просто язык разметки, он позволяет декларативно создавать объекты и определять их свойства, вместо того чтобы делать это в коде, как мы делали на предыдущем занятии.

Рассмотрим код XML, созданный по умолчанию AS. Весь код состоит из отдельных элементов-тегов, заключенных в угловые скобки. Давайте разберем, что представляет собой тег.

<имя\_тега имя\_атрибута=”значение атрибута”>

Содержимое тега

</имя\_тега>

Содержимое тега представляет собой либо текст, либо вложенные теги:

<имя\_тега имя\_атрибута=”значение атрибута”>

<имя\_вложенного\_тега>

содержимое вложенного тега

</имя\_вложенного\_тега>

</имя\_тега>

Таким образом, XML код – набор вложенных тегов. Каждый тег позволяет описывать декларативным (описательным) способом классы и объекты, участвующие в приложении. Например, в корневом теге файла activity\_main.xml описывается класс главной страницы MainActivity.java. Рассмотрим этот момент подробнее.

Именем корневого тега является тег RelativeLayout (В новых версиях Android Studio появился ConstraintLayout, который стоит заменить на RelativeLayout для выполнения данной лабораторной работы).

ReiativeLayout (относительная разметка) позволяет дочерним компонентам определять свою позицию относительно родительского компонента или относительно соседних дочерних элементов (по идентификатору элемента). В RelativeLayout дочерние элементы расположены так, что, если первый элемент расположен по центру экрана, другие элементы, выровненные относительно первого элемента, будут выровнены относительно центра экрана.

*android:id="@+id/activity\_main"*

В этом свойстве указывается идентификатор разрабатываемой страницы.

Атрибуты с префиксом xmlns используются для описания используемых пространств имен. Обратите внимание на атрибут

*xmlns:**android="http://schemas.android.com/apk/res/android"*

он создает псевдоним android для всего пространства имен, которое используется в имени корневого тега RelativeLayout.

Получается, что в корневом теге activity\_main.xml идет речь о том же классе, что и в файле MainActivity.java. Более того с помощью XML вы можете разрабатывать класс страницы одновременно с разработкой его в исходном коде в файле MainActivity.java.

Остальные атрибуты корневого тега рассмотрим на дальнейших занятиях. Важное замечание: как и в любом XMLдокументе, здесь может существовать только один корневой элемент, таким образом, вы можете описать в одном файле только один класс страницы, что вполне логично.

Следующий важный момент: содержимое корневого тега соответствует свойству страницы Content, которое мы использовали на предыдущей лабораторной работе. Каждый вложенный тег описывает какой-либо объект, аналогично тому, как мы создавали объекты в исходном коде с помощью конструктора.

Найдите среди вложенных тегов следующий:

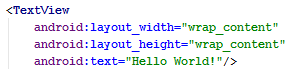


Рисунок 1 - Виджет TextView

Этот тег не является прямым потомком корневого тега, и это логично, ведь свойство Content страницы не может хранить более одного объекта (вспоминаем лаб.1). Он находится в контейнере RelativeLayout. Об этих контейнерах и вложенностях мы поговорим далее, вспомним лишь, что в лаб. 1 мы использовали контейнер Canvas (холст).

Основные элементы, которые можно добавлять на страницу были перечислены в первой лабораторной работе. Однако, нам понадобится ещё один виджет – ImageView. Данный элемент позволяет размещать изображения/примитивы внутри себя. По сути, ImageView является прямоугольником, поэтому, в случае чего, ему можно поменять цвет фона и даже цвет примитива (свойства background и tint соответственно).

Рассмотрим основные параметры, которые можно применять к элементам разметки XML:

| **Название** | **К каким элементам применять** | **Краткое описание** | **Примеры** |
| --- | --- | --- | --- |
| android:id | Все | Устанавливает уникальный идентификатор элемента, чтобы можно было найти его в java-коде | @+id/el1,  @+id/textView |
| android:layout\_width | Все | Устанавливает ширину элемента | 10px, 18dp, match\_parent (занимает всё доступное пространство родителя), wrap\_content (устанавливает размеры в зависимости от содержимого) |
| android:layout\_height | Все | Устанавливает высоту элемента |
| android:layout-margin | Все | Устанавливает внешние отступы элемента относительно других | 10dp, 150px, 200mm, 14pt |
| android:padding | Все | Устанавливает внутренний отступ у содержимого |
| android:background | Все | Устанавливает цвет задней поверхности элемента | #fff  @color/colorPrimary |
| android:gravity | Все | Устанавливает выравнивание содержимого элемента по одной из сторон | left, right, center, bottom, top...  Можно комбинировать:  left|center |
| android:rotation | Все | Поворачивает элемент на указанное число градусов | 90, 45, 180 |
| android:text | TextView, EditText | Устанавливает текст в элементе | Любая строка |
| android:textSize | Устанавливает размер текста | Принято указывать размер в единицах sp. |
| android:visibility | Все | Скрывает или отображает элемент | Visible, invisible(просто скрывает), gone(удаляет, как будто бы и не было элемента на этом месте) |
| android:src | ImageView | Устанавливает в качестве источника картинка внешний файл или графический примитив | @drawable/ic\_launcher\_foreground |

