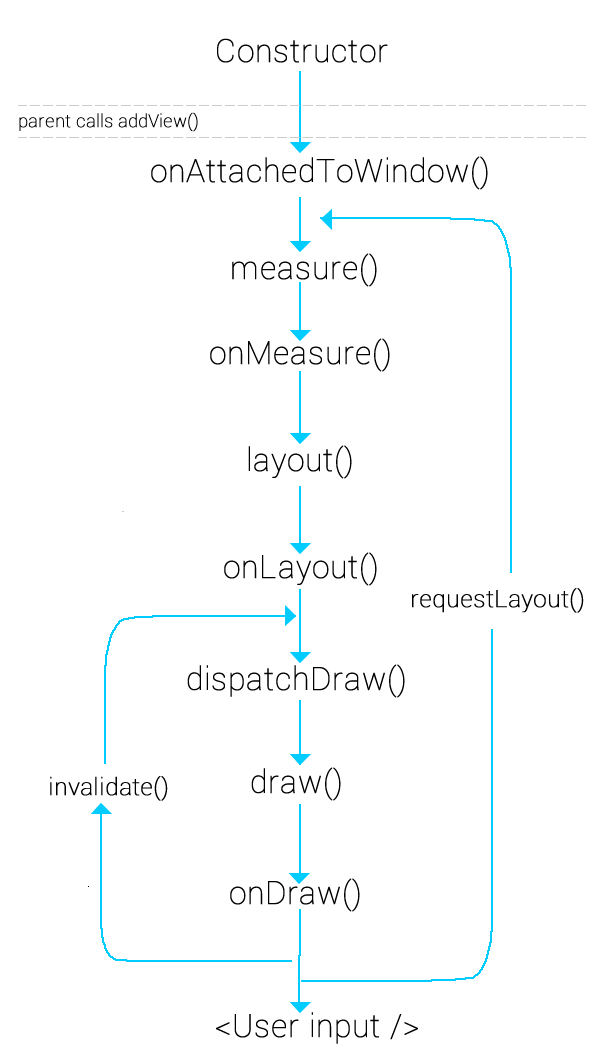
11. View - deep diving, Animations

Три причины писать свою вью:

1. Что то совсем уникальное
2. Компонент встречается много раз в приложении и состоит из базовых
3. Необходим дополнительный функционал стандартного компонента

## Жизненный цикл view



Here's a summary of some of the other standard methods that the framework calls on views:

|  |  |  |
| --- | --- | --- |
| Category | Methods | Description |
| Creation | Constructors | There is a form of the constructor that are called when the view is created from code and a form that is called when the view is inflated from a layout file. The second form should parse and apply any attributes defined in the layout file. |
| [onFinishInflate()](https://developer.android.com/reference/android/view/View.html#onFinishInflate()) | Called after a view and all of its children has been inflated from XML. |
| Layout | [onMeasure(int, int)](https://developer.android.com/reference/android/view/View.html#onMeasure(int,%20int)) | Called to determine the size requirements for this view and all of its children. |
| [onLayout(boolean, int,int, int, int)](https://developer.android.com/reference/android/view/View.html#onLayout(boolean,%20int,%20int,%20int,%20int)) | Called when this view should assign a size and position to all of its children. |
| [onSizeChanged(int, int,int, int)](https://developer.android.com/reference/android/view/View.html#onSizeChanged(int,%20int,%20int,%20int)) | Called when the size of this view has changed. |
| Drawing | [onDraw(Canvas)](https://developer.android.com/reference/android/view/View.html#onDraw(android.graphics.Canvas)) | Called when the view should render its content. |
| Event processing | [onKeyDown(int, KeyEvent)](https://developer.android.com/reference/android/view/View.html#onKeyDown(int,%20android.view.KeyEvent)) | Called when a new key event occurs. |
| [onKeyUp(int, KeyEvent)](https://developer.android.com/reference/android/view/View.html#onKeyUp(int,%20android.view.KeyEvent)) | Called when a key up event occurs. |
| [onTrackballEvent(MotionEvent)](https://developer.android.com/reference/android/view/View.html#onTrackballEvent(android.view.MotionEvent)) | Called when a trackball motion event occurs. |
| [onTouchEvent(MotionEvent)](https://developer.android.com/reference/android/view/View.html#onTouchEvent(android.view.MotionEvent)) | Called when a touch screen motion event occurs. |
| Focus | [onFocusChanged(boolean,int, Rect)](https://developer.android.com/reference/android/view/View.html#onFocusChanged(boolean,%20int,%20android.graphics.Rect)) | Called when the view gains or loses focus. |
| [onWindowFocusChanged(boolean)](https://developer.android.com/reference/android/view/View.html#onWindowFocusChanged(boolean)) | Called when the window containing the view gains or loses focus. |
| Attaching | [onAttachedToWindow()](https://developer.android.com/reference/android/view/View.html#onAttachedToWindow()) | Called when the view is attached to a window. |
| [onDetachedFromWindow()](https://developer.android.com/reference/android/view/View.html#onDetachedFromWindow()) | Called when the view is detached from its window. |
| [onWindowVisibilityChanged(int)](https://developer.android.com/reference/android/view/View.html#onWindowVisibilityChanged(int)) | Called when the visibility of the window containing the view has changed. |

