Рендеринг текста в Android

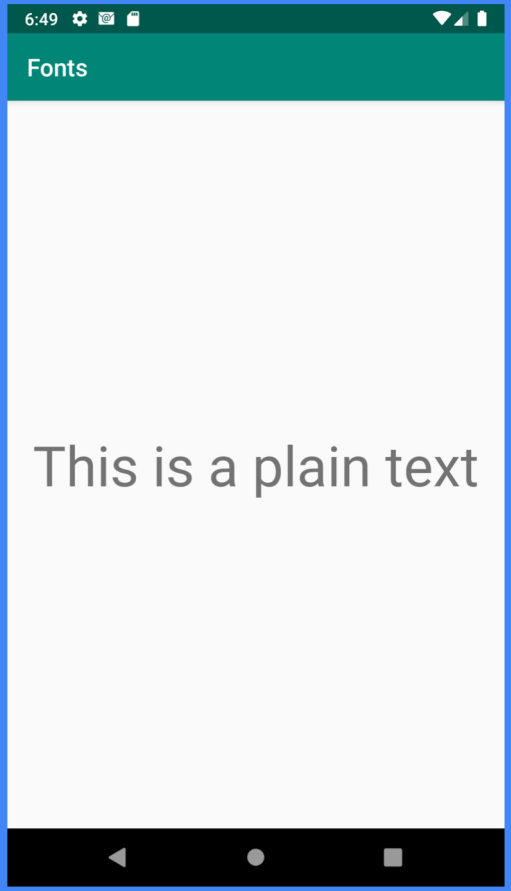
Контекст

- Кто такой?
- Чем занимаюсь?
- Зачем это все?

Как отображается текст?

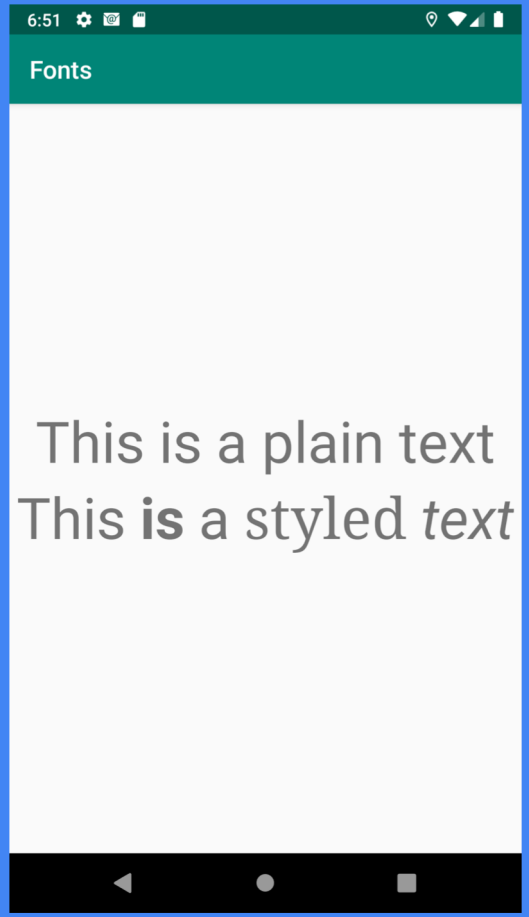
Как отображается текст?

TextView



Plain text

```
<TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="46sp"
        android:text="This is a plain text"
        />
```



Styled text

```
String string = getString(R.string.html_text);

((TextView) findViewById(R.id.textViewHtml))
        .setText(Html.fromHtml(string));
```

Html.fromHtml(string)

```
<string name="html_text">
  <![CDATA[This <b>is</b> a <font face="serif">styled</font> <i>text</i>]]>
</string>
```

Html.fromHtml(string)

```
<string name="html_text">
  <![CDATA[This <b>is</b> a <font face="serif">styled</font> <i>text</i>]]>
</string>
```

Html.fromHtml(string)

```
<string name="html_text">
  <![CDATA[This <b>is</b> a <font face="serif">styled</font> <i>text</i>]]>
</string>
```

Html.fromHtml(string)

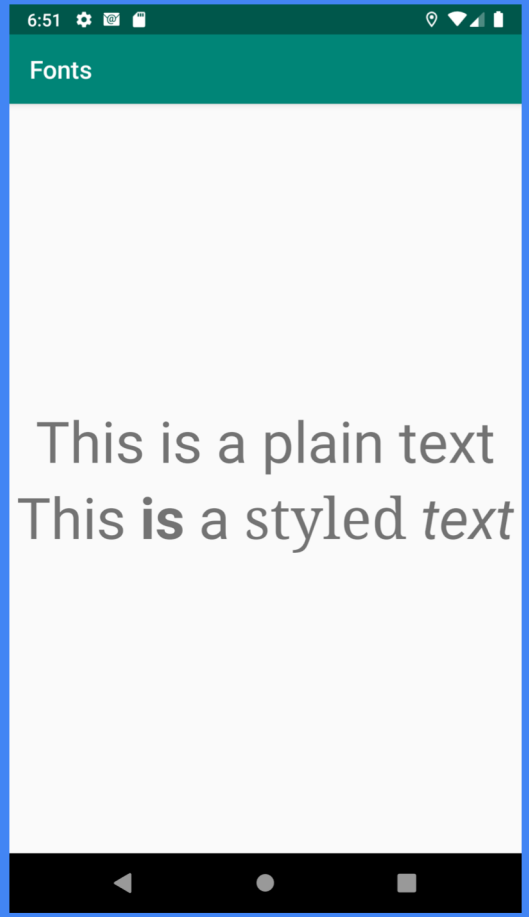
```
<string name="html_text">
  <![CDATA[This <b>is</b> a <font face="serif">styled</font> <i>text</i>]]>
</string>
```

Html.fromHtml(string)

```
<string name="html_text">
  <![CDATA[This <b>is</b> a <font face="serif">styled</font> <i>text</i>]]>
</string>
```

Html.fromHtml(string)

```
<string name="html_text">
  <![CDATA[This <b>is</b> a <font face="serif">styled</font> <i>text</i>]]>
</string>
```



Styled text

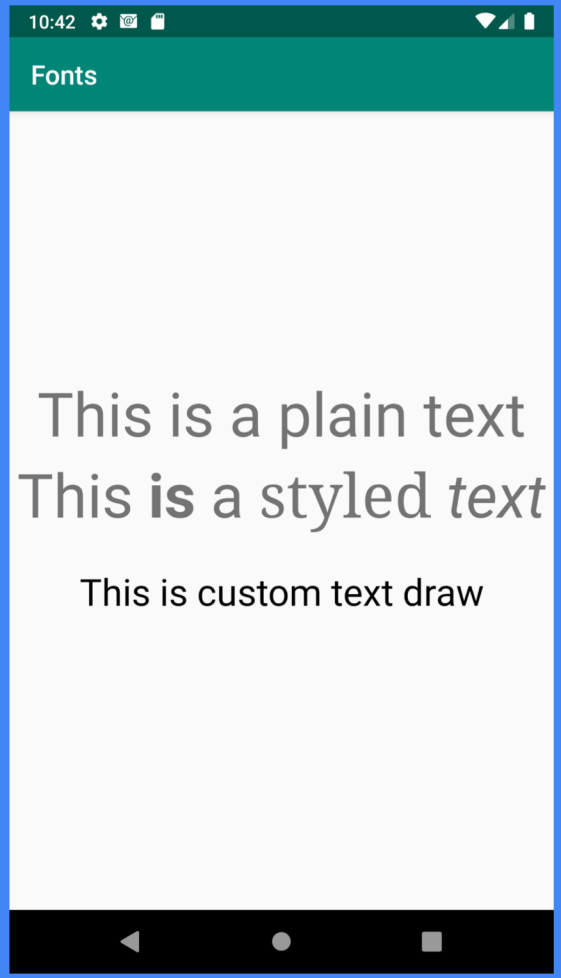
```
String string = getString(R.string.html_text);

((TextView) findViewById(R.id.textViewHtml))
        .setText(Html.fromHtml(string));
```

Как отображается текст?

TextView

Как отображается текст?

Custom View

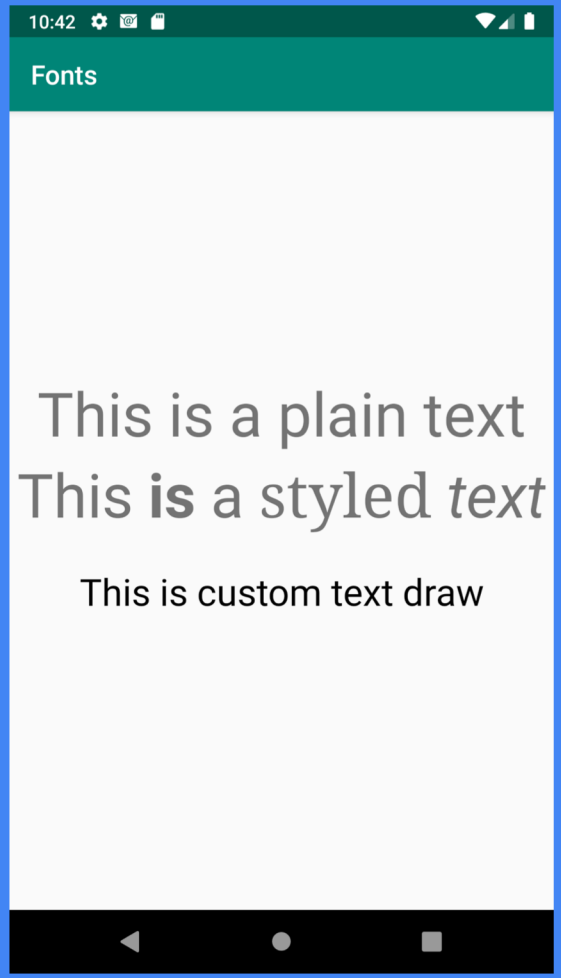


Custom view

```
private static final String TEXT_TO_DRAW
        = "This is custom text draw";
private TextPaint textPaint = new TextPaint();
private Rect textBounds = new Rect();
```

```
textPaint.setTextSize(textSize());
textPaint.getTextBounds(TEXT_TO_DRAW,
        0, TEXT_TO_DRAW.length(),
        textBounds);
textPaint.setAntiAlias(true);
```

```
canvas.drawText(TEXT_TO_DRAW,
        getWidth() / 2 - textBounds.width() / 2,
        getHeight() / 2 + textBounds.height() / 2,
        textPaint);
```

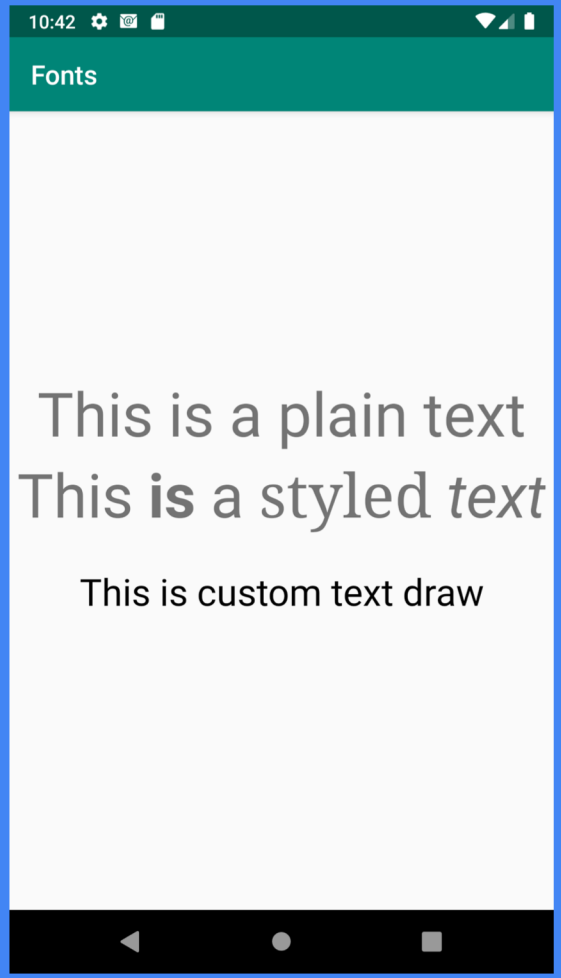


Custom view

```
private static final String TEXT_TO_DRAW
        = "This is custom text draw";
private TextPaint textPaint = new TextPaint();
private Rect textBounds = new Rect();
```

```
textPaint.setTextSize(textSize());
textPaint.getTextBounds(TEXT_TO_DRAW,
        0, TEXT_TO_DRAW.length(),
        textBounds);
textPaint.setAntiAlias(true);
```

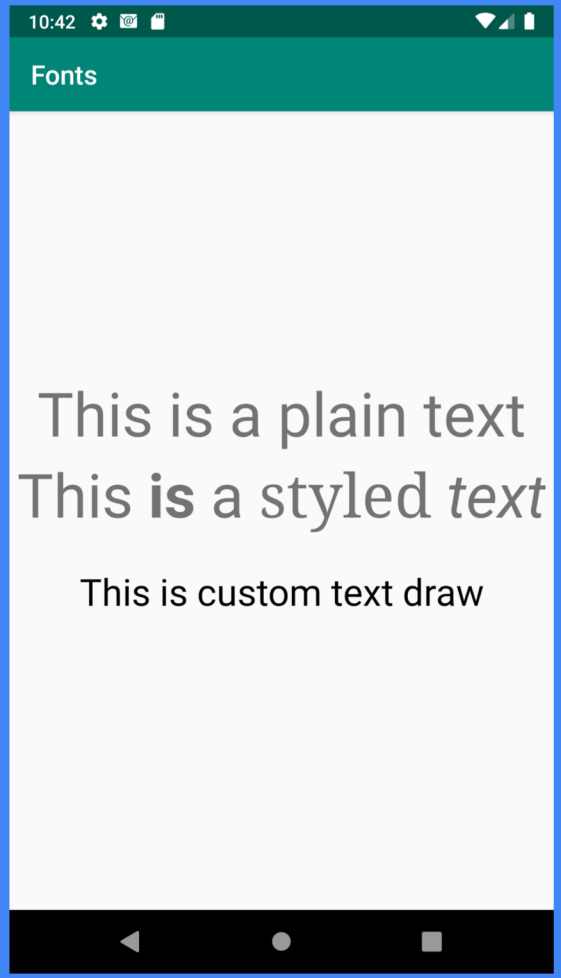
```
canvas.drawText(TEXT_TO_DRAW,
        getWidth() / 2 - textBounds.width() / 2,
        getHeight() / 2 + textBounds.height() / 2,
        textPaint);
```



Custom view

```
private static final String TEXT_TO_DRAW
        = "This is custom text draw";
private TextPaint textPaint = new TextPaint();
private Rect textBounds = new Rect();


textPaint.setTextSize(textSize());
textPaint.getTextBounds(TEXT_TO_DRAW,
        0, TEXT_TO_DRAW.length(),
        textBounds);
textPaint.setAntiAlias(true);
```

```
canvas.drawText(TEXT_TO_DRAW,
        getWidth() / 2 - textBounds.width() / 2,
        getHeight() / 2 + textBounds.height() / 2,
        textPaint);
```



Custom view

```
private static final String TEXT_TO_DRAW
        = "This is custom text draw";
private TextPaint textPaint = new TextPaint();
private Rect textBounds = new Rect();
```

```
textPaint.setTextSize(textSize());
textPaint.getTextBounds(TEXT_TO_DRAW,
        0, TEXT_TO_DRAW.length(),
        textBounds);
textPaint.setAntiAlias(true);
```

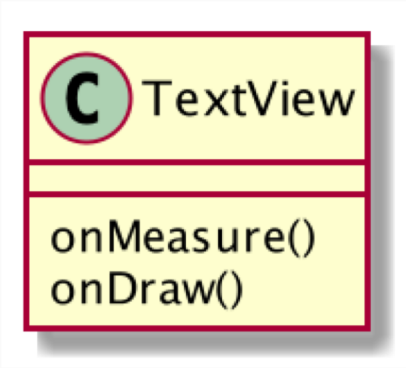
```
canvas.drawText(TEXT_TO_DRAW,
        getWidth() / 2 - textBounds.width() / 2,
        getHeight() / 2 + textBounds.height() / 2,
        textPaint);
```

Что внутри TextView?

Что делает TextView?

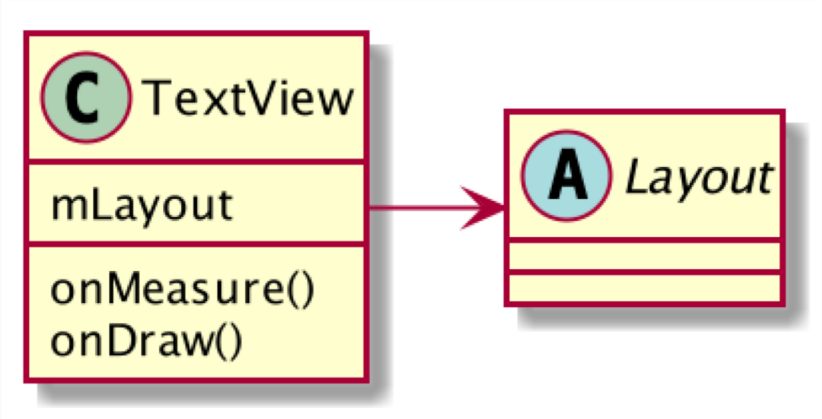
Что делает TextView для того, чтобы отобразить текст?

1. Размечает текст
2. Рисует текст



Что делает TextView?

Измерение текста: Layout



Что делает TextView?