**Контейнеры LinearLayout и Grid.**

Мы уже говорили о том, что для размещения элементов на странице, необходимо использовать контейнеры.

Термин «контейнер», мы использовали в контексте того, данный элемента может содержать в себе другие элементы. На самом деле миссия у такого «контейнера» гораздо выше. Заключается она в определении схемы размещения и компоновки других элементов. Поэтому такие «контейнеры» часто называют менеджерами размещения.

Мы использовали контейнер типа «Холст», который позволяет размещать элементы произвольным образом с помощью присоединенных свойств (attached properties).

Такой подход называется абсолютным позиционированием, т.е. позиционирование по абсолютным координатам(относительно верхнего левого края экрана). С одной стороны, это дает бесконечную гибкость в расположении элементов на странице, с другой же значительно повышает трудоемкость размещения, т.к. вам необходимо рассчитывать координаты для каждого элемента. Поэтому менеджер размещения Canvas используется только в исключительных случаях, например, рисования произвольных геометрических фигур (лаб. 1).

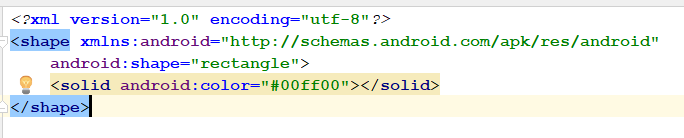
В случае разработки бизнес-приложений, используются стандартные элементы управления (кнопки, текстовые блоки, галочки и т.д.). Для их размещения чаще всего достаточно использовать простые макеты размещения, например, размещение одного элемента за другим горизонтально или вертикально. Для такого размещения используется менеджер LinearLayout.

Всего существует 5 основных менеджеров:

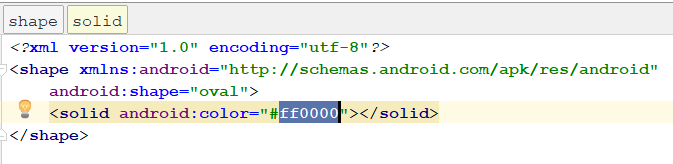
* LinearLayout (один за одним);
* RelativeLayout (располагать элементы относительно друг друга);
* ConstraintLayout (более совершенная версия RelativeLayout);
* GridLayout (сетка, позволяет помещать элементы в "ячейки");
* TableLayout (таблица, отличается от GridLayout тем, что необходимо задавать элементы в каждой ячейке);

**Ход лабораторной работы**

1. Создайте новый проект с произвольным именем.
2. Создадим два Drawable-файла с названиями “rectangle” и “circle”. Заменим исходный код:



Листинг 1 - Содержимое файла rectangle.xml



Листинг 2 - Содержимое файла circle.xml

В наш xml файл добавим LinearLayout и заполним содержимым



Листинг 3 - Содержимое файла Activity\_main.xml

Результат действий представлен на рисунке1.

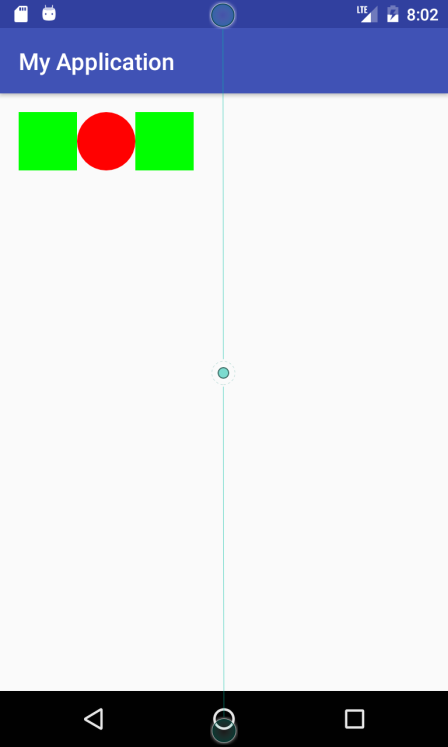


Рисунок 2 - Отображение результата на эмуляторе

1. По умолчанию LinearLayout размещает элементы горизонтально один за другим. Такое поведение можно изменить с помощью свойства android:orientation="vertical":