## Custom attributes

To define custom attributes, add <declare-styleable> resources to your project. It's customary to put these resources into a res/values/attrs.xml file. Here's an example of an attrs.xml file:

<resources>

<declare-styleable name="PieChart">

<attr name="showText" format="boolean" />

<attr name="labelPosition" format="enum">

<enum name="left" value="0"/>

<enum name="right" value="1"/>

</attr>

</declare-styleable>

</resources>

## Override onDraw()

The [android.graphics](https://developer.android.com/reference/android/graphics/package-summary.html) framework divides drawing into two areas:

* *What* to draw, handled by [Canvas](https://developer.android.com/reference/android/graphics/Canvas.html)
* *How* to draw, handled by [Paint](https://developer.android.com/reference/android/graphics/Paint.html).

## Handle Layout Events

Although [View](https://developer.android.com/reference/android/view/View.html) has many methods for handling measurement, most of them do not need to be overridden. If your view doesn't need special control over its size, you only need to override one method: [onSizeChanged()](https://developer.android.com/reference/android/view/View.html#onSizeChanged(int,%20int,%20int,%20int)).

[onSizeChanged()](https://developer.android.com/reference/android/view/View.html#onSizeChanged(int,%20int,%20int,%20int)) is called when your view is first assigned a size, and again if the size of your view changes for any reason.

If you need finer control over your view's layout parameters, implement [onMeasure()](https://developer.android.com/reference/android/view/View.html#onMeasure(int,%20int)).

<https://stackoverflow.com/questions/12266899/onmeasure-custom-view-explanation>

## Android animations

## Property animation

animation = ValueAnimator.*ofInt*(0, getWidth());

animation.setDuration(1000);

animation.setRepeatMode(ValueAnimator.*REVERSE*);

animation.setRepeatCount(ValueAnimator.*INFINITE*);

animation.start();

Показать на примерах анимации.

## Object Animator

* translationX and translationY: These properties control where the View is located as a delta from its left and top coordinates which are set by its layout container.
* rotation, rotationX, and rotationY: These properties control the rotation in 2D (rotation property) and 3D around the pivot point.
* scaleX and scaleY: These properties control the 2D scaling of a View around its pivot point.
* pivotX and pivotY: These properties control the location of the pivot point, around which the rotation and scaling transforms occur. By default, the pivot point is located at the center of the object.
* x and y: These are simple utility properties to describe the final location of the View in its container, as a sum of the left and top values and translationX and translationY values.
* alpha: Represents the alpha transparency on the View. This value is 1 (opaque) by default, with a value of 0 representing full transparency (not visible).

Показать на примерах анимации.

## Declare animations in XML

The following property animation classes have XML declaration support with the following XML tags:

* [ValueAnimator](https://developer.android.com/reference/android/animation/ValueAnimator.html) - <animator>
* [ObjectAnimator](https://developer.android.com/reference/android/animation/ObjectAnimator.html) - <objectAnimator>
* [AnimatorSet](https://developer.android.com/reference/android/animation/AnimatorSet.html) - <set>

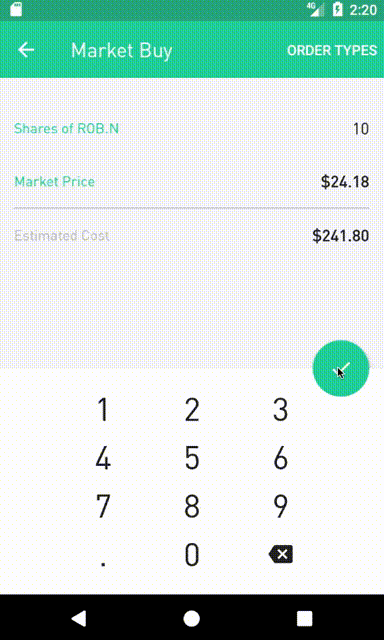
## Анимации, основанные на физике

<https://proandroiddev.com/introduction-to-physics-based-animations-in-android-1be27e468835>

## Transitions Framework

<https://habr.com/post/243363/> Рассказать?