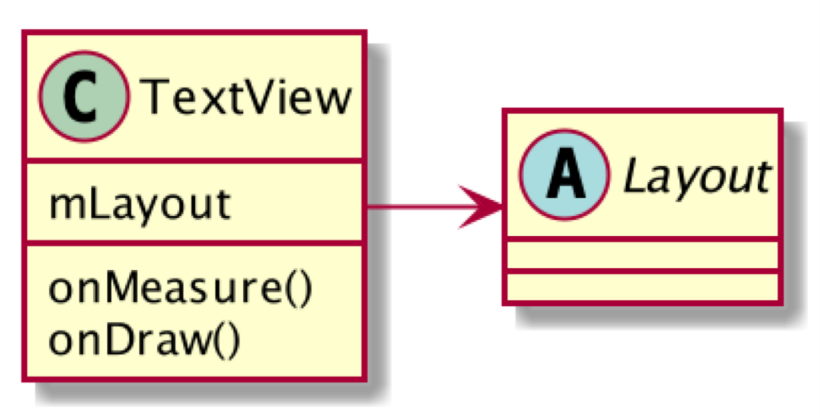
Измерение текста

makeLayout()

- Вызывается при изменении текста, размеров или настроек разметки

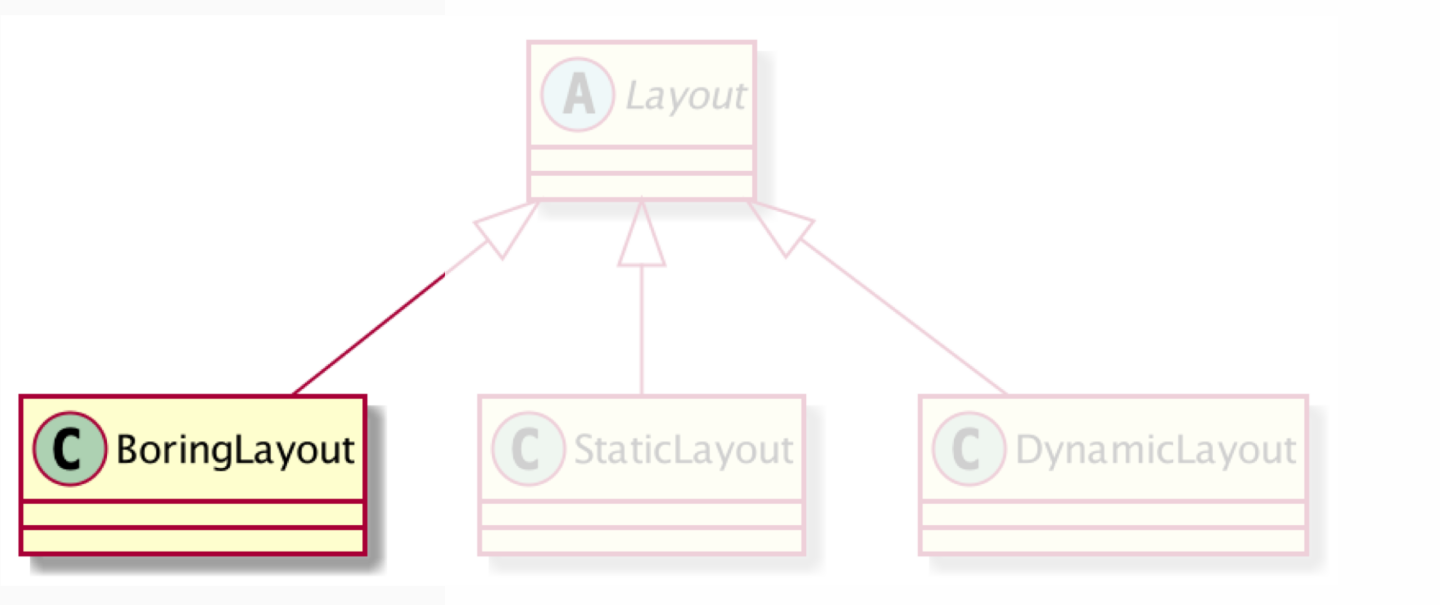
Что делает TextView?

Измерение текста



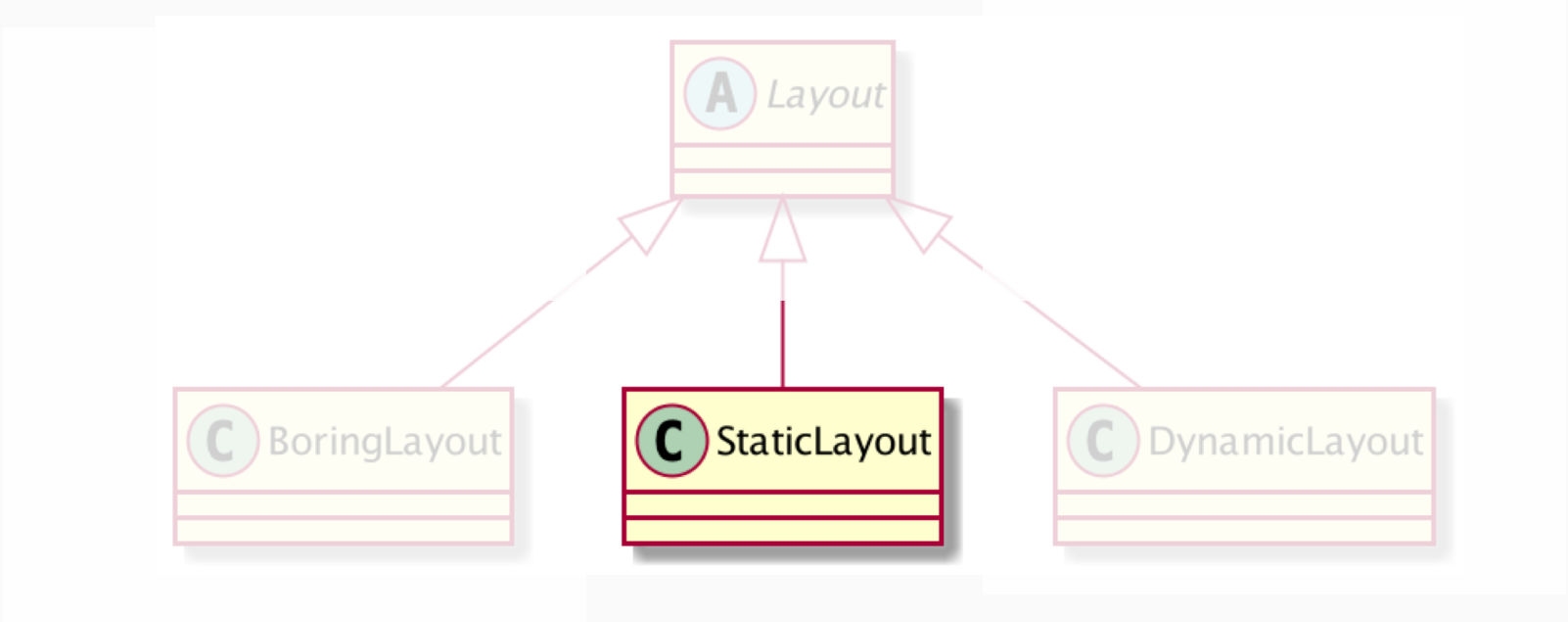
Что делает TextView?

Измерение текста



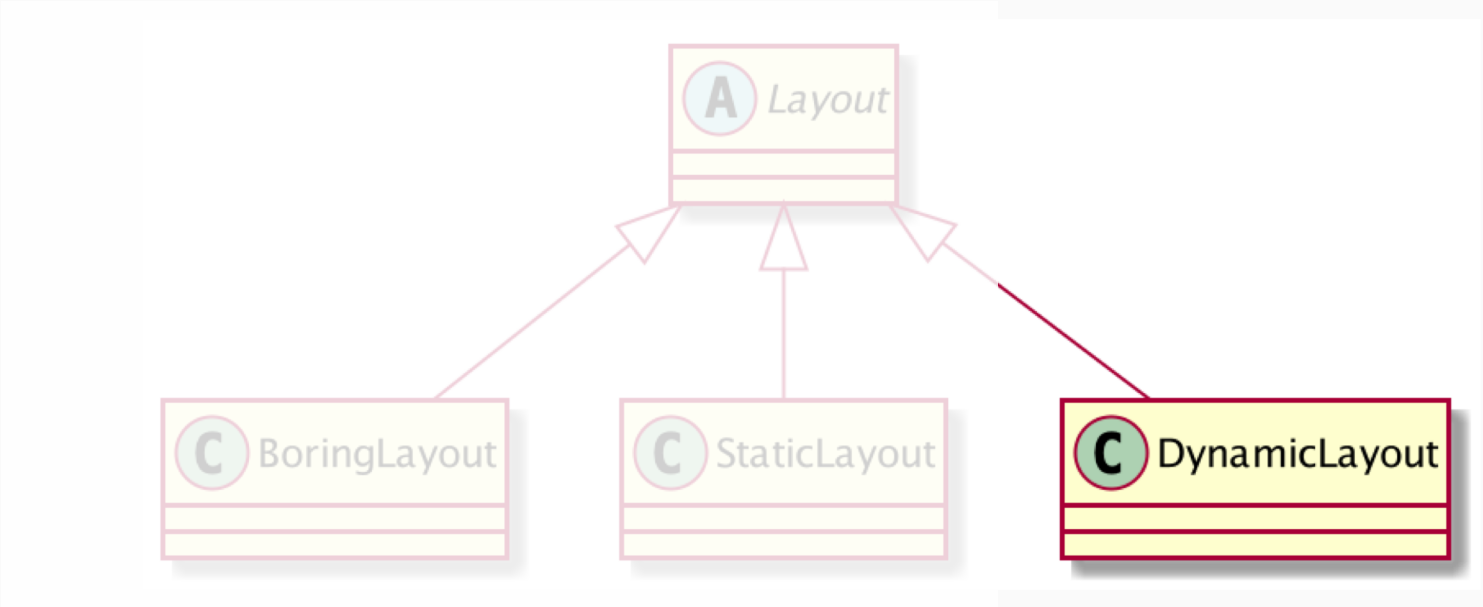
Что делает TextView?

Измерение текста



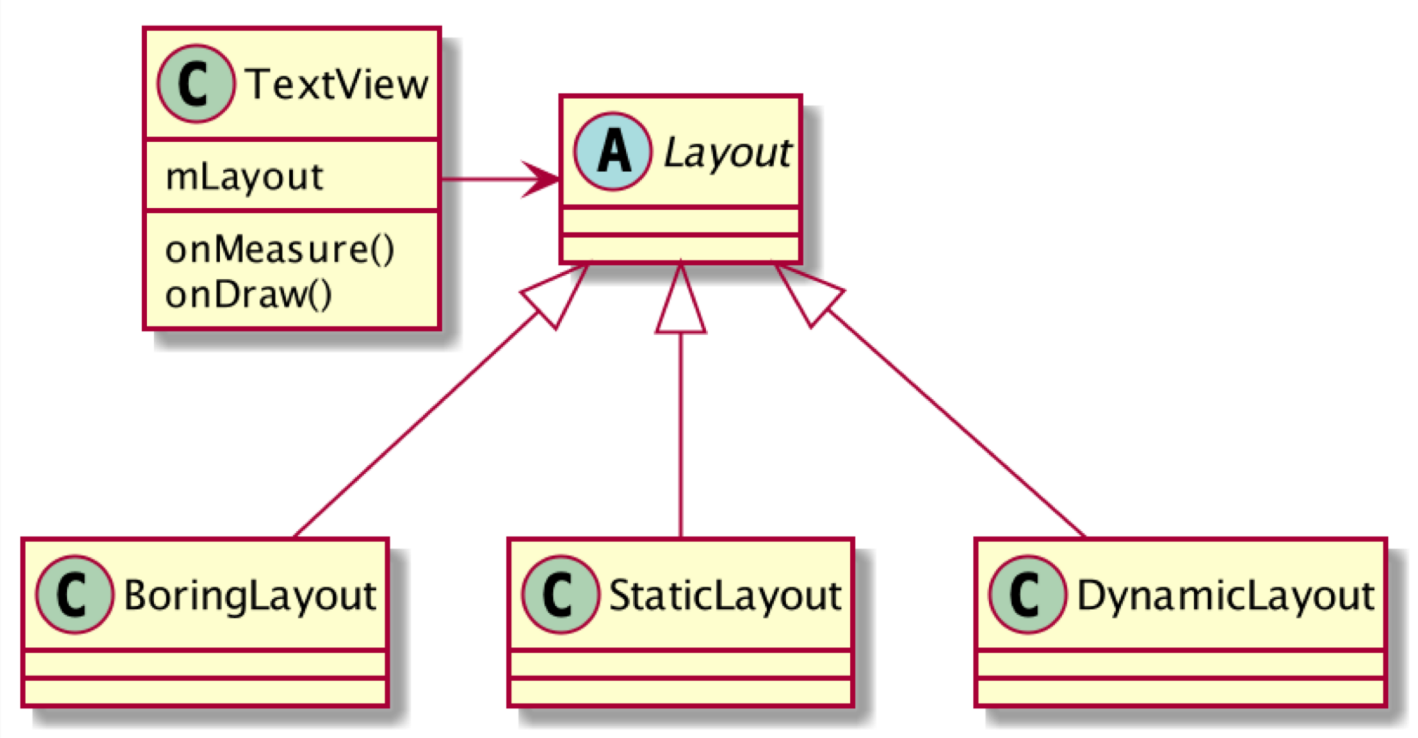
Что делает TextView?

Измерение текста



Что делает TextView?

Измерение текста



Что делает TextView?

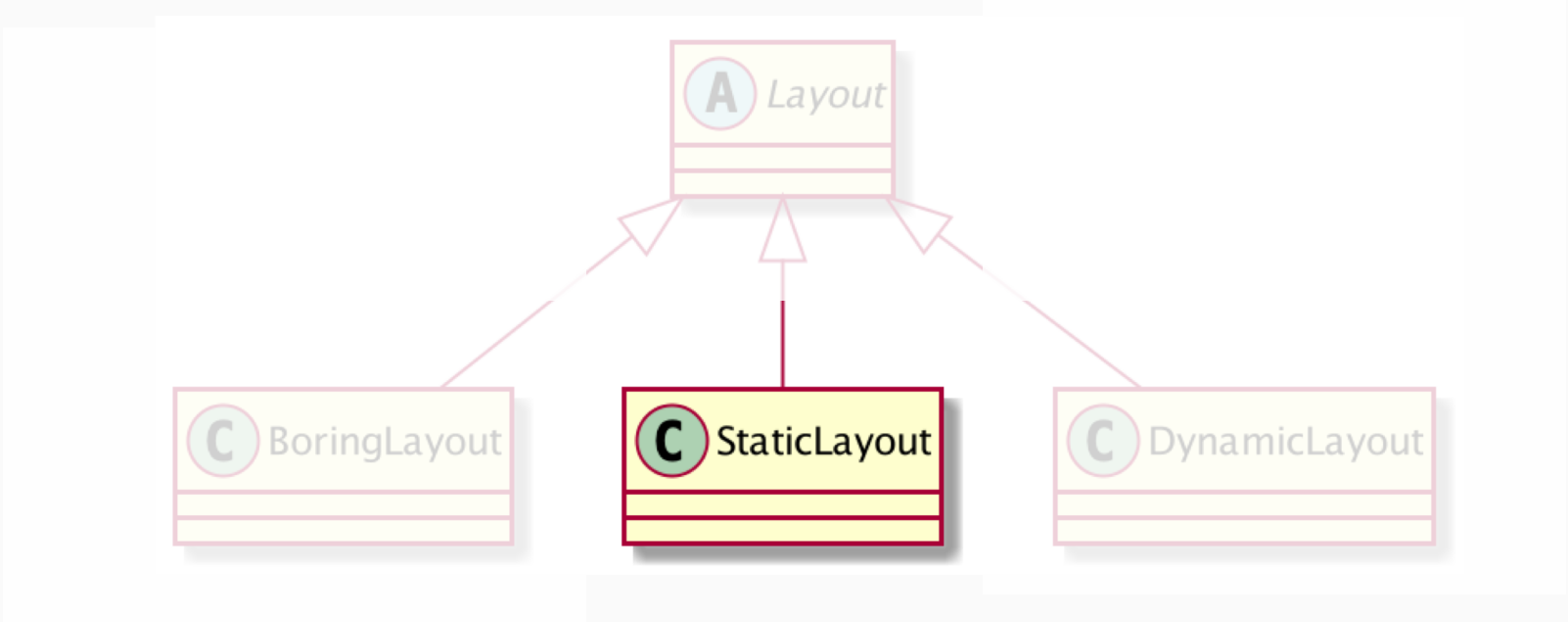
Измерение текста

makeLayout()

- Вызывается при изменении текста, размеров или настроек разметки
- Создает конкретный Layout в зависимости от условий

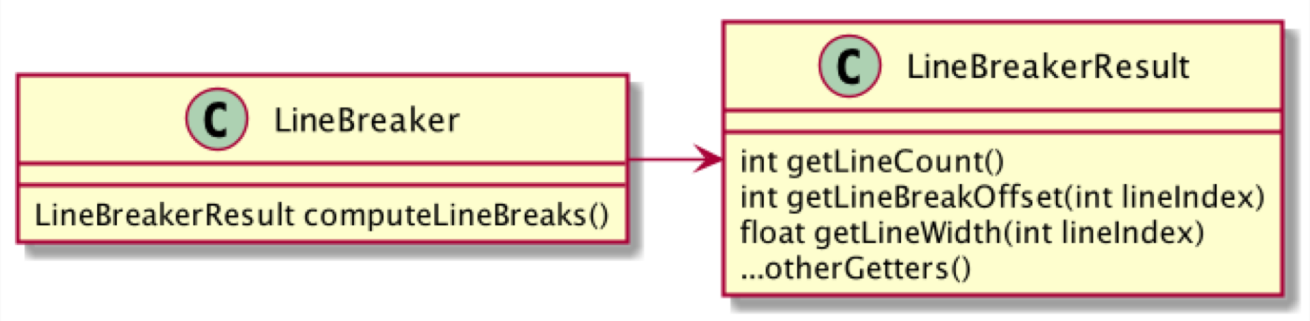
Что делает TextView?

Измерение текста



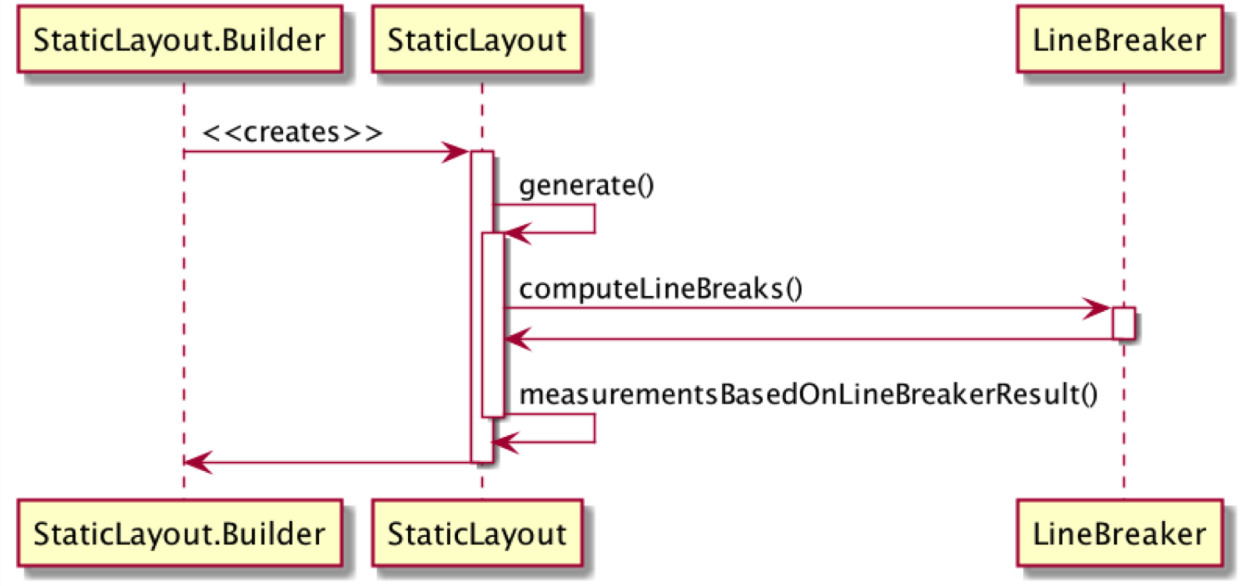
Что делает TextView?

Разбиение текста по строкам



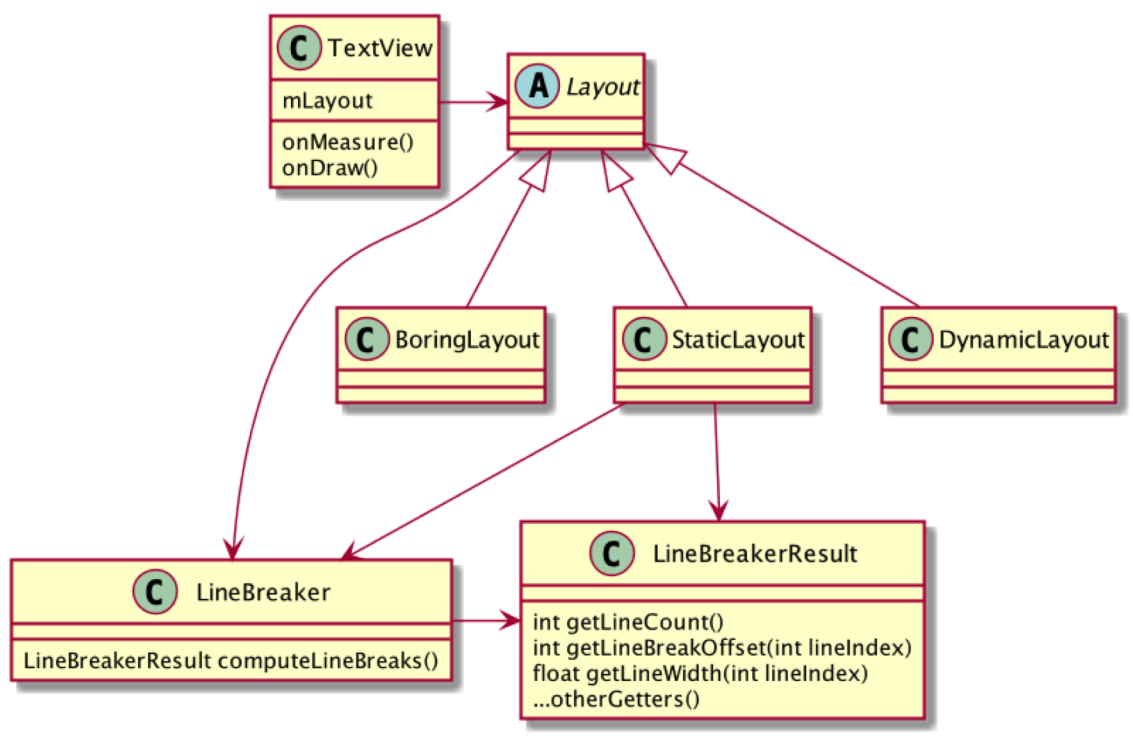
Что делает TextView?

Разбиение текста по строкам происходит синхронно



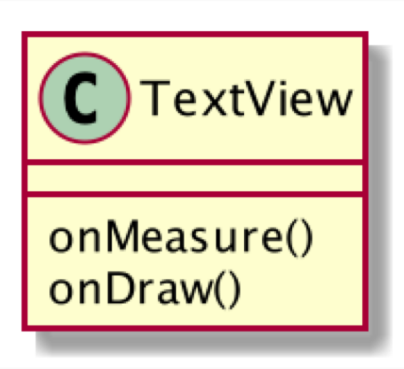
Что делает TextView?

Измерение текста



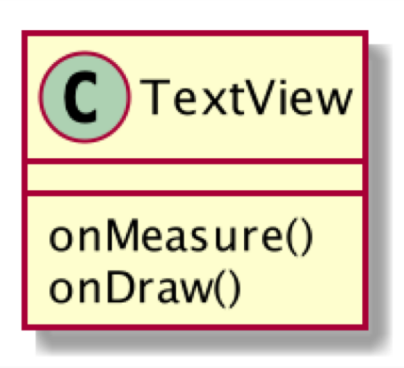
Что делает TextView?

- **Измеряет текст**
- Рисует текст



Что делает TextView?

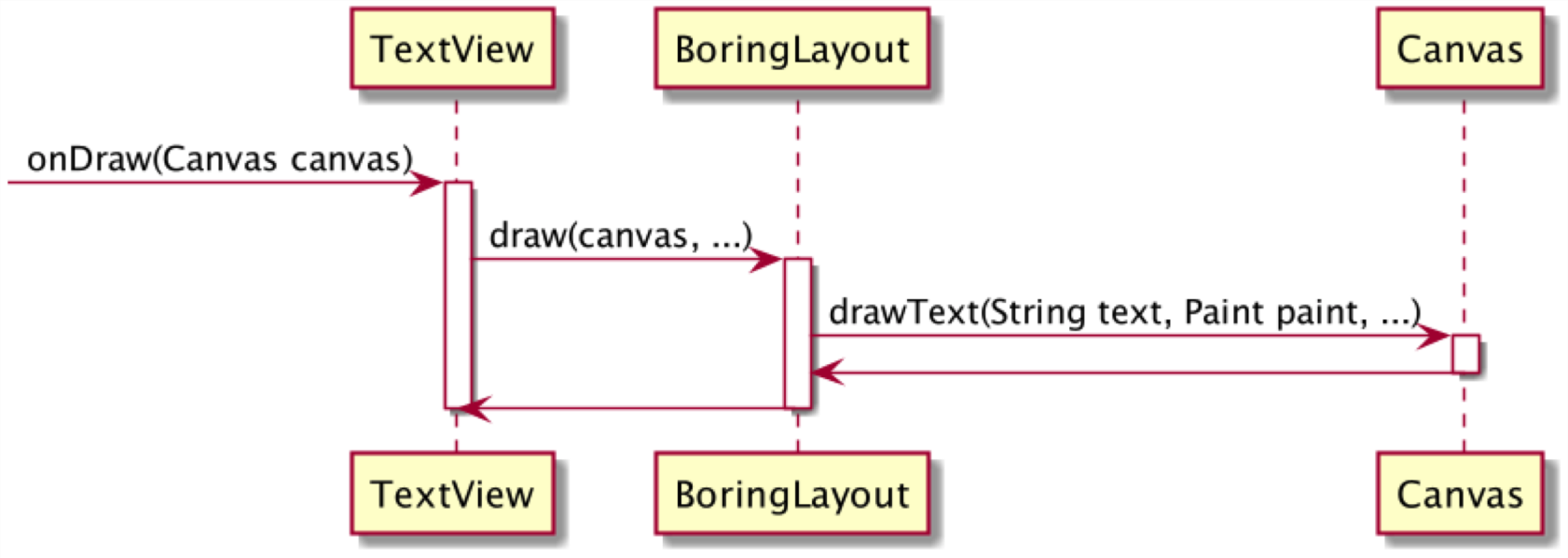
- Измеряет текст
- **Рисует текст**



Что делает TextView?

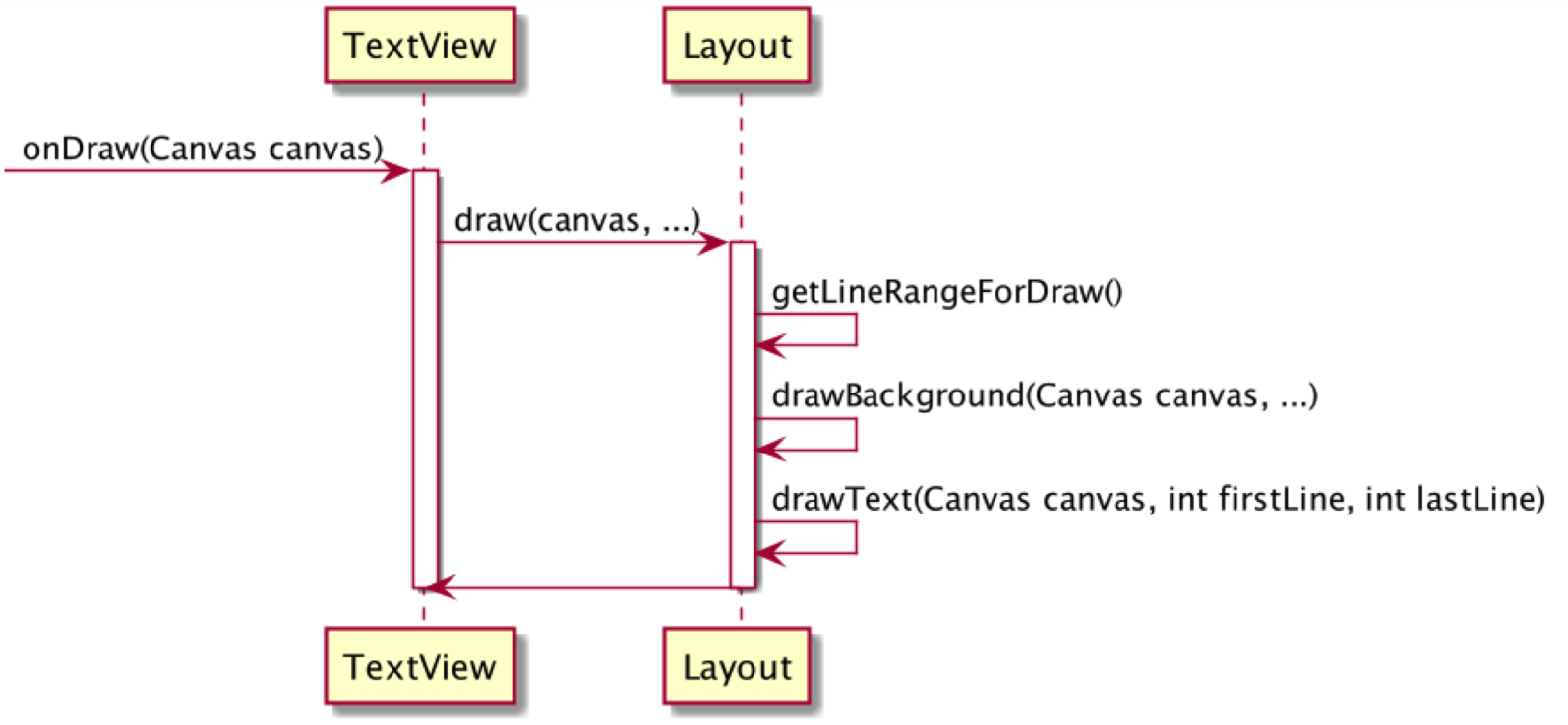
Рисование “простого” текста

Текст без переносов и форматирования



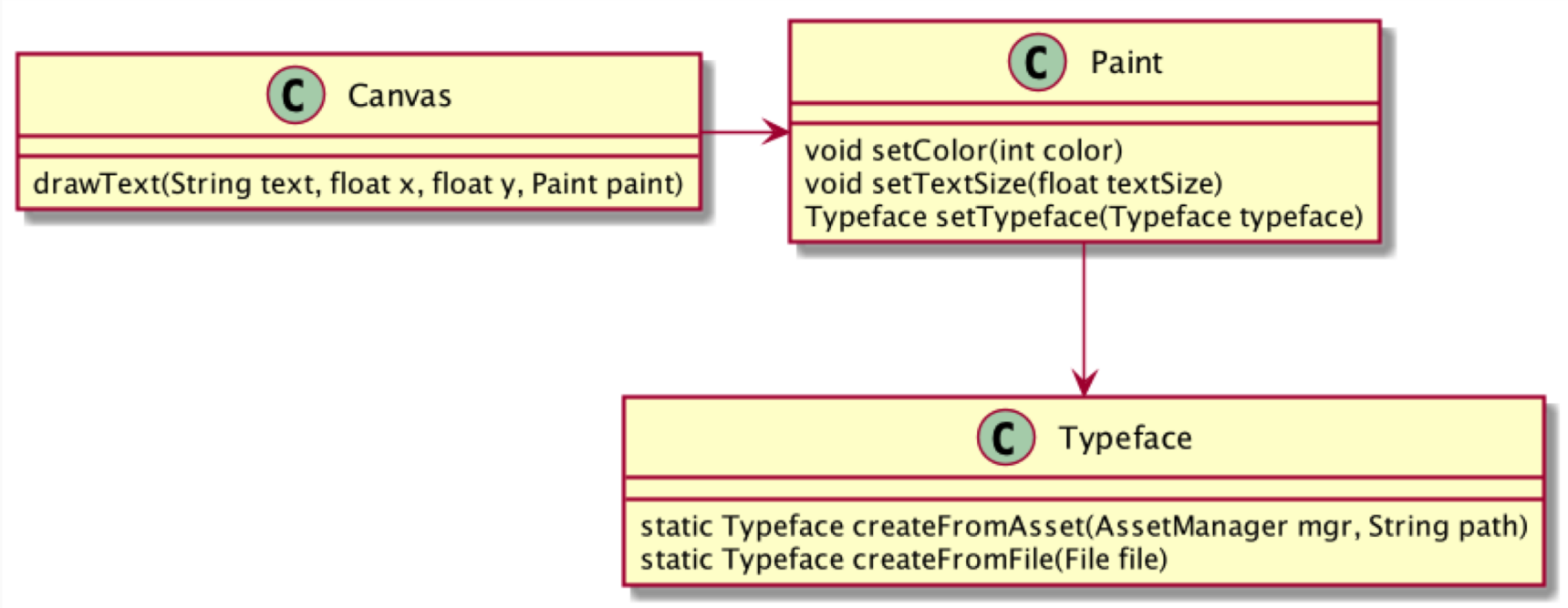
Что делает TextView?

Рисование “обычного” текста



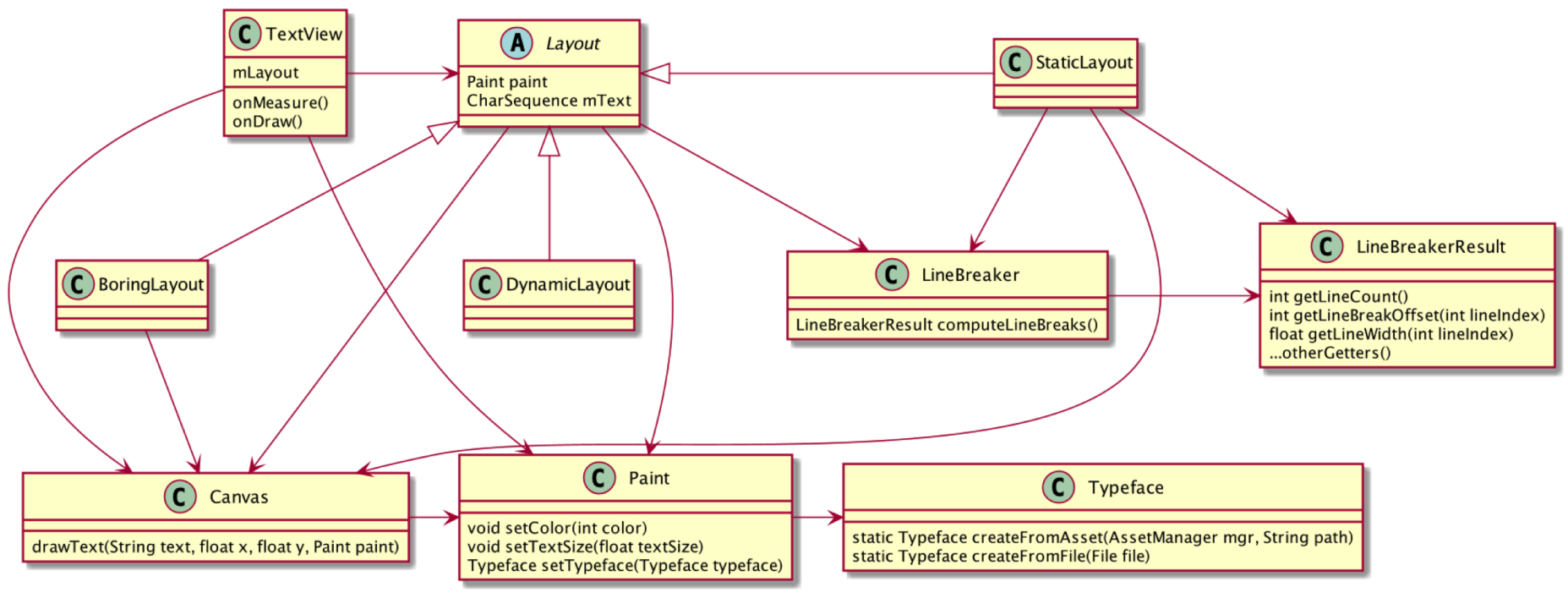
Что делает TextView?

Рисование текста



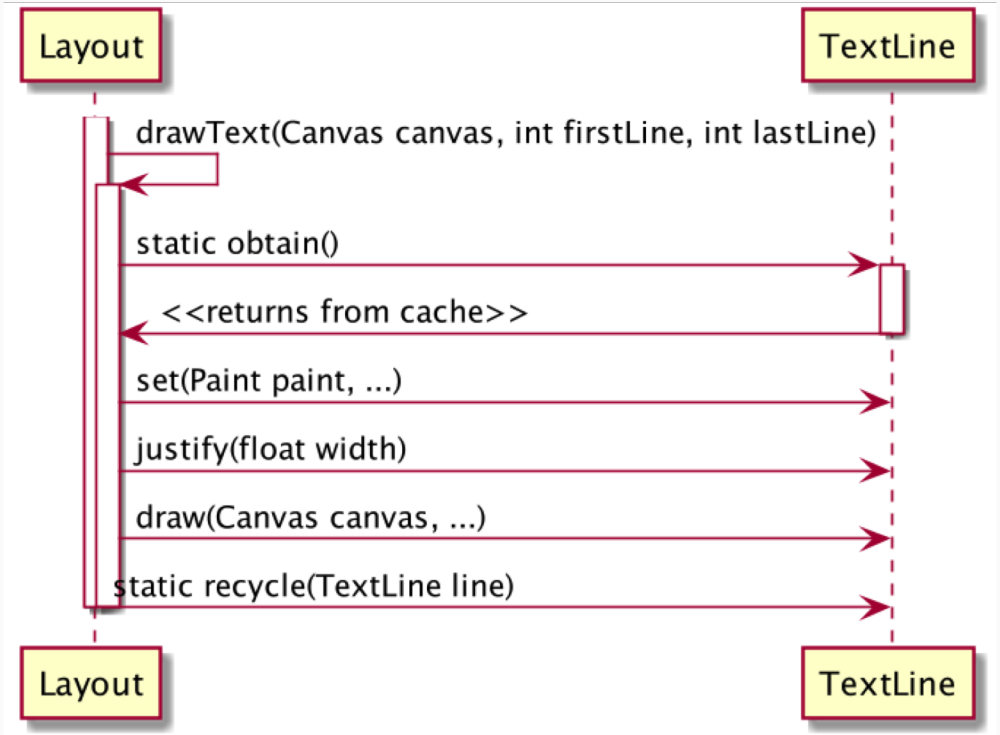
Что делает TextView?