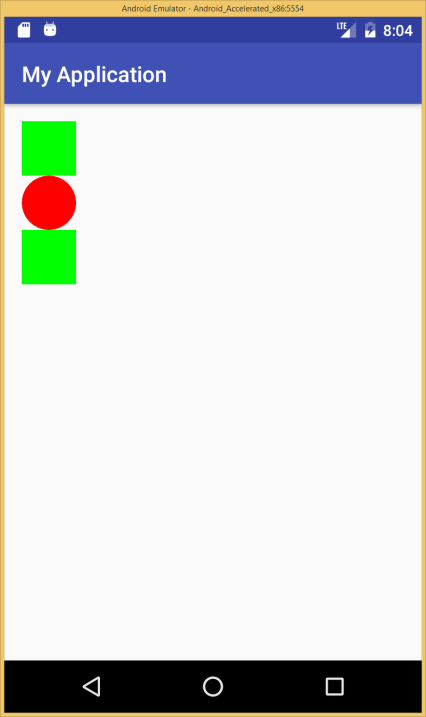


Рисунок 3 - Использование вертикальной ориентации

1. При использовании горизонтальной ориентации также можно изменить направление расположение элементов, «прилепив» их к правой границе экрана с помощью свойства layoutDirection:



Рисунок 4 - Пример использования RightToLeft направления

1. Менеджеры могут размещаться один в другом для более изощренных видов разметки. Например:



Листинг 5- Содержимое файла activity\_main.xml

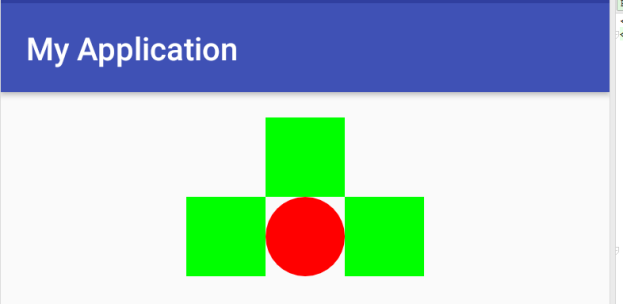


Рисунок 5 - Результат листинга 5

Не стоит переживать, что Android Studio выделяет область, где используются размеры коричневым цветом. Дело в том, что размеры у разных телефонов разные, следовательно, одни те же элементы на различных устройствах будут выглядеть по-разному. Бывают моменты, когда необходимо, чтобы элемент оставался всегда одинакового размера на разных устройствах, а иногда наоборот, чтобы подстраивалось под размеры экрана. Для первого случая следует указывать размеры в px, а во втором в dp.

**Менеджер размещения Grid**

Позволяет располагать элементы, используя строки и столбцы. При использовании менеджера размещения Grid разработчик определяет общую структуру сетки, а потом, используя присоединенные свойства, размещает элементы управления в ячейках сетки.