## Constraint animations

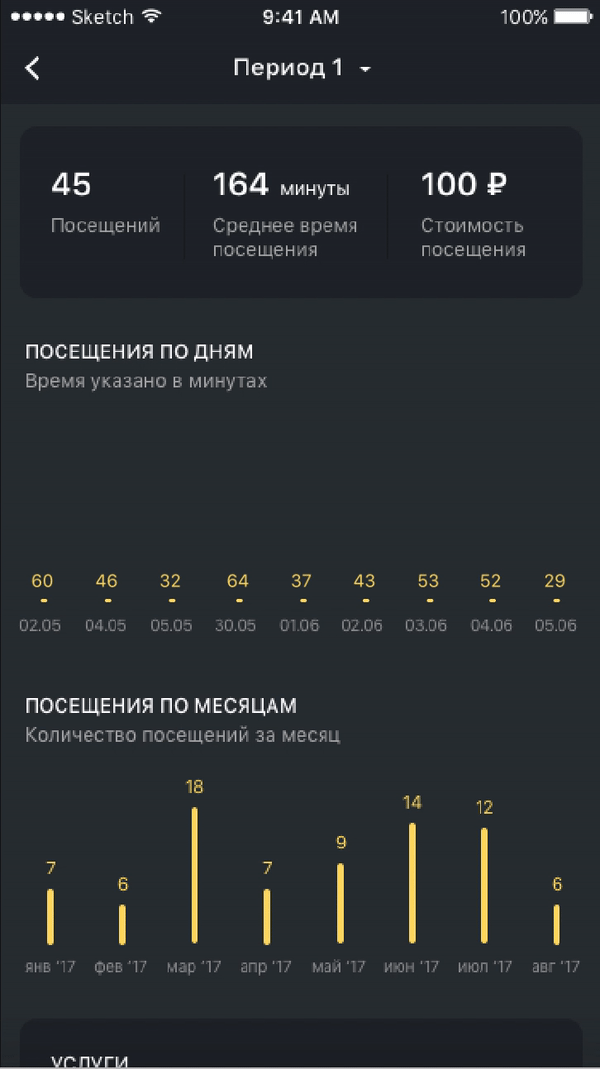


#### Задание:

В зеплине нужно найти урок 11. В задании нужно реализовать и анимировать только вот этот элемент:



Анимация должна выглядеть вот так:



Подробности:

Необходимо реализовать свою view. Для этого нужно создать класс, унаследованный от View.

Необходимо переопределить и реализовать основные методы - констуктор, onMeasure, onDraw.

Данные для столбцов задаются публичным методом. Должно поддерживаться динамическое количество столбцов - от 1 до 9. Столбцы должны равномерно распределяться по ширине экрана.

Должно быть 2 публичных метода - setData() и startMyAnimation().  
Метод animate() вызывать при нажатии в любое место этой вью.

Вью должна отрисовывать даты и значения в этих датах. В качестве дат брать любой интервал от текущего дня в прошлое, в качестве значений - случайно сгенерированное число.

Цвет текста дат и цвет столбцов должен задаваться через xml файл разметки.

Надписи “посещение по дням” и “время указано в минутах” делать не нужно.  
  
Итого - должна быть вью с столбцами, которая может анимироваться по нажатию на эту вью.

#### Полезные ссылки:

1. <https://developer.android.com/training/custom-views/create-view> - официальная документация по созданию view
2. Mastering onMeasure <https://medium.com/@quiro91/custom-view-mastering-onmeasure-a0a0bb11784d>
3. Старт андроид про рисование <https://startandroid.ru/ru/uroki/vse-uroki-spiskom/311-urok-141-risovanie-dostup-k-canvas.html>
4. <https://developer.android.com/guide/topics/graphics/prop-animation> - анимации
5. <https://proandroiddev.com/introduction-to-physics-based-animations-in-android-1be27e468835> анимации, основанные на физике