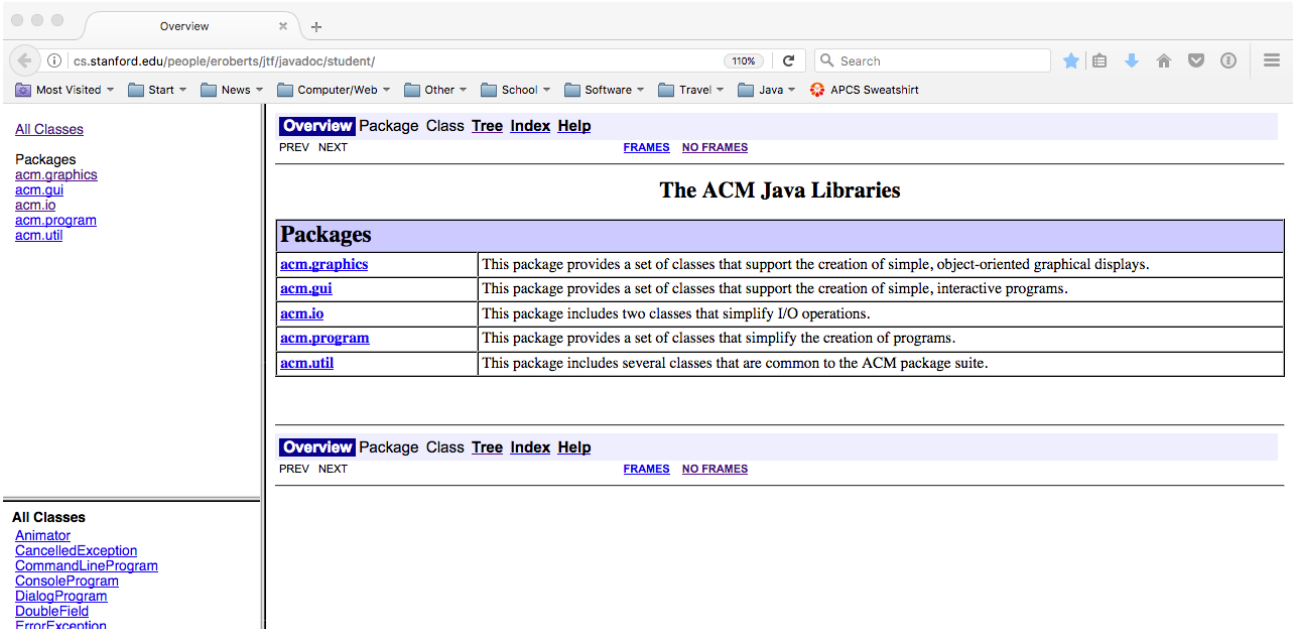
SimpleGraphics.java

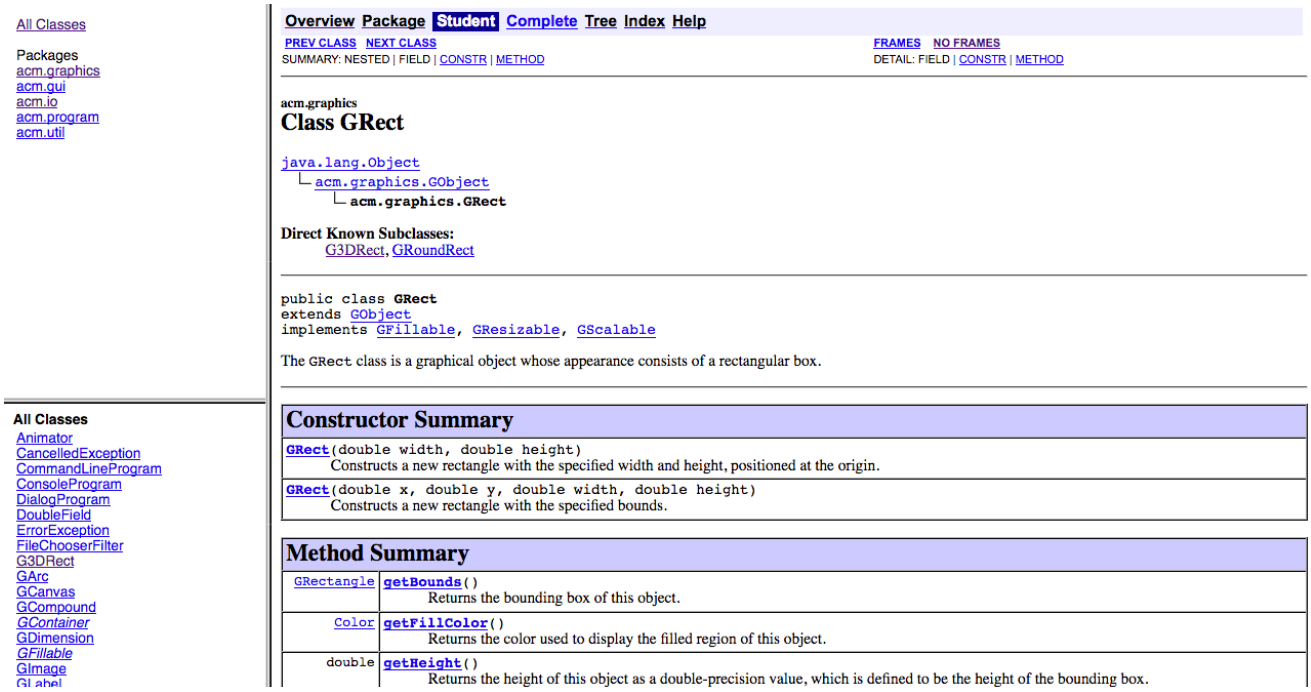
Objective: To learn how to reference an Application Programming Interface (API) and use it to design a simple graphics program.

Background:
In our **FirstAssignment.java** project we learned that ACM (Association for Computing Machinery) has provided a wonderful library of Java classes that make implementing graphics much easier. The file containing these classes is called **mvAcm.jar** and will be provided in the project zip file. To do our work, we need a "how to program" document for properly using ACM's classes.

The document is online and called the API (Application Program Interface). The ACM API web page can be found at (http://cs.stanford.edu/people/eroberts/jtf/javadoc/student/) and it looks like this:



The **mvAcm.jar** file contains "packages" and each package contains a specific group of similar functions. For example, the **acm.graphics** package contains a list of classes that will draw simple graphical objects. If you click on **acm.graphics** , you are presented with a list of classes like GArc, GLabel, GRect, etc. which create their respective graphical object. Click on **GRect** and you get the following page:



All Classes

Packages
acm.graphics
acm.gui
acm.io
acm.program
acm.util

Overview Package Student Complete Tree Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

acm.graphics
Class GRect

java.lang.Object
└ acm.graphics.GObject
└ acm.graphics.GRect

Direct Known Subclasses:
G3DRect, GRoundRect

public class GRect
extends GObject