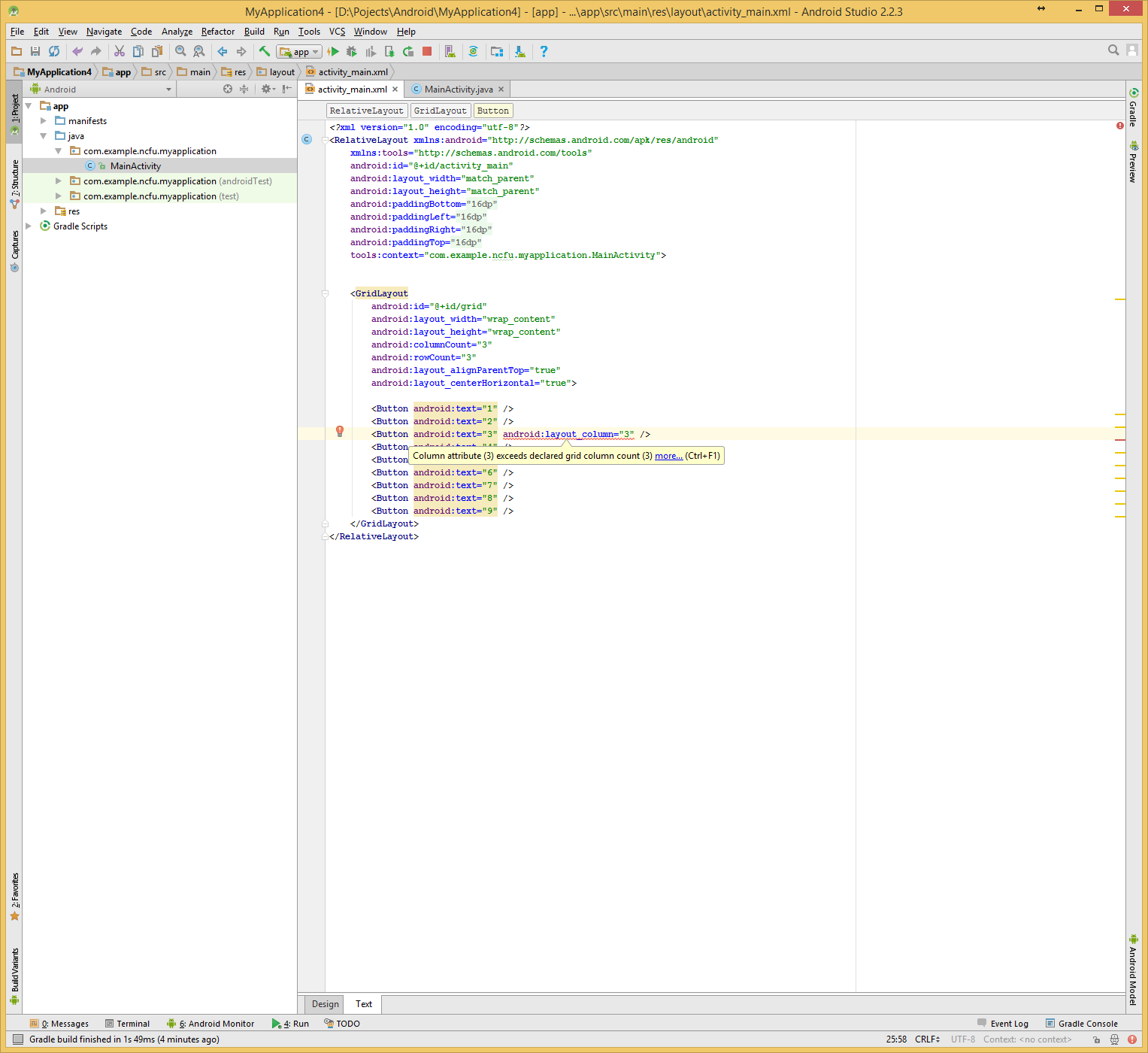


Рисунок 6 - Пример использования Grid-а

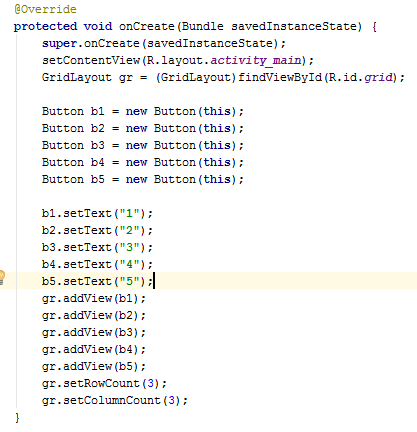
Обратите внимание на следующие моменты:

* 1. Размеры могут задаваться либо целыми числами, либо строкой «wrap\_content», указывающей на то, что размер строки или столбца должен подстроиться под размер элемента в нем находящемся, либо строкой «match\_parent», указывающей на то, что строка или столбец займет все оставшееся место на странице.
  2. После определения структуры располагаются размещаемые на экране элементы. В данном случае это объекты типа Button (кнопка).
  3. Определяя дочерним элементам ту или иную строку (столбец) следует помнить, что не нужно выходить за границу определяемого column\_count и row\_count. Например, если grid содержит всего 3 столбца, мы не сможем поместить его в колонку с номером 3 (индексы отсчитываются с 0 элемента, как в программировании).

