

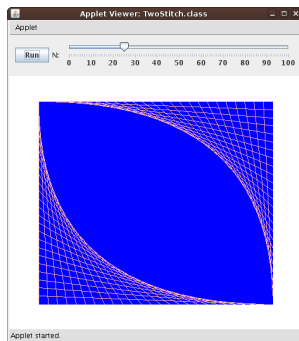
MAT 2170: Laboratory 5

Key Concepts

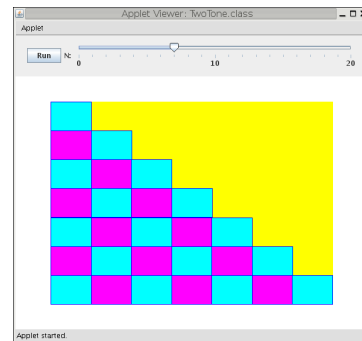
1. `for` & `while` loops
2. `if` statements
3. graphics, coordinate translation

The main purpose of this lab is to give you more practice using loops, and to introduce you to using `if` statements. As usual, create these projects in a `mat2170/lab5` directory.

Exercises

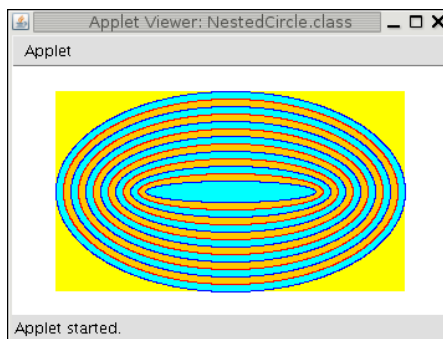


Exercise 1. Double Curve Stitch

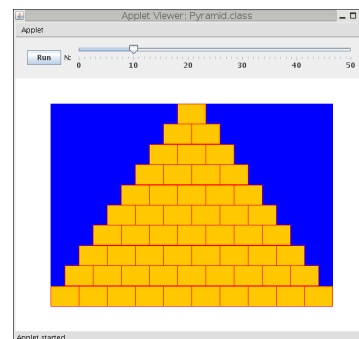


Exercise 2. Two-Tone Triangle

1. (Curve Stitch, Revisited) In this new version, display a rectangle in the graphics window which is 80% of the width and height of the window and centered in it, then add the curve stitch to the bottom left and upper right corners. Do not use magic numbers for the dimensions of the rectangle — calculate them using 80% of `getWidth()` and `getHeight()`. A blue background with pink stitches is quite lovely, but you may choose your own colors as long as everything shows up and isn't just black and white. Your solution is to be a slider program with a slider range of 3 to 50.
2. (Lower-left Triangle, Revisited) As with the Curve Stitch exercise above, display the triangle in a centered rectangle that is 80% of the width and height of the graphics window. Your solution is to be a slider program with a slider range of 1 to 20, and graphics window size of 600 (width) by 500 (height). Your final version is to output a triangle with the number of rows and columns determined by the slider. The blocks should alternate colors: a yellow background with cyan or magenta blocks with blue outlines may be just the thing. Use an `if` statement to set the colors.



Exercise 3. Nested Ovals



Exercise 4. Pyramid

3. (Nested Ovals) This is a variation of the **Target** project, but you are required to use a loop to control the number of stacked or nested ovals. Fit the nested ovals in a centered rectangle that is 80% of the graphics window's width and height. Each oval should be smaller by a *step* that is 6% of the **smaller of the width or height** of the background rectangle. The colors of the ovals are to alternate: perhaps a yellow background with cyan (with blue border) and orange (with red border) ovals. The drawing of ovals is to stop when either the width or the height of the oval to be displayed is "too small," i.e., when there's no more room to take another step. Use an **if** statement to set the colors. No slider necessary for this project, so just extend the **GraphicsProgram**.
4. (Pyramid) This exercise is a slightly modified version of one from your textbook. A stack of blocks should fit "exactly" into a centered rectangle that is 80% of the graphics window's width and height. Use a slider program with a slider range of 1 to 50, where the slider indicates the number of rows to be placed in the rectangle. Set the graphics window size to 600×500 . Thus, in this exercise (unlike the book's), the width and height of each brick is directly related to the size of the window. You may start at either the top or bottom of the pyramid – it can be built in either direction. Make the background rectangle blue and the bricks orange with red outlines.

Finishing Up

When you have completed the lab,

1. Print each of your programs
2. Submit an electronic copy of lab5
3. Publish these programs to your web site
4. Complete the Postlab worksheet, staple your printouts to it, and hand it in at the beginning of Lab 6.

