CSE 142, Autumn 2006
Programming Assignment #4: SimpleFigure and Targets (20 points)
Due: Tuesday, October 24, 2006, 2:00 PM

Program Description:

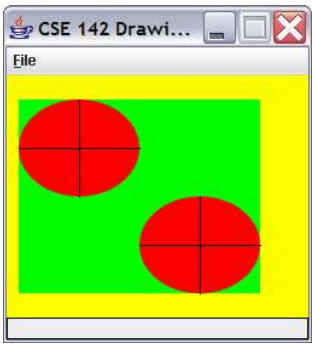
This assignment will give you practice with value parameters, using Java objects, and graphics. This assignment has 2 parts; you will turn in two Java files named `SimpleFigure.java` and `Targets.java`.

To compile and run this assignment, you must download the file `DrawingPanel.java` from the Assignments section of the class web page and save it in the same folder as your code. You do not have to turn in `DrawingPanel.java`.

Part A: SimpleFigure (4 points)

For the first part of this assignment, turn in a file `SimpleFigure.java` that draws a figure using the `DrawingPanel` provided in class. You may draw any figure you like that is at least 100 x 100 pixels, contains at least three shapes, uses at least two distinct colors, and is not highly similar to your figure for Part B. Be creative and draw any figure you like. After the assignment is turned in, the instructors will anonymously post student figures on the course web site.

If you do not want to create your own figure, at right is a default figure you may draw for Part A. If your code draws this figure, you will receive full credit for Part A. This figure contains a `DrawingPanel` of size 250 x 200 with a yellow background, a green rectangle with top-left corner at (10, 20) and size 200x160, red ovals with top-left corners at (10, 20) and (110, 100) of size 100x80, and black lines from (10, 60) to (110, 60), from (60, 20) to (60, 100), from (110, 140) to (210, 140), and from (160, 100) to (160, 180).

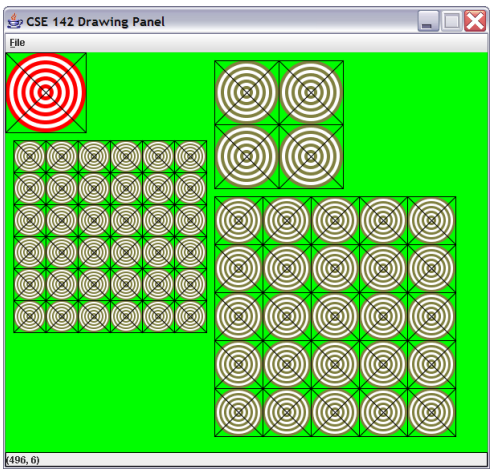


You may optionally wish to try using parameterized methods to capture the repetition in the figure. However, your score for Part A will be based solely on its external correctness as defined above; it will not be graded for internal correctness or style.

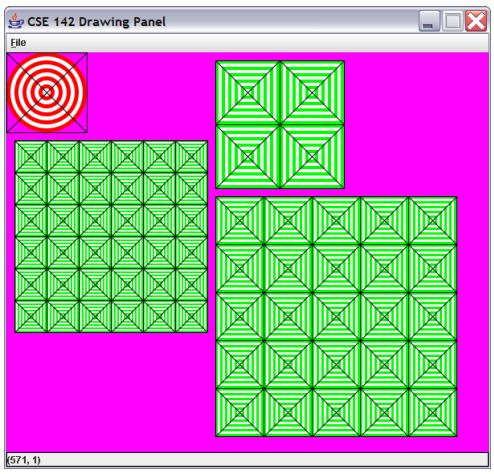
Part B: Targets (16 points)

The second part of this assignment asks you to turn in a file `Targets.java` that draws a specific complex figure of grids of targets. Your program should exactly reproduce the images on the next page. (The image in this document was taken when running the program on Windows; if you use another platform, the border around the window may look slightly different.)

```
Type the background/foreground colors and shape.
purple, green, or brown? green
purple, green, or brown? brown
circle or square? circle
```




```
Type the background/foreground colors and shape.
purple, green, or brown? purple
purple, green, or brown? green
circle or square? square
```



The Part B image has several levels of structure. There is a basic "target" subfigure that occurs throughout, which contains concentric circles or squares inside it. The target is repeated to form larger grid figures.

The overall drawing panel is size 600 x 500. Its background color is determined by user input and is either purple (magenta), green, or brown (Red=128, Green=128, Blue=64). Each target subfigure alternates between a certain foreground color and white. The foreground colors are also chosen from purple, green, or brown by the user. Each target also has a black rectangular outline and X lines drawn on it.

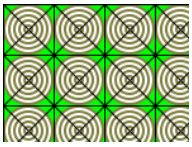
The four figures on your drawing panel should have the following properties.

Description	(x, y) position	color	size of targets	circles/target	rows/cols
top-left	(0, 0)	always red	100 x 100	10	1 x 1
bottom-left	(10, 110)	input by user	40 x 40	10	6 x 6
top-right	(260, 10)	input by user	80 x 80	10	2 x 2
bottom-right	(260, 180)	input by user	60 x 60	10	5 x 5

Implementation Guidelines for Part B:

To receive full credit on Part B, you should write static methods that use a great deal of parameter-passing and complex numeric computations. You are required to have two particular static methods described below. Because it is difficult to correctly implement such a complex program all at once, instead start with a smaller piece of the problem.

In particular, you should write a static method that draws one target; initially draw targets of concentric circles only. Start with the subfigure in the upper-left part of the screen. We suggest writing this code incrementally making incremental improvements. Your first version of this method could draw the specific subfigure in the upper-left corner, but you will want to generalize this with the use of parameters. Eventually you should be able to call it with different sizes, (x, y) locations, colors, and so on. Assume that every target's size will be a multiple of 20, since this means that each of your concentric circles will differ in size by an equal amount.



Once you have completed the static method that produces one of these subfigures, write another static method that produces a square grid of targets. You will call this method several different times from `main` to produce the grids of the overall figure. It will need a lot of parameters to be flexible enough to draw each of these grids. The key point is that a single method can be called three times to produce the three grids.

We recommend that you write your program to use fixed colors/shapes first and add the user interaction with Scanner last. Assume that the user enters valid input; the user will type a valid color or shape from the given choices in lowercase.

For full credit, both your graphical output and text output should match the examples provided, given the same user input.

Stylistic Guidelines:

For this assignment you are limited to the language features in Chapters 1 through 4; you are not allowed to use more advanced features to solve the problem.

Continue to use static methods to structure your solution as described previously; this time, write static methods that use parameters. In grading, we will require at least the two particular static methods named previously: one that draws a single target, and one to draw a grid of targets that is called many different times to produce the various figures on the screen. Use other methods as needed to capture structure and redundancy.

You are required to properly indent your code and will lose points if you make significant indentation mistakes. Give meaningful names to methods and variables in your code. Follow Java's naming standards as specified in Chapter 1 of the textbook. Localize variables whenever possible -- that is, declare them in the smallest scope in which they are needed. Include a comment at the beginning of your program with basic information and a description of the program and include a comment at the start of each method.