implements GFillable, GResizable, GScalable

The GRect class is a graphical object whose appearance consists of a rectangular box.

All Classes
Animator
CancelledException
CommandLineProgram
ConsoleProgram
DialogProgram
DoubleField
ErrorException
FileChooserFilter
G3DRect
GArc
GCanvas
GCompound
GContainer
GDimension
GFillable
GImage
GLabel

Constructor Summary

GRect(double width, double height)
Constructs a new rectangle with the specified width and height, positioned at the origin.

GRect(double x, double y, double width, double height)
Constructs a new rectangle with the specified bounds.

Method Summary

GRectangle	getBounds() Returns the bounding box of this object.
Color	getFillColor() Returns the color used to display the filled region of this object.
double	getHeight() Returns the height of this object as a double-precision value, which is defined to be the height of the bounding box.

The ***Constructor Summary*** shows you two different ways to create a GRect rectangle. The first constructor is passed the width and height of the rectangle:

```
GRect rectangle1 = new GRect (width, height);
```

When added to the window, the rectangle will be located at the origin (0,0). The second constructor specifies both the (x,y) coordinate location and the width and height.

The ***Method Summary*** contains all of the methods that the GRect object may call. For example, you can get the height of the rectangle:

```
double height = rectangle1.getHeight();
```

Your job will be to use the API reference pages and your prior programming knowledge to complete the following assignment.