Рисование текста



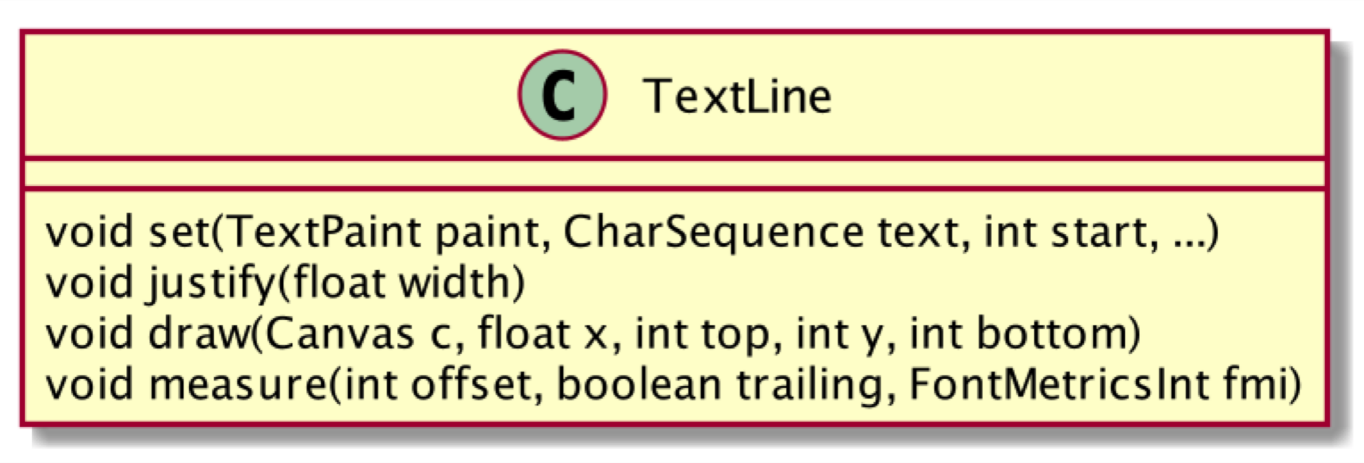
Что делает TextView?

Рисование текста построчно



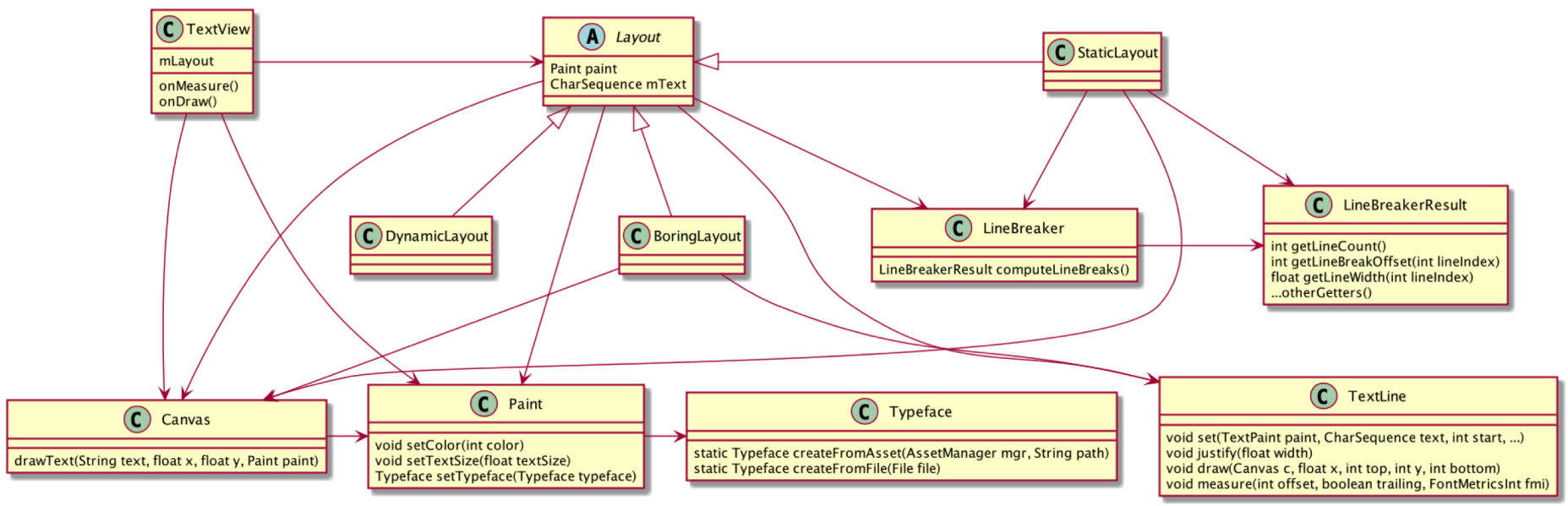
Что делает TextView?

Рисование текста



Что делает TextView?

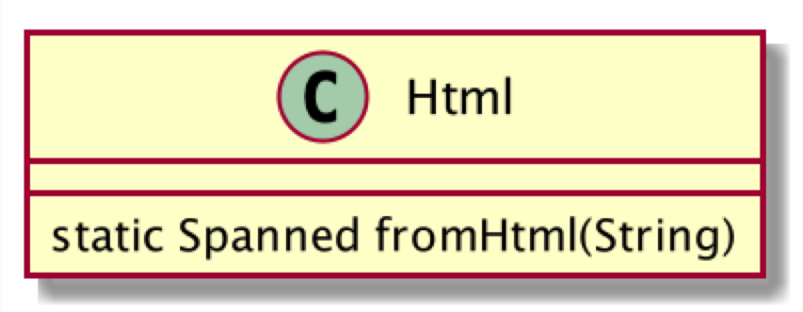
Рисование текста



Html.fromHtml(string)

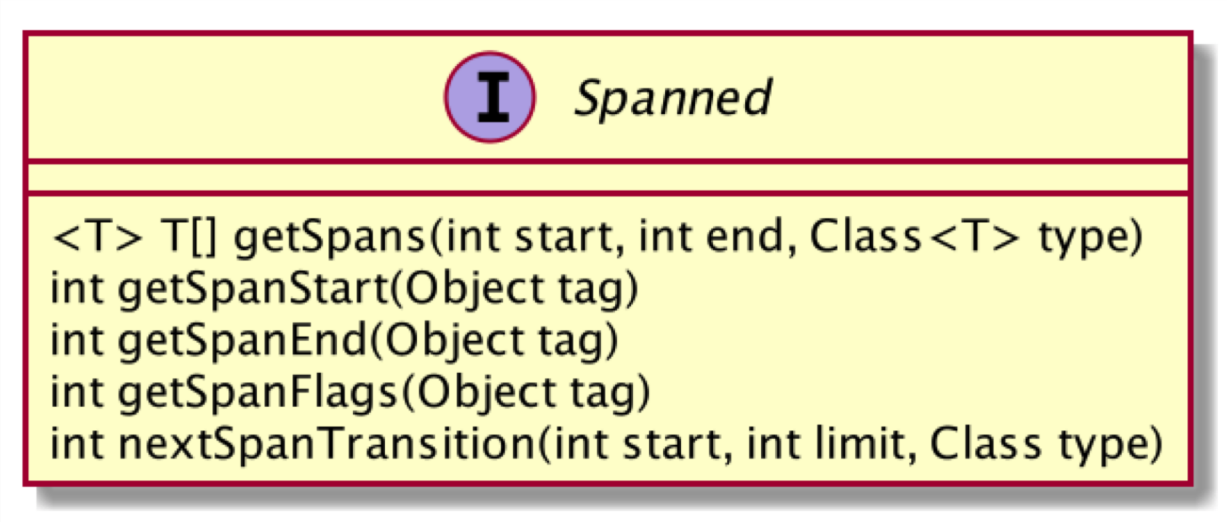
```
<string name="html_text">
  <![CDATA[This <b>is</b> a <font face="serif">styled</font> <i>text</i>]]>
</string>
```

Html.fromHtml(string)



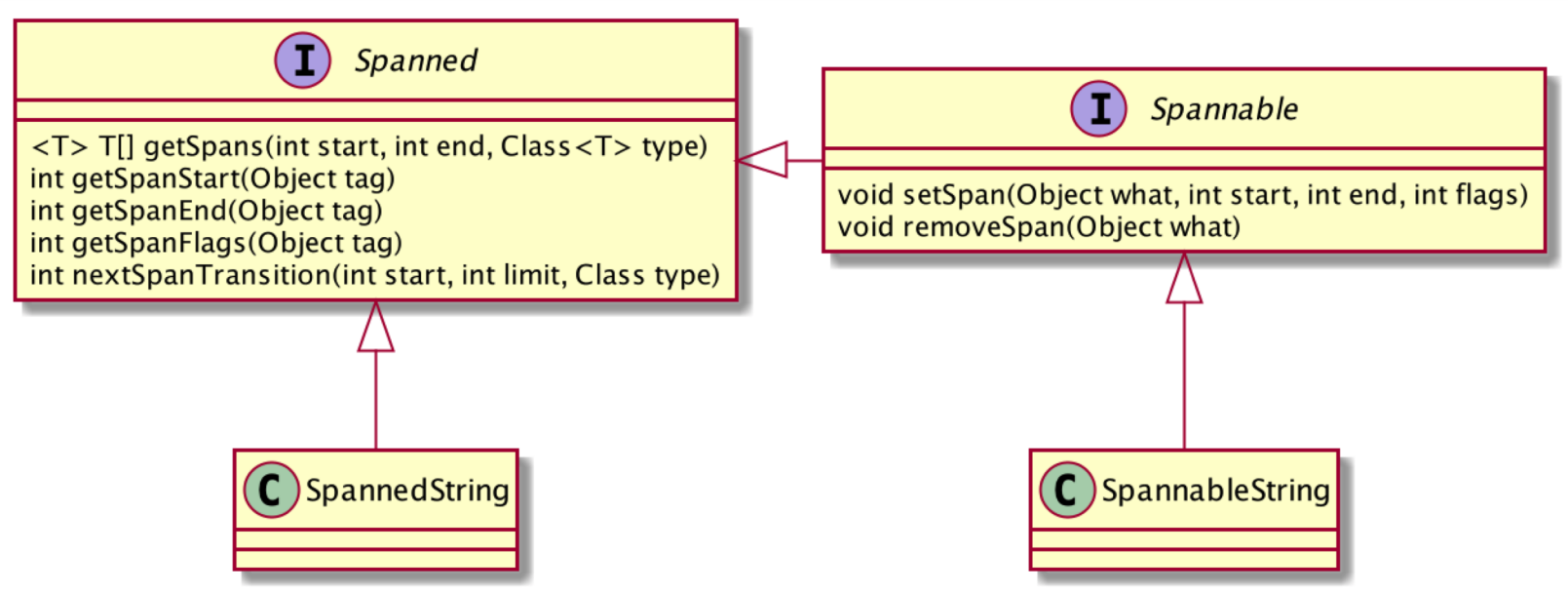
Что делает TextView?

Spans



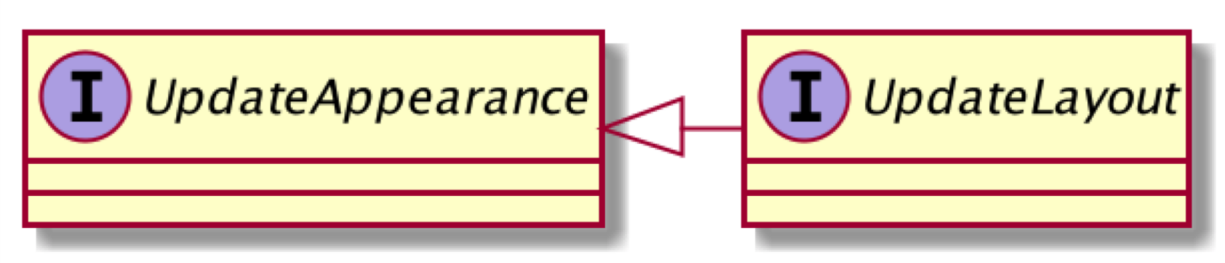
Что делает TextView?

Spans



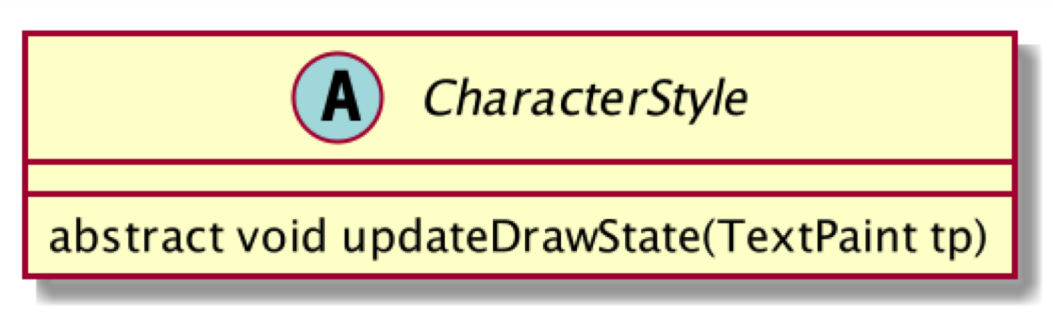
Что делает TextView?

Spans



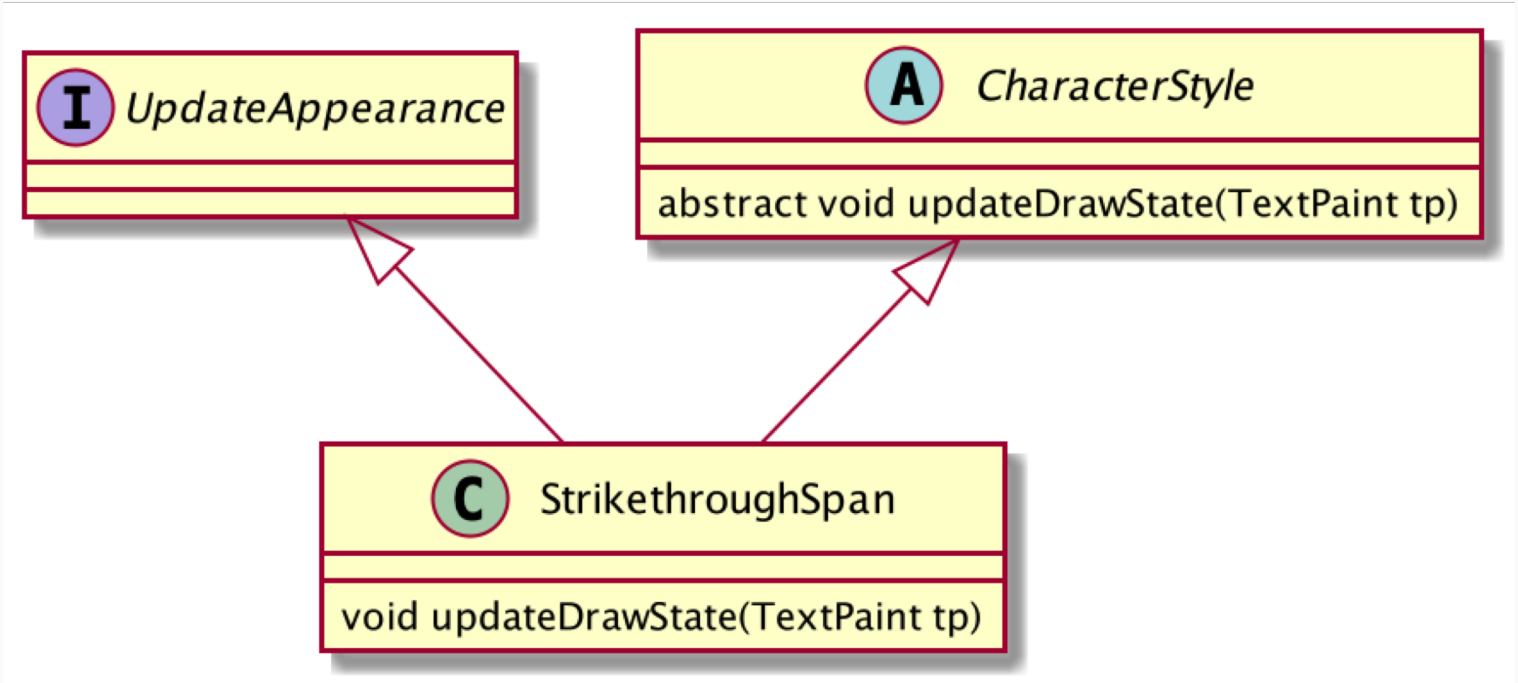
Что делает TextView?

Spans



Что делает TextView?

Spans



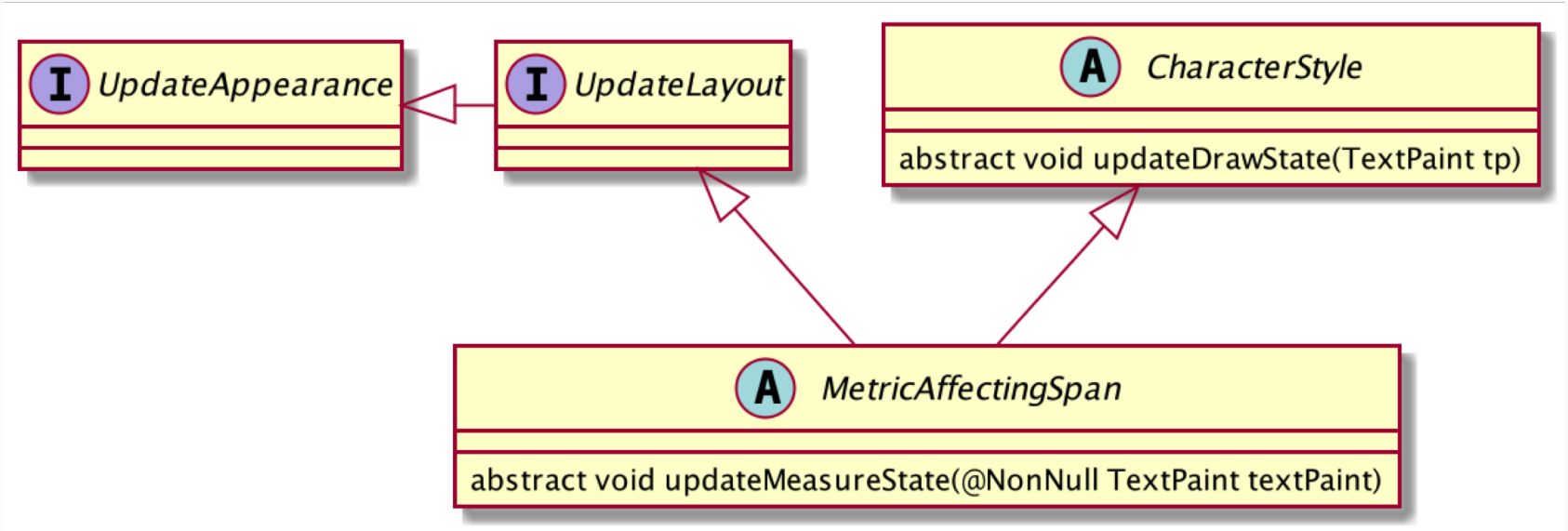
Что делает TextView?