Чтобы явно указать, в какую ячейку поместить объект, используются свойства layout\_column (столбец) и layout\_row (строка). Так как grid является подобием таблицы, присутствует возможность объединять ячейки. Для объединения строк используется layout\_rowSpan, где передаётся количество ячеек для объединения, а для столбцов соответственно layout\_columnSpan.



Листинг 6 - Пример неправильного обращения к индексам



Листинг 7 - Пример создания и использования Grid-a, используя java



Рисунок 7 - Пример объединения ячеек

**Задания:** для выполнения лабораторной работы необходимо выполнить следующее:

1. Реализуйте размещение элементов, как представлено на рисунке 5.
2. Нарисуйте на экране композицию, представленную на рисунке 8:

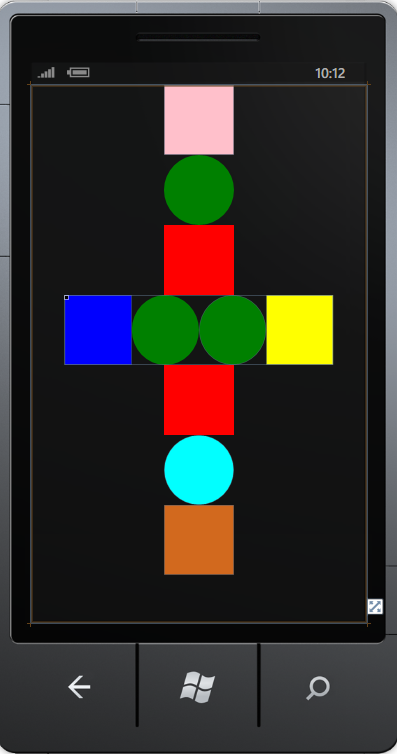


Рисунок 8 - Задание для проверки способностей

1. Выполните задание по варианту с помощью XML. При выполнении задания лабораторной работы необходимо использовать менеджеры разметки Grid и LinearLayout.

**Варианты для задания 3:**

1. Рисунок 9.
2. Рисунок 10.
3. Рисунок 11.
4. Рисунок 12.
5. Рисунок 13.
6. Рисунок 14.
7. Рисунок 15.
8. С помощью менеджеров размещения напишите букву Н с помощью кнопок на весь экран
9. С помощью менеджеров размещения выведите букву П из кнопок на весь экран
10. С помощью менеджеров размещения выведите букву Г из кнопок на весь экран

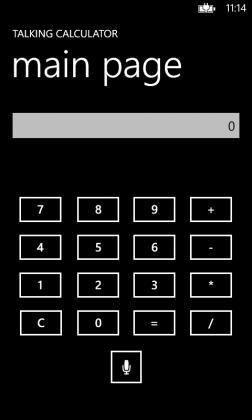


Рисунок 9

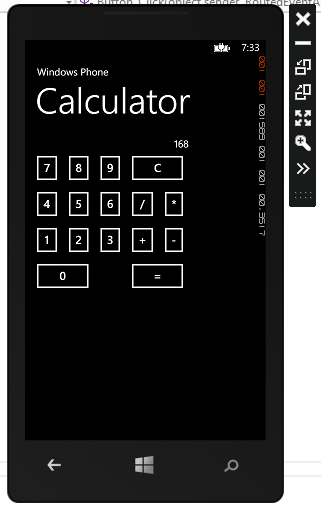


Рисунок 10

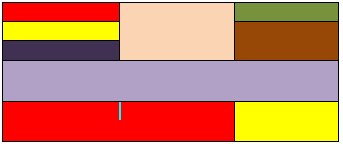


Рисунок 11

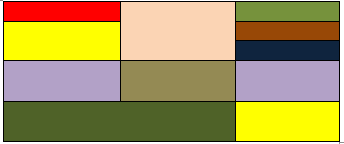


Рисунок 12

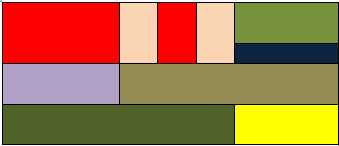


Рисунок 13

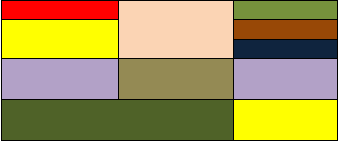


Рисунок 14



Рисунок 15

**Оборудование и материалы:** для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 (8,8.1,10) и программными продуктами: AndroidStudio.

**Указания по технике безопасности:** к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

**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 выровнен 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. Язык XML
2. Тег
3. Менеджер размещения
4. Grid
5. LinearLayout

**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