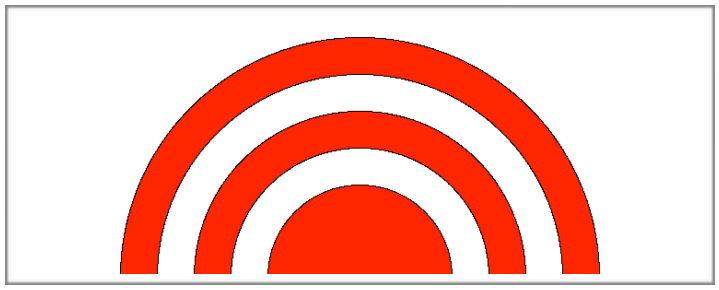
Assignment:
Download the **SimpleGraphics.zip** file from the web site and unzip. It will create the directory **SimpleGraphics**. Do all your work in that directory.

1. A sample program called **SimpleGraphics.java** is provided. Examine the code using Geany. The program draws a simple red circle and blue rectangle on a white background. You can compile and run the **SimpleGraphics.java** file provided using these commands:

   ```
   % javac -cp .:mvAcm.jar SimpleGraphics.java
   % java -cp .:mvAcm.jar SimpleGraphics
   ```

   You will use this code as a template to construct the following two exercises. You may delete and add code as you see fit.

2. Copy all of the code in **SimpleGraphics.java** to a new file and call it **Target.java**. This exercise is to draw the following semicircular figure:

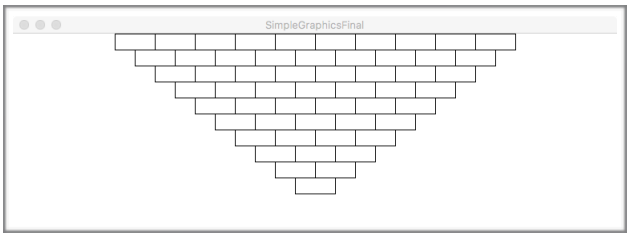


The graphic is created with a series of five alternating red/white concentric circles (**GOval**'s) drawn with their centers at the bottom center of the window. Only half of the circles appear in the window since the centers are at the bottom edge of the window.

Hints:

1) Draw the largest circle first, then the next largest and so on.
2) Use a one-dimensional array for the GOval's.

3. Copy all of the code in **SimpleGraphics.java** to a new file and call it **Bricks.java**. In this exercise your program will draw the following block figure:



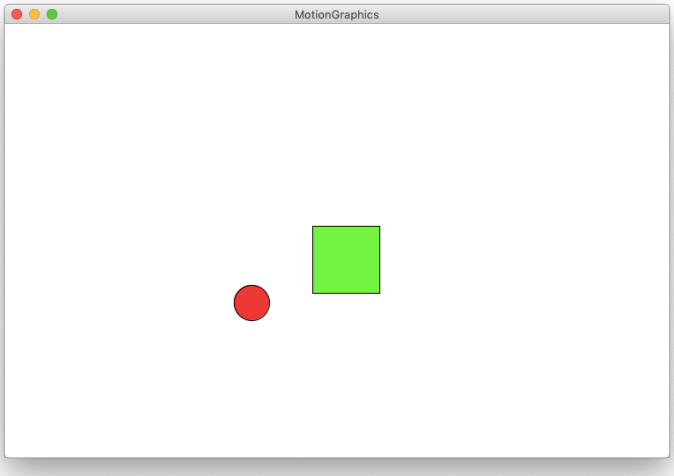
These are a series of rectangles (**GRect**'s) of height 20 and width 50 stacked in ten rows from the top of the window down in an inverted pyramid configuration. The pyramid should be centered in the window.

Hint: Use a one-dimensional array for the GRect's.

Challenge (not for credit)

You can add motion to your graphics by looping commands in the **run** method. You can see a demonstration of graphic motion with the **MotionGraphics.class** bytecode file. Use this command to execute the code:

```
% java -cp .:mvAcm.jar MotionGraphics
```



Click on the window and the graphics will move. Click again and they stop. The circle bounces around the window like a pool ball. The green square grows and shrinks in size.

Your challenge is to create graphics that move. Adding mouse and keyboard functions is also challenging.

Show your results to Mr Greenstein.