Spans

```
@Override
public void updateDrawState(@NonNull TextPaint ds) {
    ds.setStrikeThruText(true);
}
```

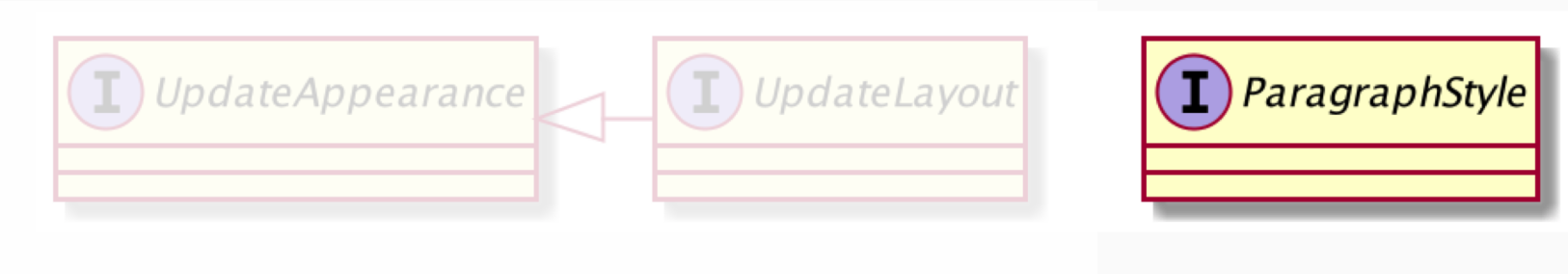
Что делает TextView?

Spans



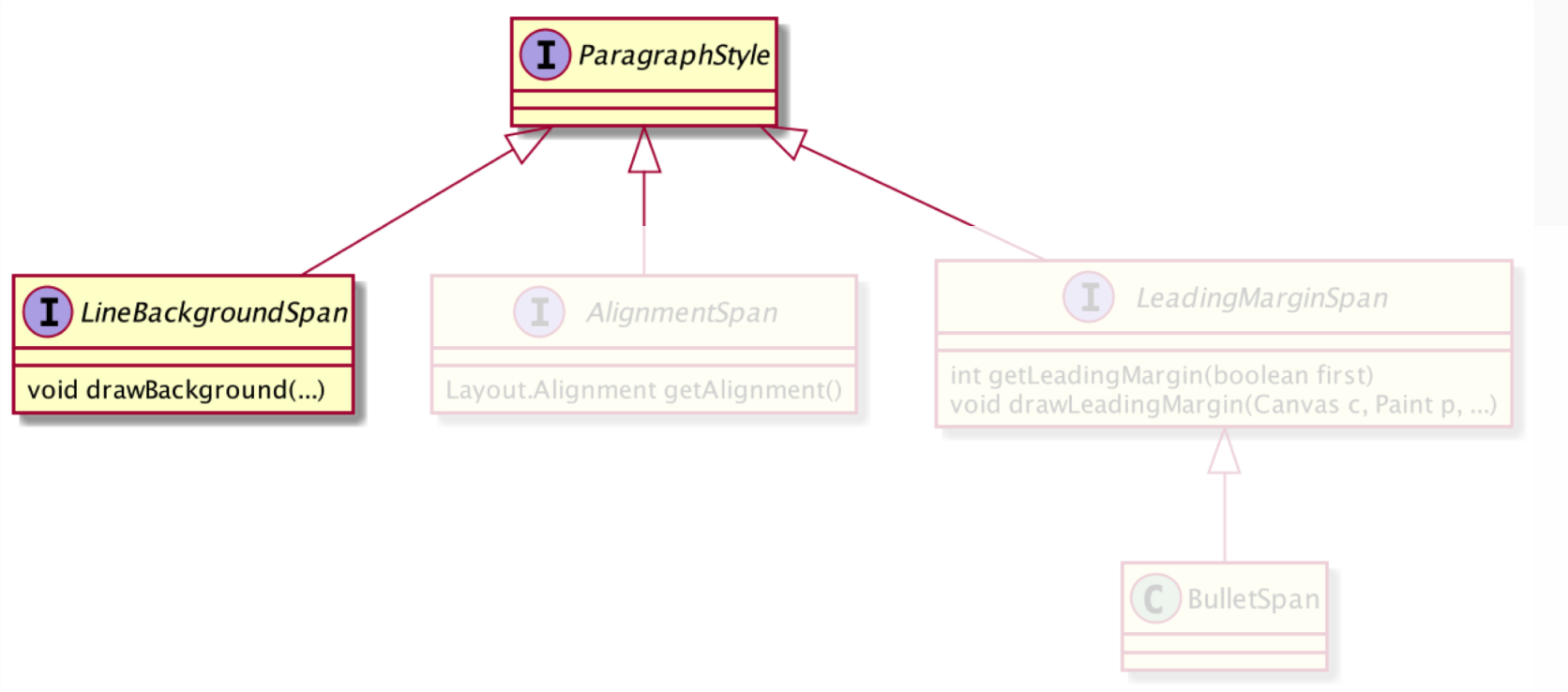
Что делает TextView?

Spans



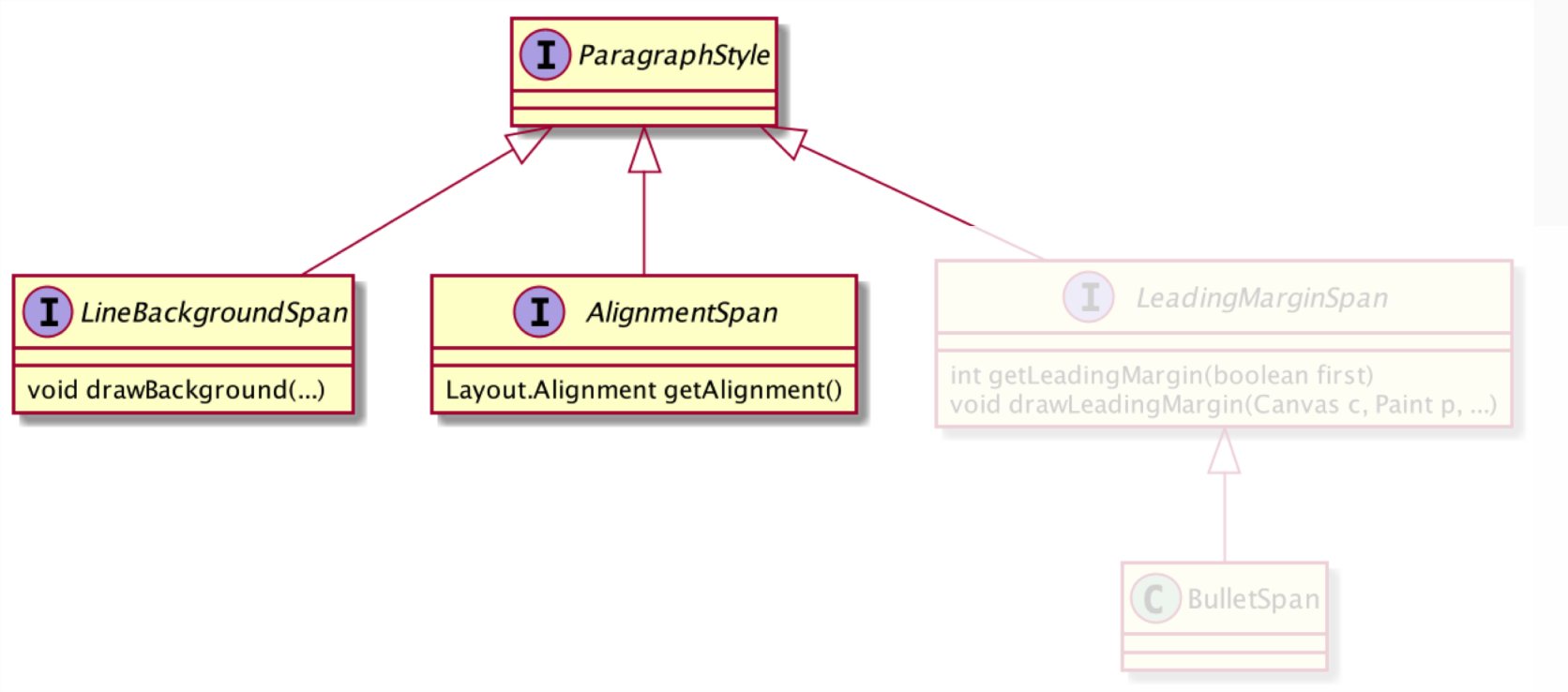
Что делает TextView?

Spans



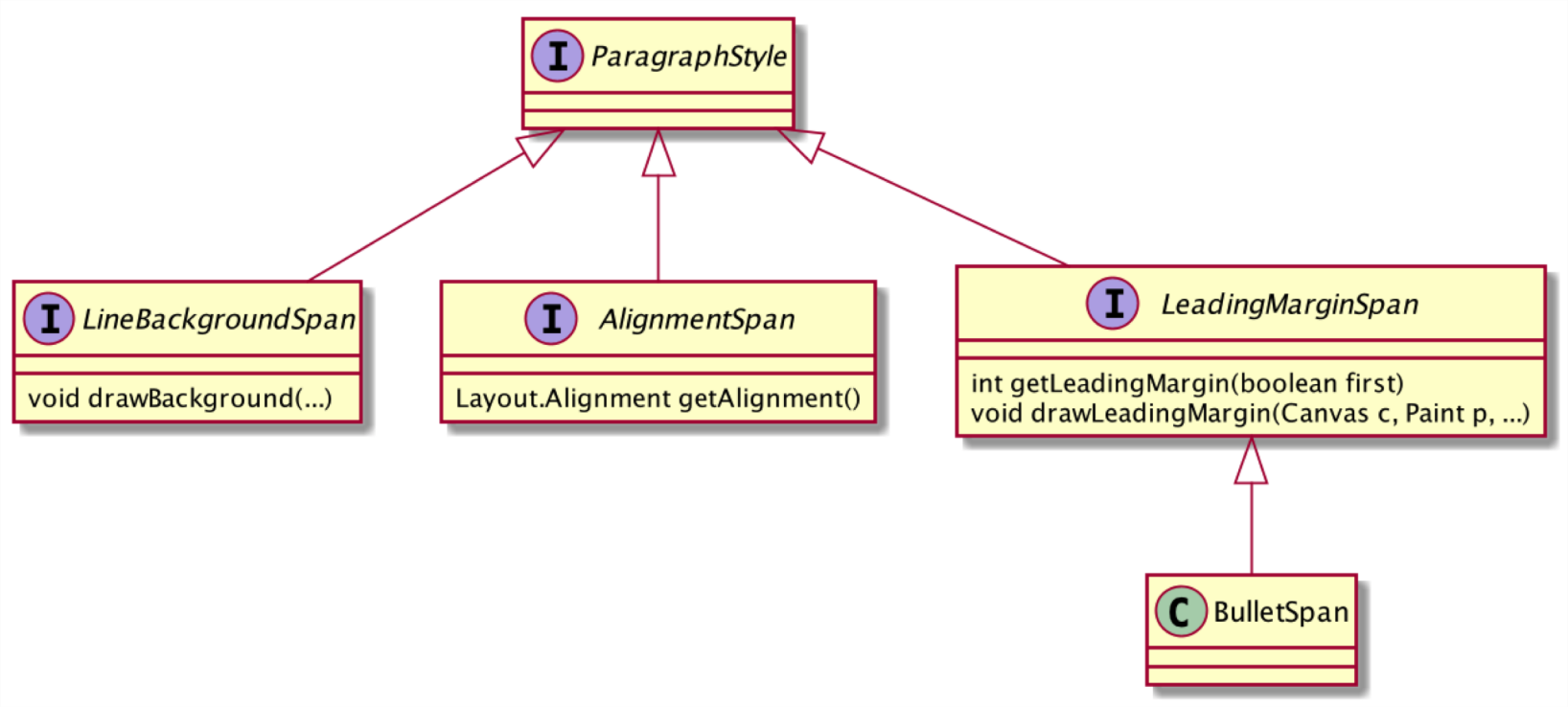
Что делает TextView?

Spans



Что делает TextView?

Spans



Что делает TextView?

Spans

Span - это стратегия обновления TextPaint’а

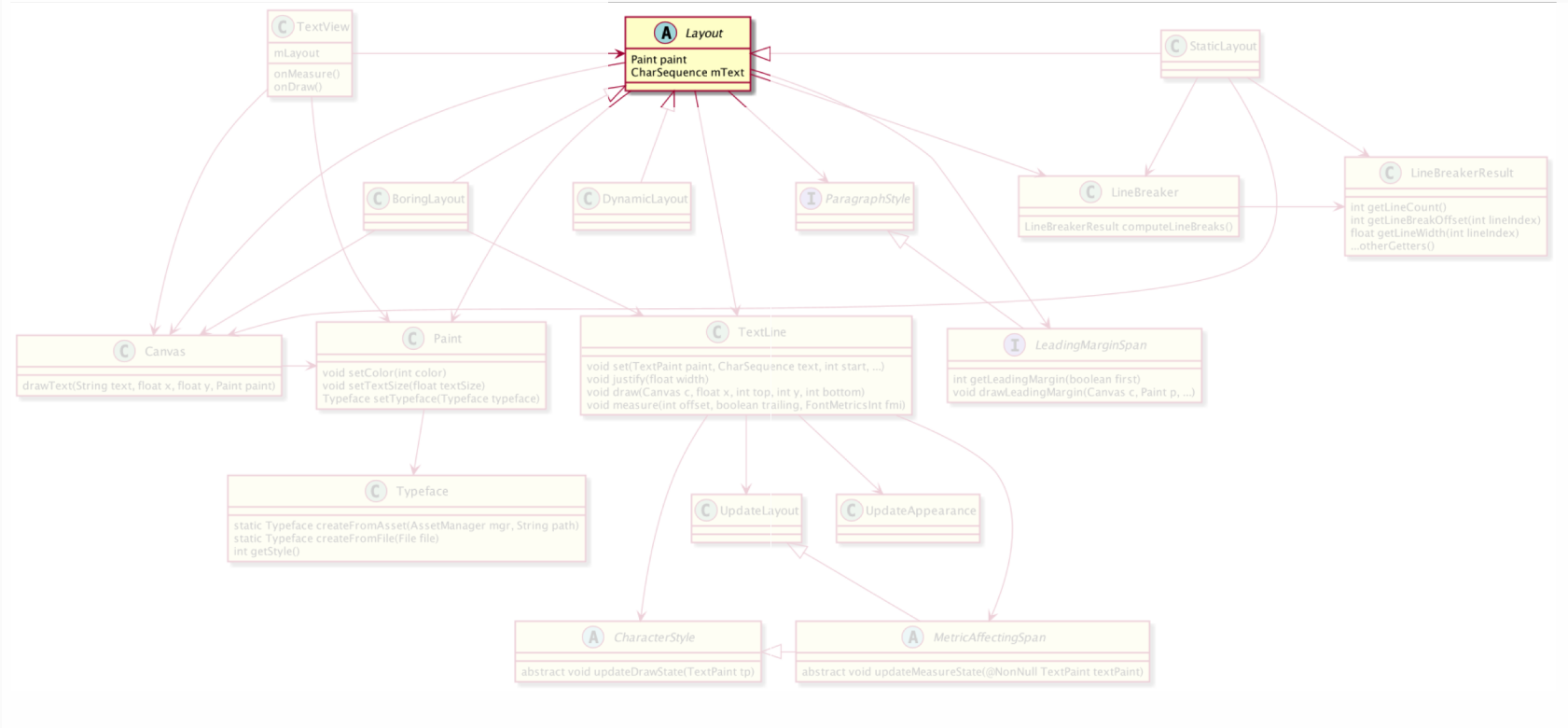
Что делает TextView?

Spans

Как span’ы участвуют в отображении текста?

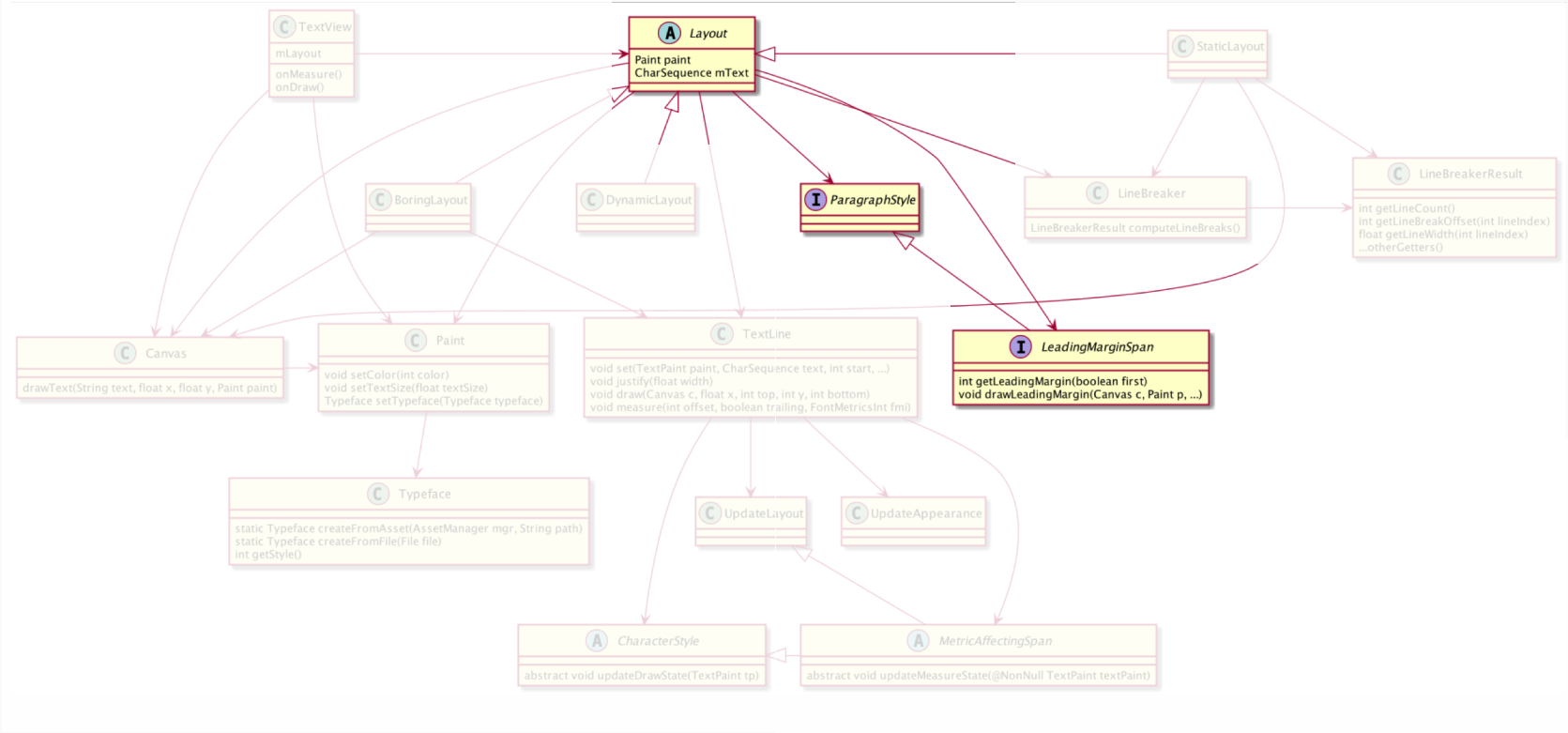
Что делает TextView?

Spans



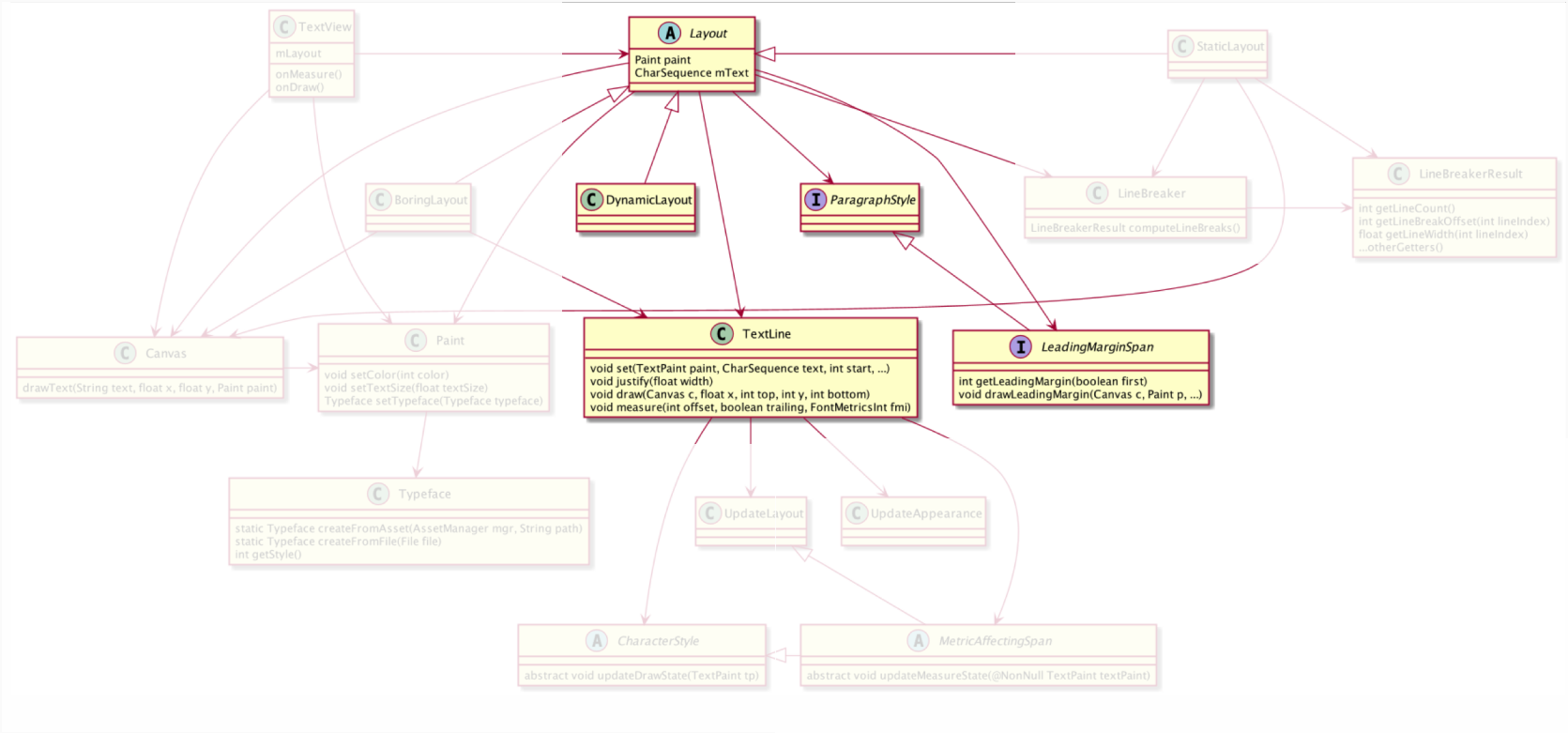
Что делает TextView?

Spans



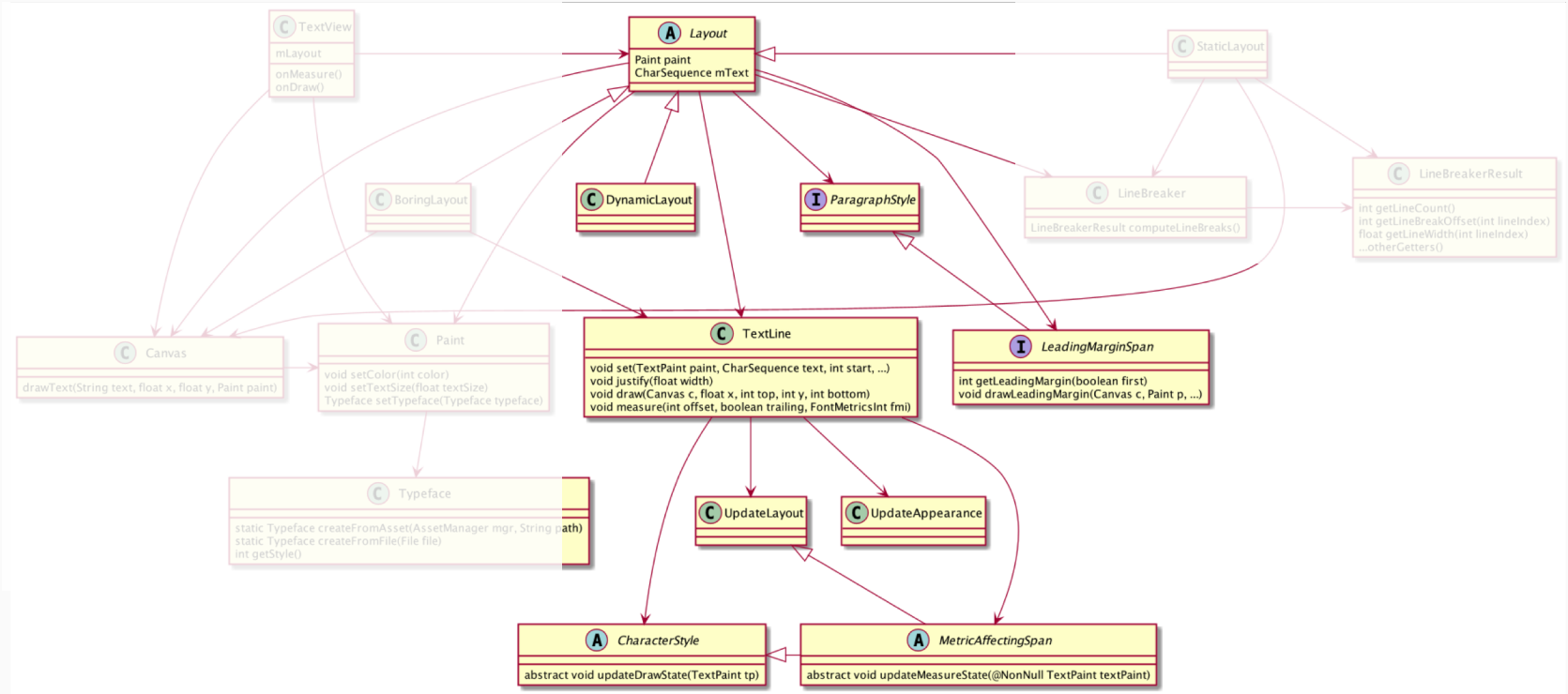
Что делает TextView?

Spans



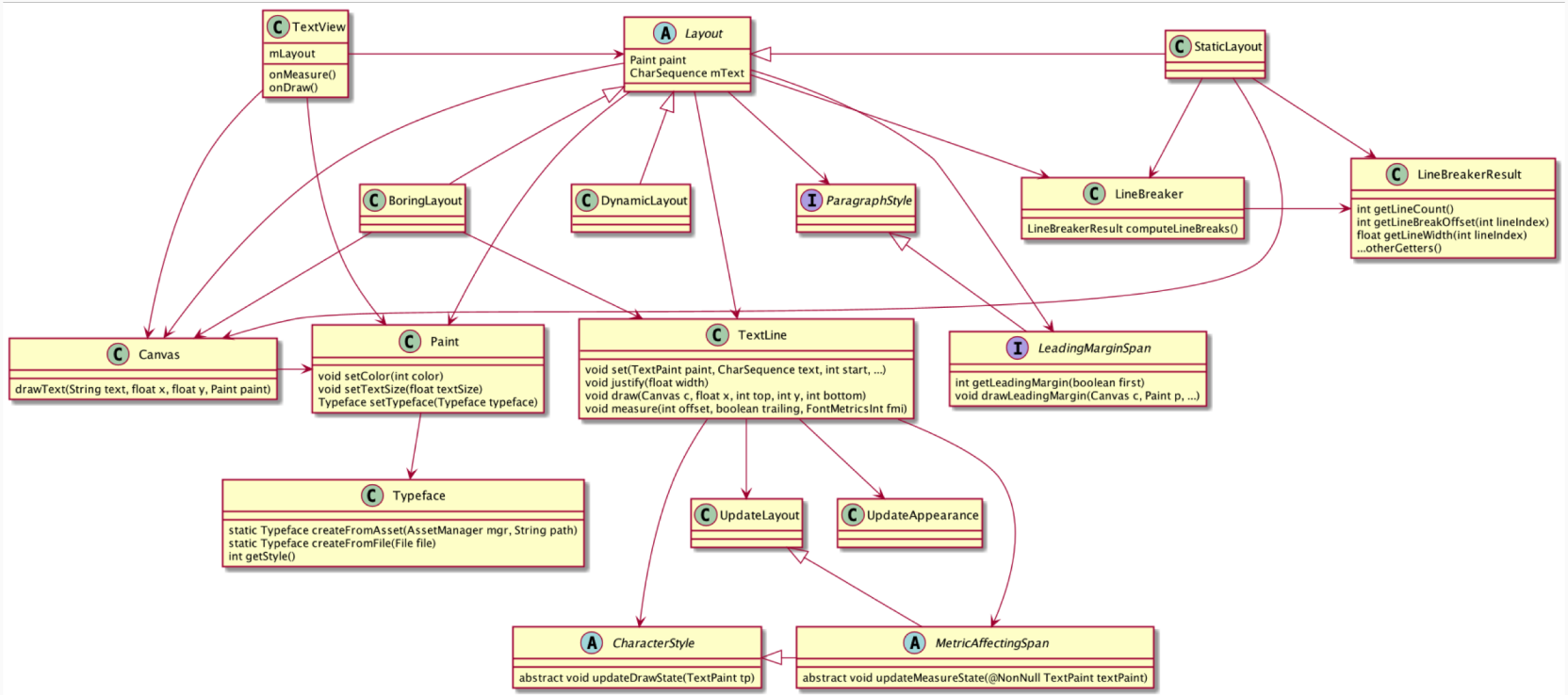
Что делает TextView?

Spans



Что делает TextView?

Spans



Html.fromHtml(string)

```
<string name="html_text">
  <![CDATA[This <b>is</b> a <font face="serif">styled</font> <i>text</i>]]>
</string>
```

1. String -> Spannable
2. TextView -> setText() -> createLayout()
3. TextView.onMeasure()
    a. Layout.measure() -> TextLine.measure()
    b. MetricAffectingSpan.updateMeasureState()
    c. Complete measuring based on span info
4. TextView.onDraw()
    a. Layout.draw() -> TextLine.draw()
    b. CharacterStyle.updateDrawState()
    c. Complete drawing with updated text paint instance

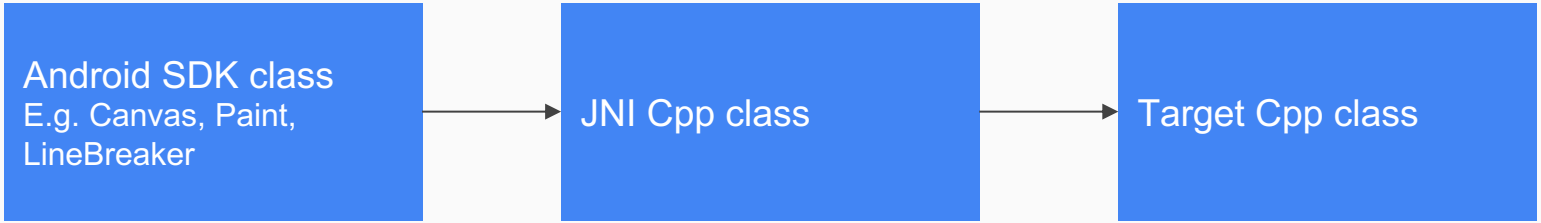
Html.fromHtml(string)

```
<string name="html_text">
  <![CDATA[This <b>is</b> a <font face="serif">styled</font> <i>text</i>]]>
</string>
```

This **is** a styled *text*

Стек компонентов

Android JNI




```
long mNativePointer = nativeInit()
```

```
reinterpret_cast<T*>(ptr)
```

Text breaking & measurement

Text breaking & **measurement**

Шрифт / Font

A *font* is a collection of various character images that can be used to display or print text.