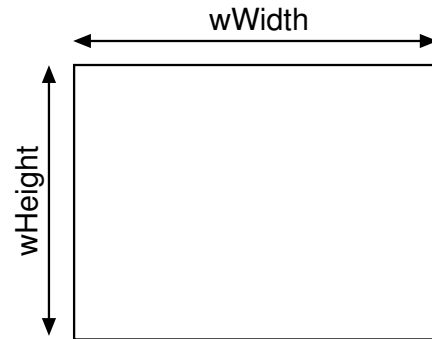
Prelab

1. All of the programs this week require a centered background rectangle which is 80% of the width and height of the graphics window.

- (a) In the graphics window at right, draw a background rectangle about 80% of the window's width and height, centering it. Mark the position of the rectangle with a \star . What size are the left and top margins?

left margin is:

top margin is:



- (b) Complete the code fragment below to determine the background's width and height and display it centered in the window.

```
double wWidth = getWidth(); // get the window's width
double wHeight = getHeight(); // get the window's height
// Calculate background rectangle's width and height
double backWidth =

double backHeight =

// Determine position of background rectangle (centered in window)
double backX =

double backY =

// Create and display a blue background rectangle
GRect background = new GRect(

background.setFilled(true);
background.setColor(
add(background);
```

2. The curve stitch problem in last week's lab began with the endpoints of the first line used to form the curve flush to the left side of the window. In that case:

```
xTop = 0.0;    yTop = 0.0;    xBottom = 0.0;    yBottom = getHeight();
```

This week, the curve must fit in the background rectangle, as created in the first exercise above, and you'll need two lines – one flush to the left side, the other to the top of the background.

- (a) Give the code to determine the left and right x values for the endpoints of the lines of the curve stitch program:

```
double xLeft =
```

```
double xRight =
```

- (b) Give the code to determine the top and bottom y values for the endpoints of the lines of the curve stitch program:

```
double yTop =
```

```
double yBottom =
```

- (c) Give the code to determine `dx` and `dy`, the change in the `x` and `y` coordinates of the endpoints, given a slider value of `n`:

```
double dx =
```

```
double dy =
```

3. In the two-tone triangle problem, what boolean expression can be used to determine the color of the current `block`?
4. In the nested ovals problem, the next concentric oval is to shrink in width and height by `step`, which is to be set to 6% of the smaller of the width or height of the background rectangle. Complete the following Java code fragment to determine the value of `step`:

```
double wWidth = getWidth();  
double wHeight = getHeight();  
double step;
```

Lab 5— Postlab

Name: _____

Detach this page, staple your printouts to it, and turn in at the beginning of Lab 6

1. Complete the algorithm below to create a two-toned triangle of n rows in the **upper-right** corner of a rectangle 80% of the width and height of the window and centered in it. The triangle should span the width and height of the rectangle (not the window).

Let N be the value of the slider, `getN()`

Let `wWidth` and `wHeight` be the window's width (`getWidth()`) and height (`getHeight()`)

background width (`backWidth`) is $0.8 * wWidth$

background height (`backHeight`) is $0.8 * wHeight$

`leftX` of background rectangle is $(wWidth - backWidth) / 2.0$

`topY` of background rectangle is $(wHeight - backHeight) / 2.0$

create and draw background rectangle

`blockWidth` is $backWidth / N$

`blockHeight` is $backHeight / N$

for row starts at _____ as long as _____ update: `row++`

 for col starts at _____ as long as _____ update: `col++`

 calculate position of current block:

 X of block is _____

 Y of block is _____

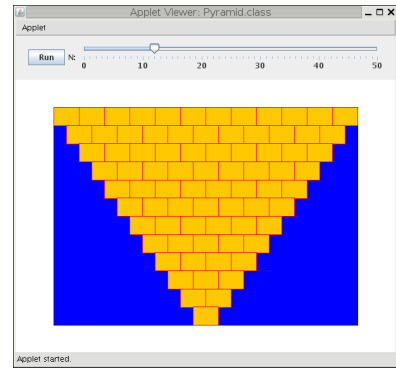
 create block at X, Y with `blockWidth`, `blockHeight`

 set block filled, color

 draw block

continued on back

2. Complete the algorithm below in a similar manner to the first exercise (on reverse side of this sheet). The goal is to create an upside-down pyramid of N rows in a rectangle centered in a window (as illustrated at right). The pyramid should span the width and height of the rectangle (not the window). Extra room is left for additional statements.



Initialize N , $wWidth$, $wHeight$, $backWidth$, $backHeight$, $leftX$, $topY$ as before

create and draw background rectangle

$blockWidth$ is _____

$blockHeight$ is _____

for row starts at _____ as long as _____ update $row++$

for col starts at _____ as long as _____ update _____

calculate position of current block:

X of block is _____

Y of block is _____

create block at X, Y with $blockWidth$, $blockHeight$

set block filled, color

draw block

Update for next column:

Update for next row: