绘图思路

* 一次将所有数据都绘制完，外层嵌套一个scrollview以实现滚动效果。局限：对于view的宽度有限制，适合一屏绘制很多数据但是总宽度不长的情景下
  + 具体的绘制思路：绘制前预先计算所需view的宽度-🡪OnDraw()，若需要再次加载数据，则接收到数据后需要重新计算宽度（requestLayout），根据最新的宽度进行绘制。
  + 该绘制方法依赖于scrollview的滚动，不适合在当前视图上做手势缩放等操作
* 每次只绘制当前屏幕的数据，根据滚动后的偏移量实时绘制当前显示区域的数据。局限：每次滚动都会触发重绘，非常耗能（要求尽量减少循环次数，并优化查找排序等算法，最好不要再重绘的时候创建对象），适合绘制一屏数据不多的情景。
  + 具体的绘制思路：重绘的时候只获取显示区域的数据，将数据转换到屏幕中的像素值坐标

**实例**（基于第二种思路，具体需要根据实际场景选择）

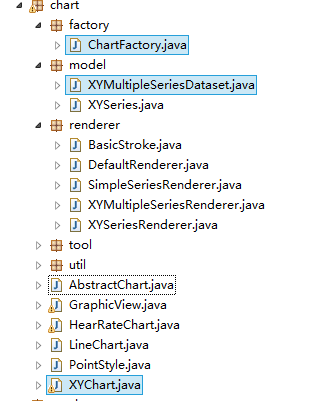
Y

X

A

绘图中需要注意的上图中的5个点，左上角为坐标原点（0，0）,当然在视图偏移后，视图上左上角不再是恒定的，而是坐标（getScrollX , getScrollY），根据这些可以将实际的数据转换到屏幕中实际的像素坐标。

初步封装后的折线图：



ChartFactory : 生成折线图的工厂类

XYMultipleSeriesDataSet:可以看作是所需要绘制线的容器，包含待绘制的序列（折线，点等）

XYSeries:包含一条线的所有数据，负责提供给画板

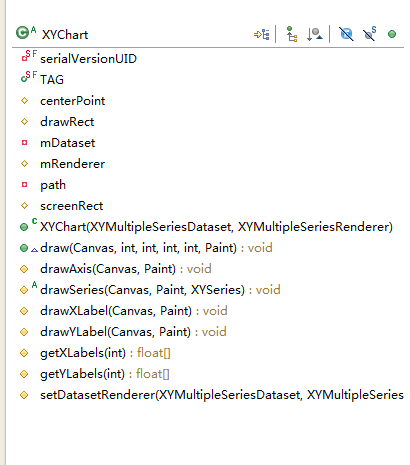
XYMultipleRenderer:渲染器，负责渲染整幅波形图

XYSeriesRenderer:渲染器，渲染单一一条线，一个series对应一个XYSeriesRenderer

XYChart:承担整个绘图的逻辑

GraphicalView:封装了自定义的图形，包括缩放活动等事件的处理

具体的绘制方法



XYChart封装了部分绘制方法，需要自定义则只要重写这些方法即可。

每次绘制只绘制当前一屏的内容，需要从series中获取到当前一屏对应的数据之后进行绘制

drawRect封装了波形绘制的坐标(除去四周的间距)

screenRect封装了当前的屏幕的坐标（包含间距）

绘制过程中主要根据一屏需要绘制的点的个数 将其转换成屏幕的像素后，绘制在指定的区域

drawRect

getScrollX

getScrollX+Width