Семейство шрифтов/ Гарнитура / Font Family / Type Face

Roboto

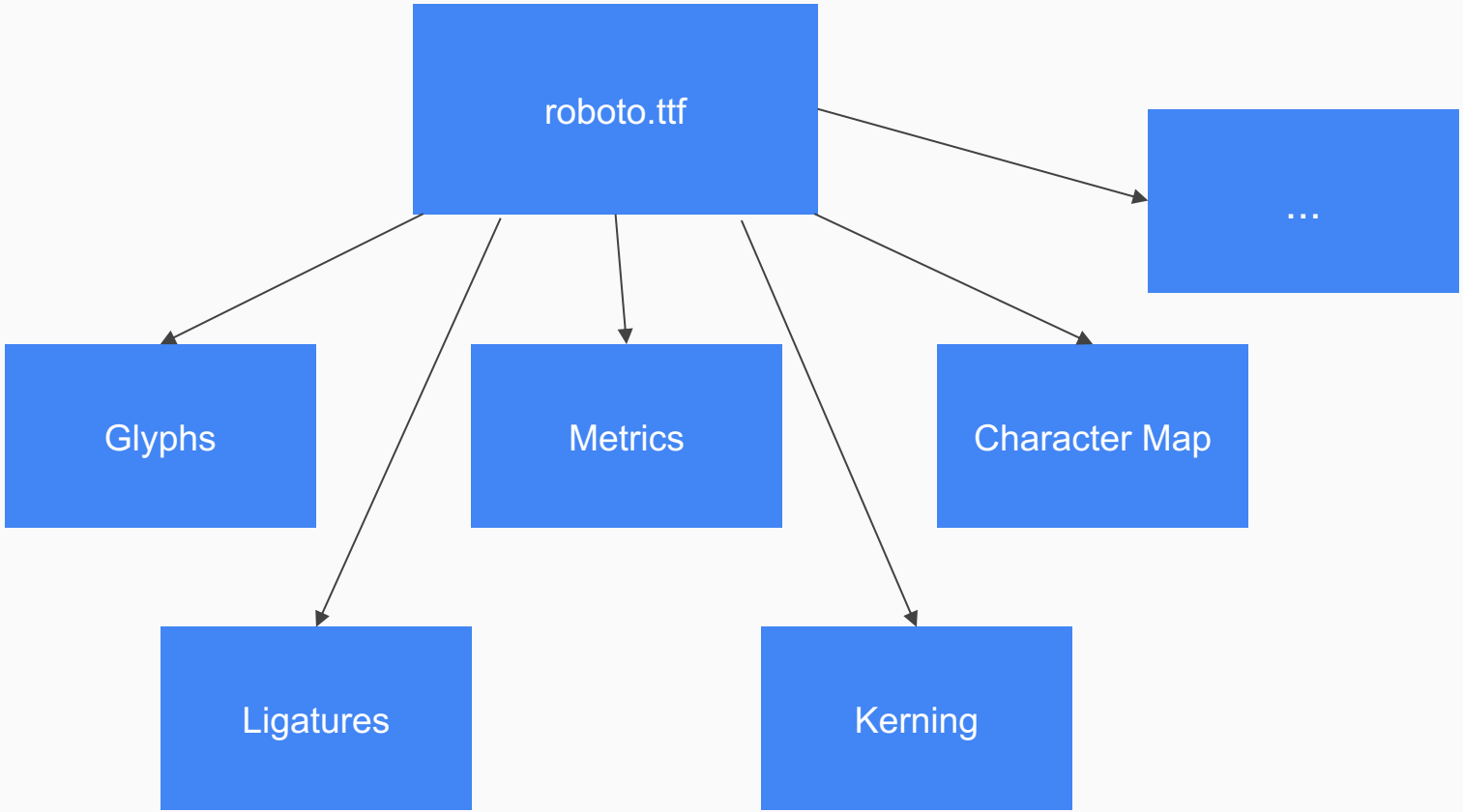
Шрифт / Font-face

1. Roboto Thin
2. Roboto Light
3. Roboto Regular
4. Roboto Medium
5. **Roboto Bold**
6. **Roboto Black**

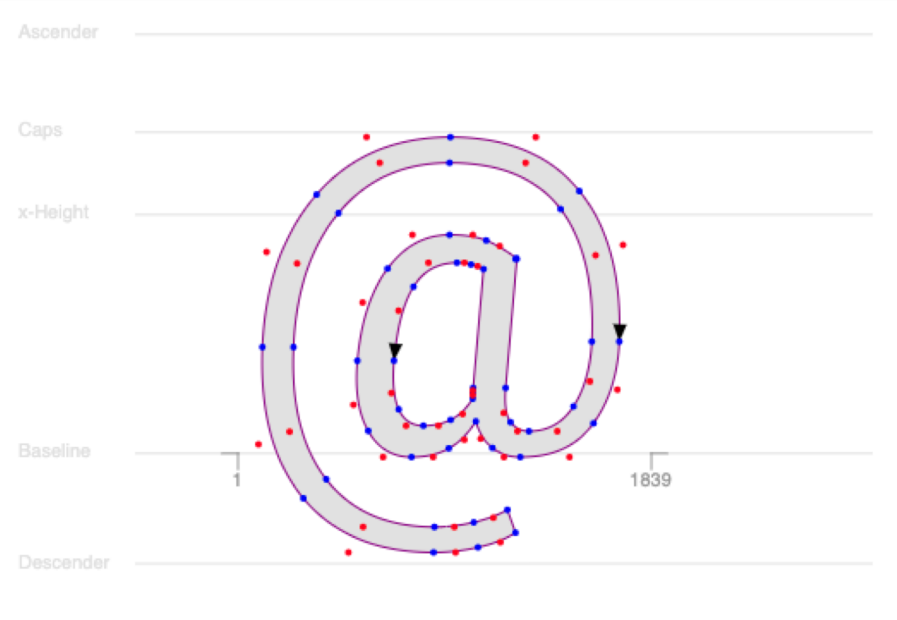
Файл шрифта / Font file

roboto-regular.ttf

Roboto.ttf

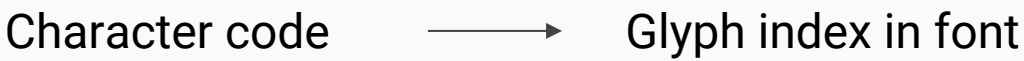

roboto.ttf
...
Glyphs
Metrics
Character Map
Ligatures
Kerning

Glyph



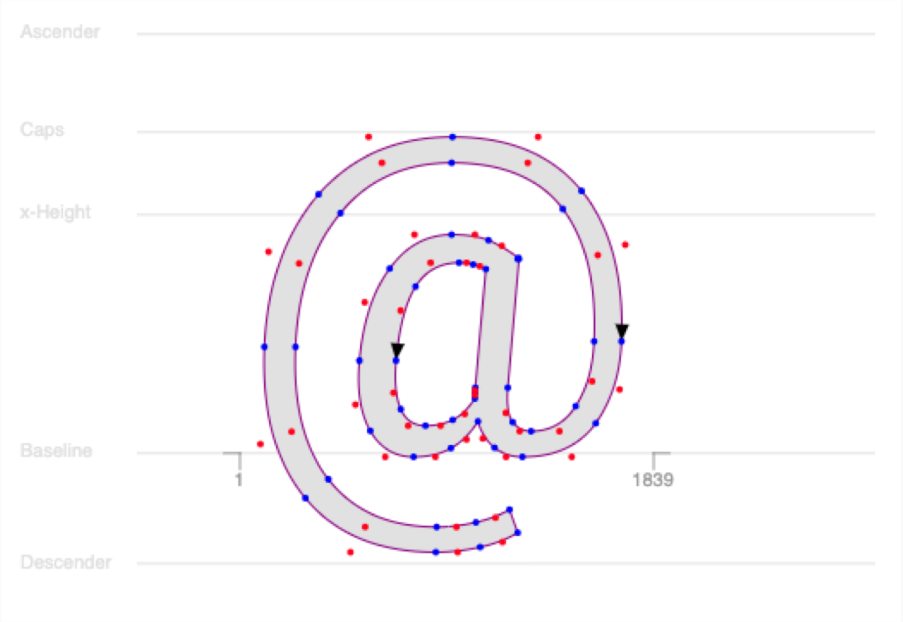
name	at
unicode	0040
index	34
xMin	97
xMax	1752
yMin	-453
yMax	1430
advanceWidth	1839
leftSideBearing	97

Карта символов / Character map

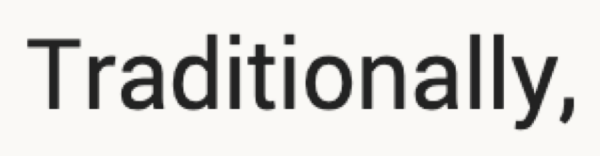


Метрики / Metrics

glyph placement, cursor advances, maximum glyph bounding box



Kerning: off



Kerning: on

Ligatures

prose fiction, non-fiction,

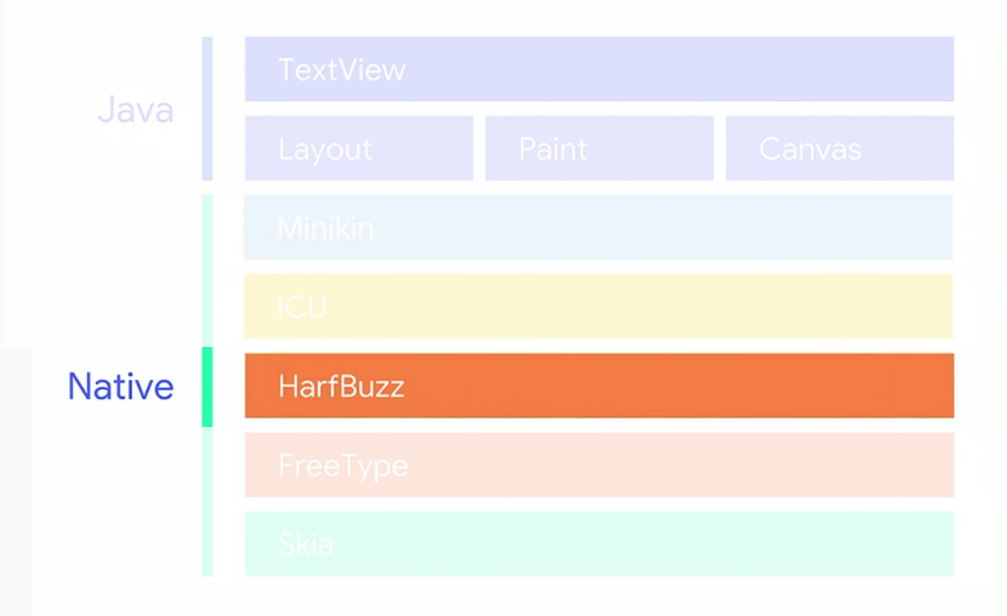
Ligatures

Текст — это не отдельные символы

HarfBuzz

Shaping — Conversion of text strings into positioned glyphs

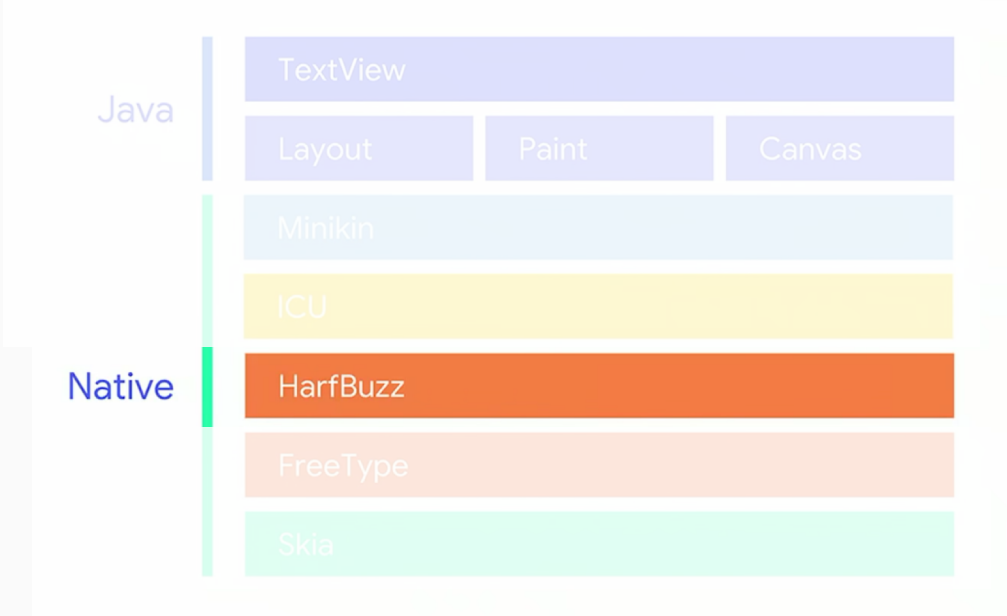
HarfBuzz is used in Android, Chrome, ChromeOS, Firefox, GNOME, GTK+, KDE, LibreOffice, OpenJDK, PlayStation, Qt, XeTeX, and other places



HarfBuzz

Shaping — Conversion of text strings into positioned glyphs

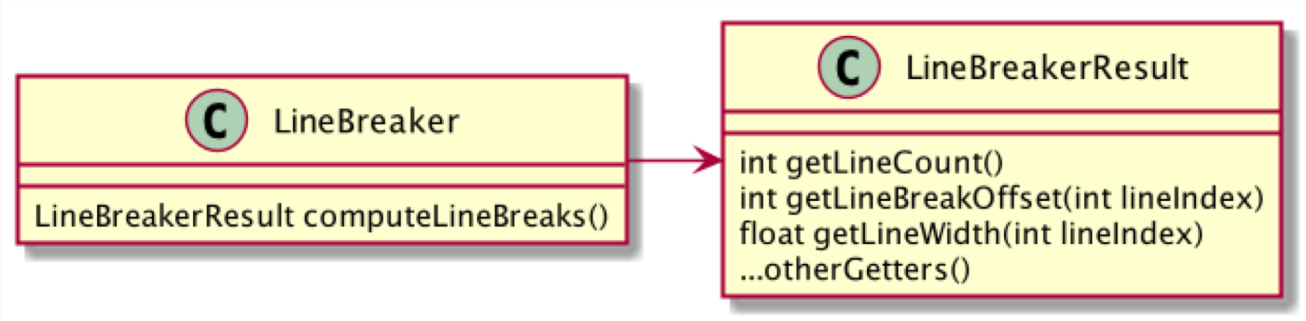
```
#include <hb.h>


void
hb_shape (hb_font_t *font,
          hb_buffer_t *buffer,
          const hb_feature_t *features,
          unsigned int num_features);
```



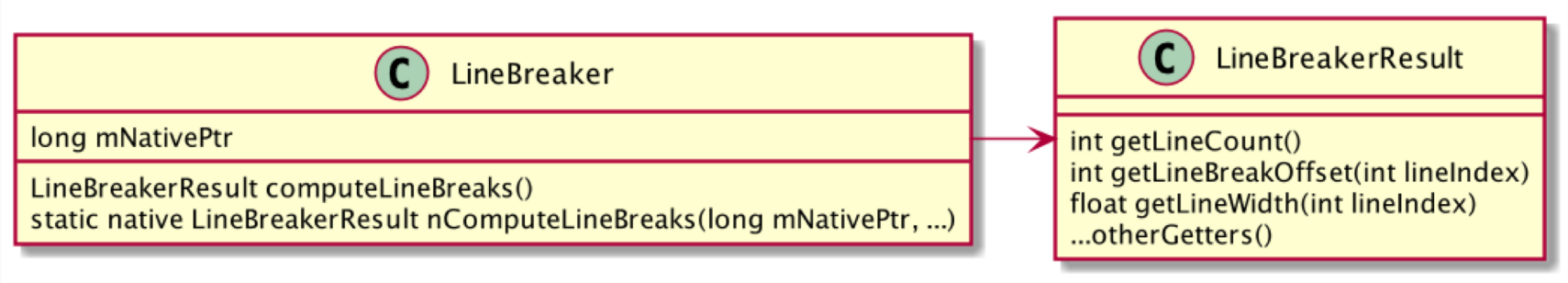
Text breaking & **measurement**

Text **breaking** & measurement

LineBreaker



LineBreaker



LineBreaker

core / jni / android / graphics / text / **LineBreaker.cpp**

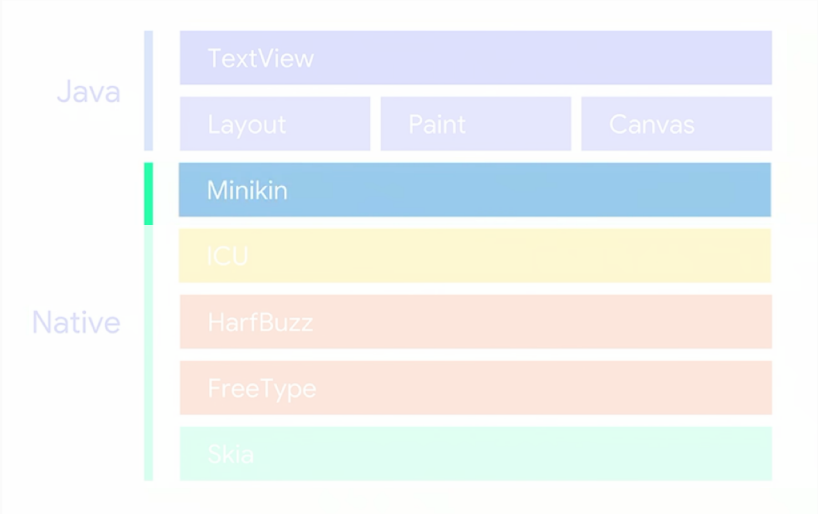
```
int register_android_graphics_text_LineBreaker(JNIEnv* env) {
  return RegisterMethodsOrDie(env, "android/graphics/text/LineBreaker", gMethods,NELEM(gMethods));
}

{"nComputeLineBreaks", "(...)J", (void*) nComputeLineBreaks}
```

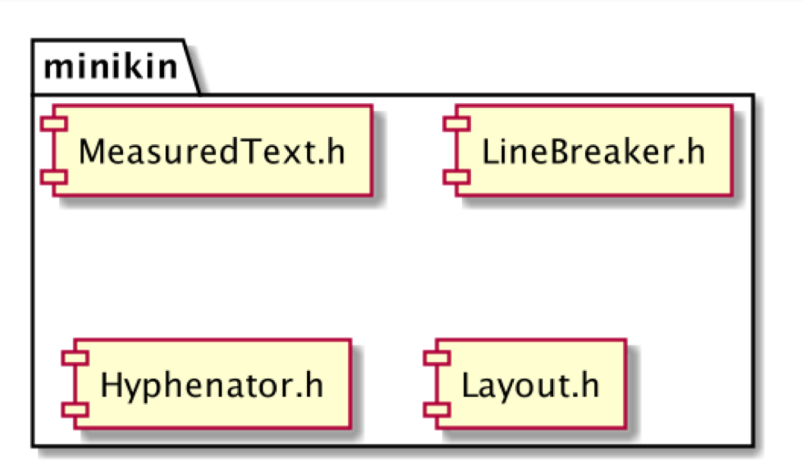
LineBreaker

core / jni / android / graphics / text / **LineBreaker.cpp**

```
static jlong nComputeLineBreaks(...){
  minikin::android::StaticLayoutNative* builder = toNative(nativePtr);
  ...
  ...
  builder->computeBreaks(...)
  ...
}
```

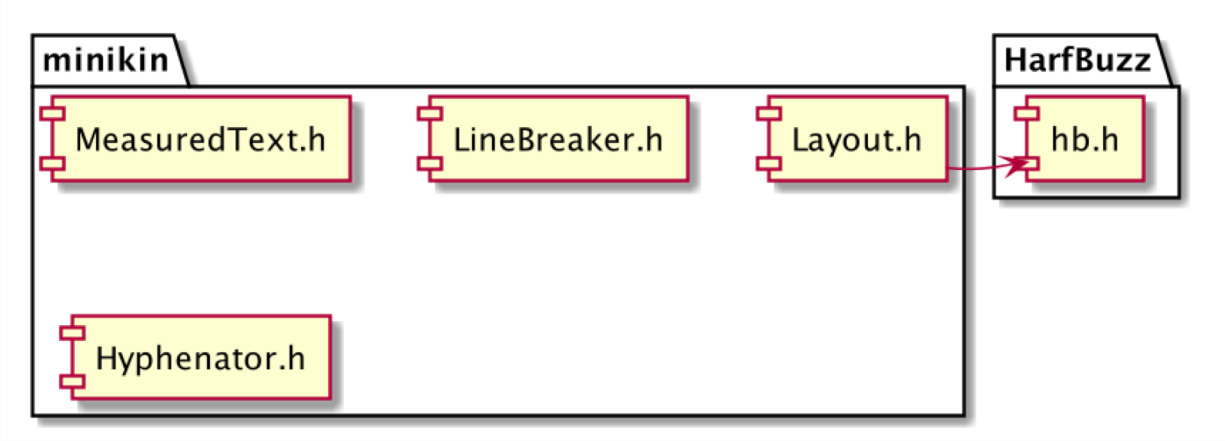
Minikin


Java
TextView
Layout
Paint
Canvas
Minikin
Native
ICU
HarfBuzz
FreeType
Skia


minikin
MeasuredText.h
LineBreaker.h
Hyphenator.h
Layout.h

Text **breaking** & measurement

HarfBuzz


minikin
MeasuredText.h
LineBreaker.h
Layout.h
Hyphenator.h
HarfBuzz
hb.h

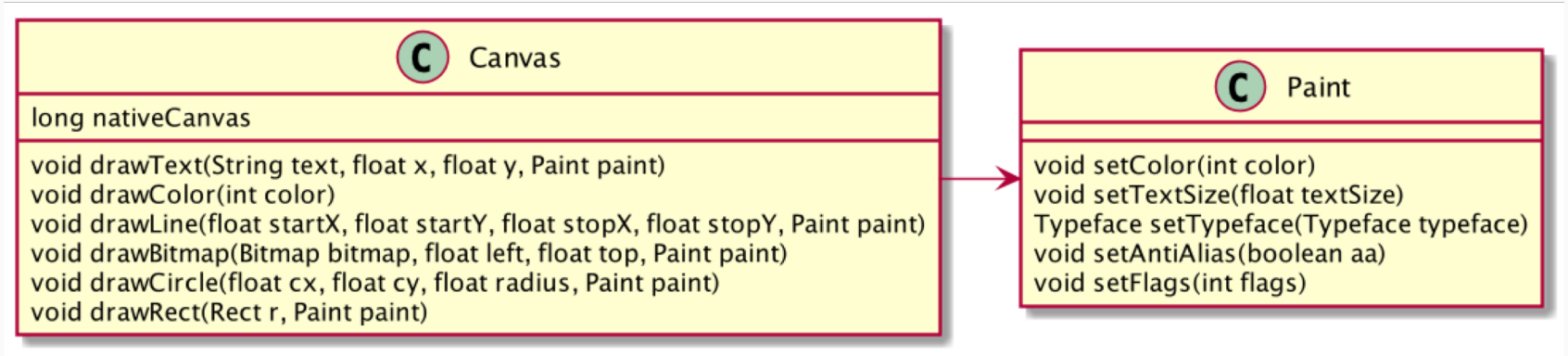
Text drawing

Text **drawing**

How to draw a rect


```
Paint paint = new Paint();
paint.setStyle(Paint.Style.FILL);
paint.setColor(Color.CYAN);
Rect rect = new Rect(0, 0, 10, 10);
canvas.drawRect(rect, paint);
```

android.graphics.Canvas



android.graphics.Canvas - Java

```
public void drawRect(float left, float top, float right, float bottom, @NonNull Paint paint) {
    ...
    nDrawRect(mNativeCanvasWrapper, left, top, right, bottom, paint.getNativeInstance());
}
```

```
private static native void nDrawRect(long nativeCanvas, float left, float top, float right,
        float bottom, long nativePaint);
```

android_graphics_Canvas.cpp - JNI

```
int register_android_graphics_Canvas(JNIEnv* env) {
    int ret = 0;
    ret |= RegisterMethodsOrDie(env, "android/graphics/Canvas", gMethods, NELEM(gMethods));
    ...
    return ret;
}
```

android_graphics_Canvas.cpp - JNI

```
int register_android_graphics_Canvas(JNIEnv* env) {
    int ret = 0;
    ret |= RegisterMethodsOrDie(env, "android/graphics/Canvas", gMethods, NELEM(gMethods));
    ...
    return ret;
}


static const JNINativeMethod gDrawMethods[] = {
    ...
    {"nDrawRect","(JFFFFJ)V", (void*) CanvasJNI::drawRect},
    ...
    };
```

android_graphics_Canvas.cpp - JNI

```
int register_android_graphics_Canvas(JNIEnv* env) {
    int ret = 0;
    ret |= RegisterMethodsOrDie(env, "android/graphics/Canvas", gMethods, NELEM(gMethods));
    ...
    return ret;
}


static const JNINativeMethod gDrawMethods[] = {
    ...
    {"nDrawRect","(JFFFFJ)V", (void*) CanvasJNI::drawRect},
    ...
    };
```

android_graphics_Canvas.cpp - JNI

```
static void drawRect(JNIEnv* env, jobject, jlong canvasHandle, jfloat left, jfloat top,
                     jfloat right, jfloat bottom, jlong paintHandle) {
  const Paint* paint = reinterpret_cast<Paint*>(paintHandle);
  get_canvas(canvasHandle)->drawRect(left, top, right, bottom, *paint);
}

static Canvas* get_canvas(jlong canvasHandle) {
  return reinterpret_cast<Canvas*>(canvasHandle);
}
```

hwui/Canvas.h - HWUI

HWUI - внутренний нативный графический фреймворк Android’a

```
virtual void drawRect(float left, float top, float right, float bottom, const SkPaint& paint) = 0;
```

hwui/SkiaCanvas.h/.cpp - SKIA

```
SkCanvas* mCanvas;


void SkiaCanvas::drawRect(float left, float top, float right, float bottom, const SkPaint& paint) {
    if (CC_UNLIKELY(paint.nothingToDraw())) return;
    mCanvas->drawRect({left, top, right, bottom}, *filterPaint(paint));
}
```

Skia

Skia

OSS, maintained by Google, BSD

Skia

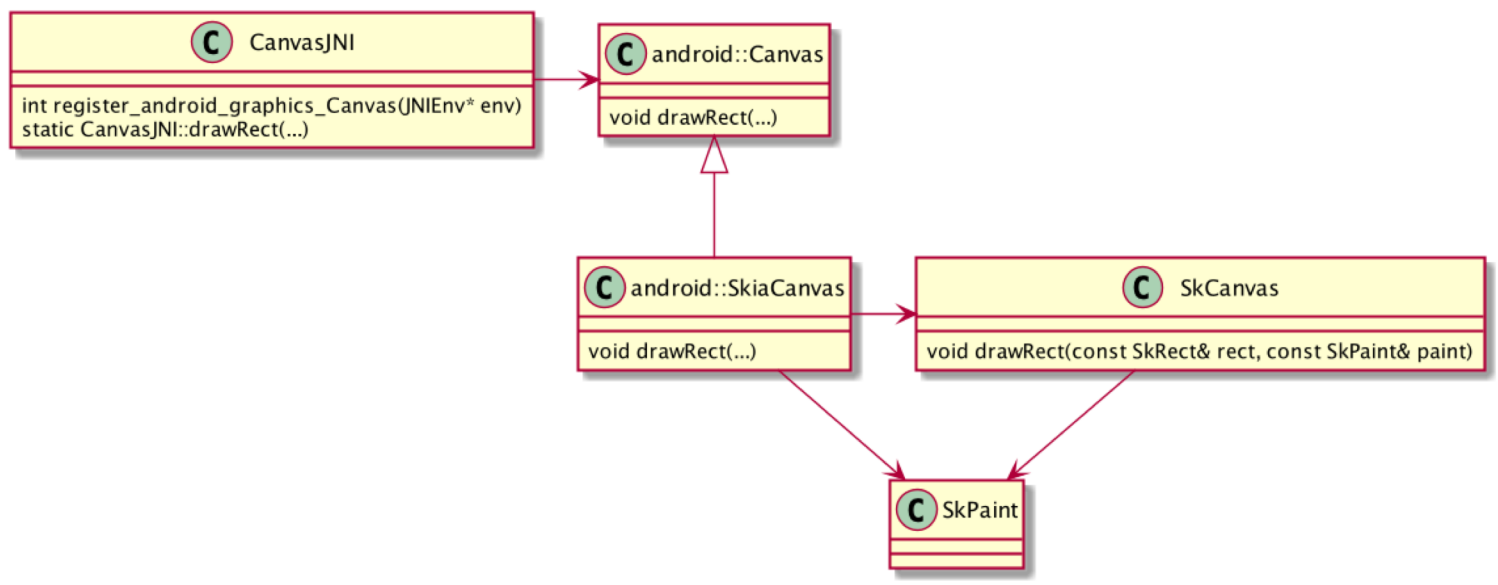
Skia

- SkAutoCanvasRestore - Canvas save stack manager
- SkBitmap - two-dimensional raster pixel array
- SkBlendMode - pixel color arithmetic
- SkCanvas - drawing context
- SkColor - color encoding using integer numbers
- SkColor4f - color encoding using floating point numbers
- SkFont - text style and typeface
- SkImage - two dimensional array of pixels to draw
- SkImageInfo - pixel dimensions and characteristics
- SkIPoint - two integer coordinates
- SkIRect - integer rectangle
- SkMatrix - 3x3 transformation matrix
- SkPaint - color, stroke, font, effects
- SkPath - sequence of connected lines and curves
- SkPicture - sequence of drawing commands
- SkPixmap - pixel map: image info and pixel address
- SkPoint - two floating point coordinates
- SkRRect - floating point rounded rectangle
- SkRect - floating point rectangle
- SkRegion - compressed clipping mask
- SkSurface - drawing destination
- SkTextBlob - runs of glyphs
- SkTextBlobBuilder - constructor for runs of glyphs

Skia

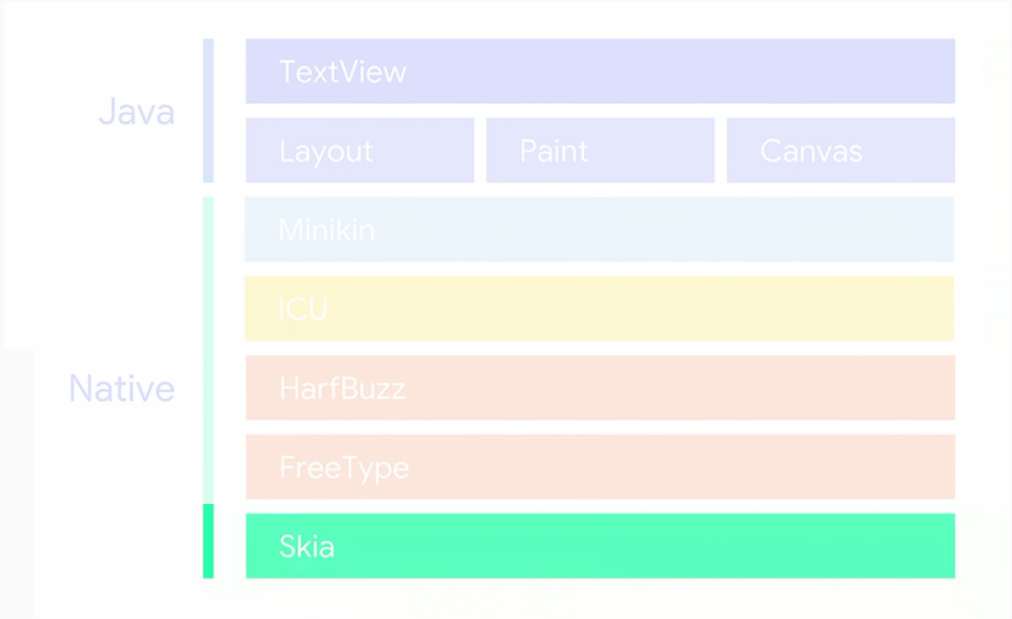
- SkAutoCanvasRestore - Canvas save stack manager
- SkBitmap - two-dimensional raster pixel array
- SkBlendMode - pixel color arithmetic
- **SkCanvas - drawing context**
- SkColor - color encoding using integer numbers
- SkColor4f - color encoding using floating point numbers
- SkFont - text style and typeface
- SkImage - two dimensional array of pixels to draw
- SkImageInfo - pixel dimensions and characteristics
- SkIPoint - two integer coordinates
- SkIRect - integer rectangle
- SkMatrix - 3x3 transformation matrix
- **SkPaint - color, stroke, font, effects**
- SkPath - sequence of connected lines and curves
- SkPicture - sequence of drawing commands
- SkPixmap - pixel map: image info and pixel address
- SkPoint - two floating point coordinates
- SkRRect - floating point rounded rectangle
- SkRect - floating point rectangle
- SkRegion - compressed clipping mask
- SkSurface - drawing destination
- SkTextBlob - runs of glyphs
- SkTextBlobBuilder - constructor for runs of glyphs

How to draw a rect



Skia


SKIA

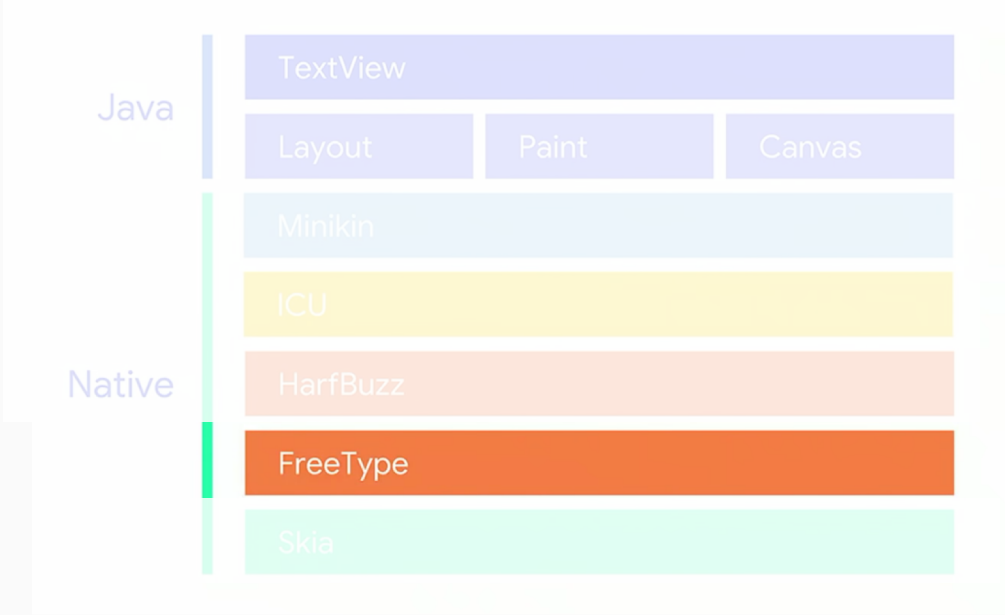

Java
TextView
Layout
Paint
Canvas
Native
Minikin
ICU
HarfBuzz
FreeType
Skia

Text **drawing**

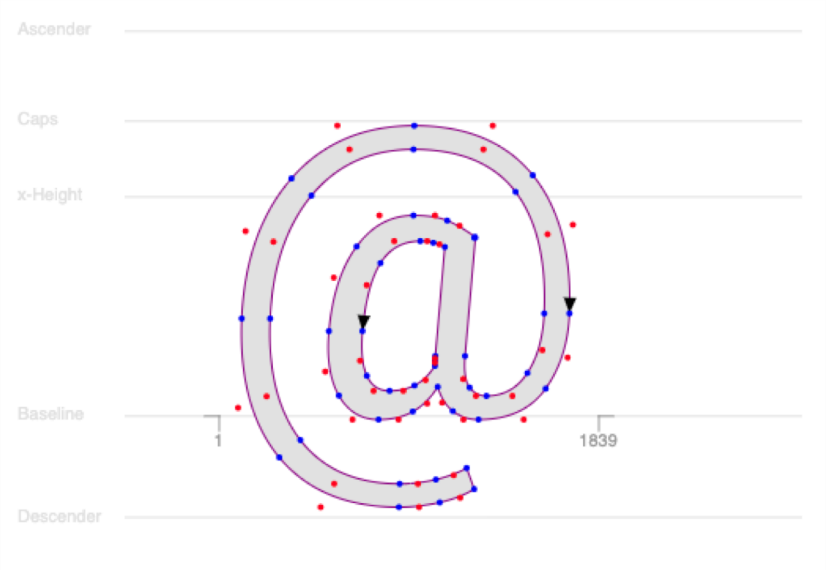
Text drawing

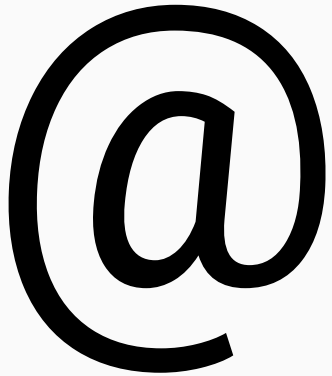
FreeType

FreeType is a freely available software library to render fonts



FreeType





FreeType

```
#include <ft2build.h>
#include FT_FREETYPE_H

FT_Error error;

error = FT_Load_Glyph( face, glyph_index, FT_LOAD_DEFAULT );
if ( error )
  return;

error = FT_Render_Glyph( face->glyph, FT_RENDER_MODE_NORMAL );
if ( error )
  return;

my_draw_bitmap( &slot->bitmap, slot->bitmap_left, slot->bitmap_top );
```

FreeType

```
#include <ft2build.h>
#include FT_FREETYPE_H

FT_Error error;

error = FT_Load_Glyph( face, glyph_index, FT_LOAD_DEFAULT );
if ( error )
  return;

error = FT_Render_Glyph( face->glyph, FT_RENDER_MODE_NORMAL );
if ( error )
  return;

my_draw_bitmap( &slot->bitmap, slot->bitmap_left, slot->bitmap_top );
```

FreeType

```
#include <ft2build.h>
#include FT_FREETYPE_H

FT_Error error;

error = FT_Load_Glyph( face, glyph_index, FT_LOAD_DEFAULT );
if ( error )
  return;

error = FT_Render_Glyph( face->glyph, FT_RENDER_MODE_NORMAL );
if ( error )
  return;

my_draw_bitmap( &slot->bitmap, slot->bitmap_left, slot->bitmap_top );
```

FreeType

```
#include <ft2build.h>
#include FT_FREETYPE_H

FT_Error error;

error = FT_Load_Glyph( face, glyph_index, FT_LOAD_DEFAULT );
if ( error )
  return;

error = FT_Render_Glyph( face->glyph, FT_RENDER_MODE_NORMAL );
if ( error )
  return;

my_draw_bitmap( &slot->bitmap, slot->bitmap_left, slot->bitmap_top );
```

FreeType

```
#include <ft2build.h>
#include FT_FREETYPE_H

FT_Error error;

error = FT_Load_Glyph( face, glyph_index, FT_LOAD_DEFAULT );
if ( error )
  return;

error = FT_Render_Glyph( face->glyph, FT_RENDER_MODE_NORMAL );
if ( error )
  return;

my_draw_bitmap( &slot->bitmap, slot->bitmap_left, slot->bitmap_top );
```

Стек компонентов

This **is** a styled *text*

Все немного сложнее...

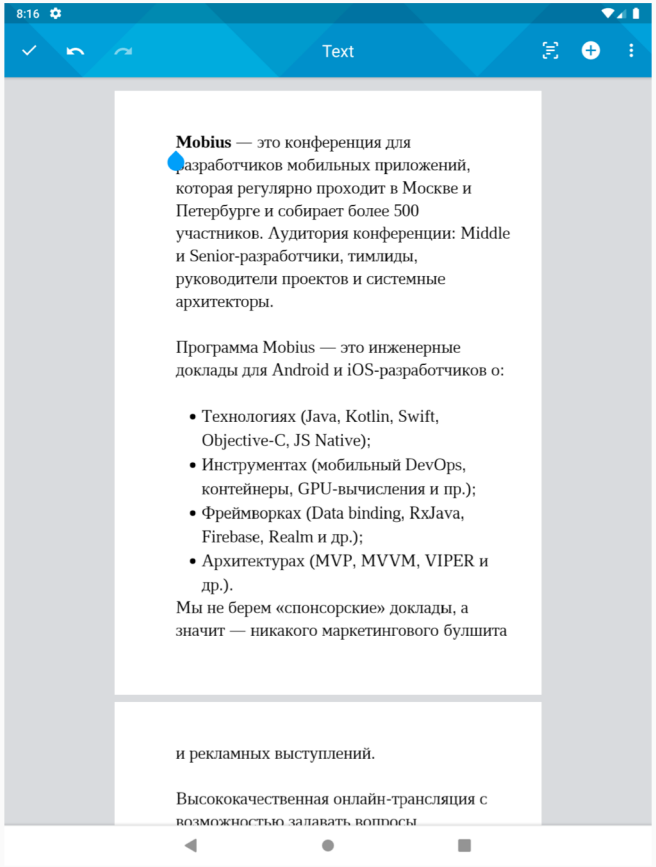
- Аппаратное ускорение
- Растеризация глифов на GPU
- Кеширование глифов
- Кеширование измерений текста,
- ...

Все немного сложнее...

Как это может пригодиться?

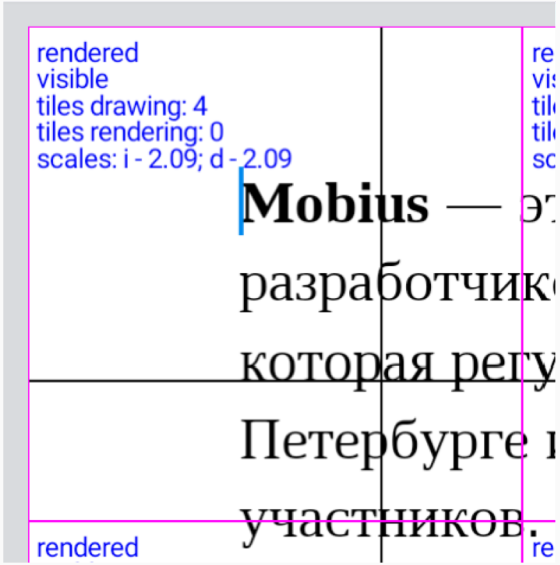
Кроме того, что это просто интересно

Рендеринг текста в приложении МойОфис



- Кросс-платформенное ядро на C++
- “Родной” UI на каждой платформе
- Бизнес-логика рендеринга документа расположена на стороне JVM
- Ресурсоемкая логика расположена на стороне C++

Рендеринг текста в приложении МойОфис



- Кросс-платформенное ядро на C++
- “Родной” UI на каждой платформе
- Бизнес-логика рендеринга документа расположена на стороне JVM
- Ресурсоемкая логика расположена на стороне C++
- SKIA + NDK + JVM

Спасибо

twitter.com/E13Mort