CS 129 HW8
Selecting from the Cast

Write a program that supports four complex “shapes” or cartoons:

- A snowman
- A fox
- A smiley face
- One other shape of your choice

All four should share a common base size of 50 (the size of the main shape of the smiley/fox, and the base of snowman). All four also start in a random color. Some drawing code for the first three is provided on the next page.

At application start, 25 of these shapes are drawn randomly around the screen, in random positions, and with the choice of shape randomly picked with equal odds.

When the user clicks and drags the mouse, display a light gray selection rectangle originating from the initial click point, like you see in any other drawing application (like MS Paint) when you click and drag to select a space. Shapes inside the rectangle become selected, which turns their outline red instead of black.

The selection ends when the user lets go of the mouse click, and the selection rectangle should no longer be visible. When the user clicks anywhere again, everything becomes deselected, and the selection rectangle starts again. Deselected shapes have a black outline again.

Hint: For ellipses, like the snowman, there’s a built-in method to retrieve the rectangle that surrounds the circle that should come in handy here.

When the “a” key is pressed, all selected shapes, and only the selected shapes, get a new random color.

When the “s” key is pressed, all selected shapes, and only the selected shapes, increase in size by 5 units.

Extra credit (up to 5%) for exceptionally cool custom shapes.

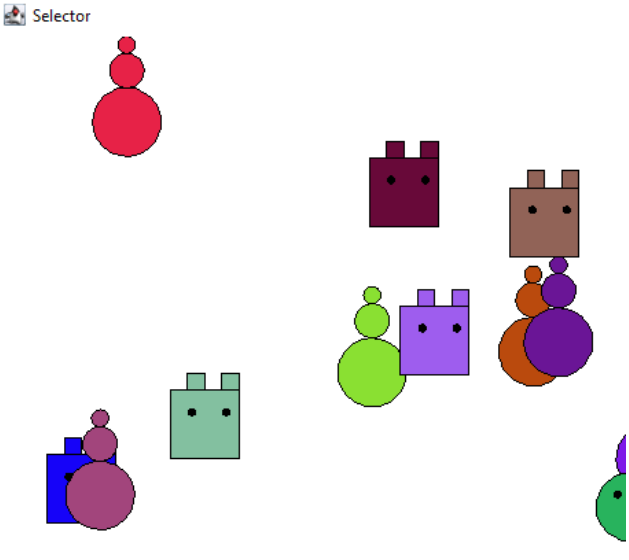
Extra credit (up to 5%) if the “d” key inverts the current selection: selected shapes become deselected and vice-versa.

As always, up to 5% extra credit for documentation.

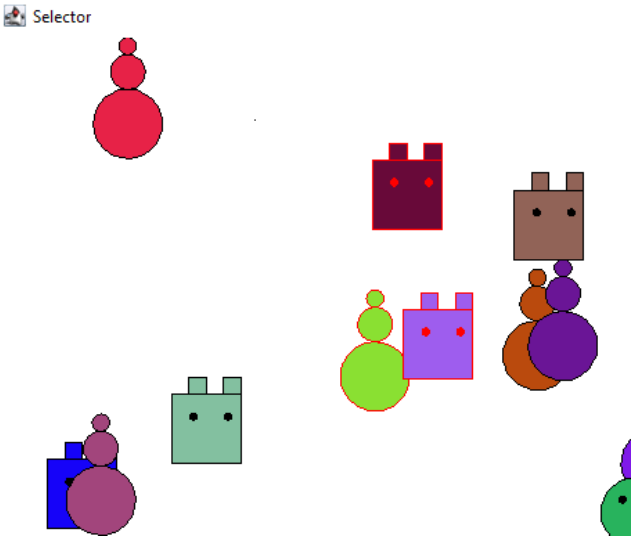
Hint: you could use interfaces for this assignment, but it’s also equally easy to do without them.

```
//Snowman (all ellipses)

                    base = new Ellipse2D.Double(x, y, baseSize, baseSize);
                    center = new Ellipse2D.Double(x + baseSize / 2 / 2, y - baseSize / 2, baseSize / 2, baseSize / 2);
                    top = new Ellipse2D.Double(x + baseSize / 4 / 2 + baseSize / 2 / 2, y - baseSize / 2 - baseSize / 4,
                                        baseSize / 4, baseSize / 4);


//Smiley (all ellipses)

                    base = new Ellipse2D.Double(x, y, baseSize, baseSize);
                    leftEye = new Ellipse2D.Double(x + baseSize * 3 / 4, y + baseSize / 4, baseSize / 8, baseSize / 8);
                    rightEye = new Ellipse2D.Double(x + baseSize / 4, y + baseSize / 4, baseSize / 8, baseSize / 8);


//Fox (base and ears are rectangles, eyes are ellipses)

                    base = new Rectangle2D.Double(x, y, baseSize, baseSize);
                    leftEar = new Rectangle2D.Double(x + baseSize * 3 / 4, y - baseSize / 4, baseSize / 4, baseSize / 4);
                    rightEar = new Rectangle2D.Double(x + baseSize / 4, y - baseSize / 4, baseSize / 4, baseSize / 4);
                    leftEye = new Ellipse2D.Double(x + baseSize * 3 / 4, y + baseSize / 4, baseSize / 8, baseSize / 8);
                    rightEye = new Ellipse2D.Double(x + baseSize / 4, y + baseSize / 4, baseSize / 8, baseSize / 8);



//Some code you might find useful for the selector:

          Point start;
          Rectangle2D.Double display = new Rectangle2D.Double(0, 0, 0, 0);

          public Selector() {

          }

          /**
           * Plants one corner of this selector at the given point. Used to start a drag
           * operation
           *
           * @param p
           */
          void startDrag(Point p) {
                    start = p;
          }

          /**
           * Adjusts the x, y, width and height of the selector so that the opposite
           * corner of the selector (relative to {@link #start} is at the given point
           *
           * @param p
           */
          void dragToPoint(Point p) {
                    Point end = p;
                    if (p.x > start.x && p.y > start.y) {
                              display.x = start.x;
                              display.y = start.y;
                    } else if (p.x < start.x && p.y < start.y) {
                              display.x = end.x;
                              display.y = end.y;
                    } else if (p.x < start.x && p.y > start.y) {
                              display.x = end.x;
                              display.y = start.y;
                    } else {
                              display.x = start.x;
                              display.y = end.y;
                    }
                    display.width = Math.abs(end.x - start.x);
                    display.height = Math.abs(end.y - start.y);
          }

          /**
           * Resets the selector to a size of zero, ending the display
           */
          void endDrag() {
                    display.width = 0;
                    display.height = 0;
          }

          /**
           * Returns true if the selector contains all of the given rectangles
           *
           * @param others
           * @return
           */
          public boolean contains(Rectangle2D[] others) {
                    for (int i = 0; i < others.length; i++) {
                              if (!display.contains(others[i])) {
                                        return false;
                              }
                    }
                    return true;
          }
```

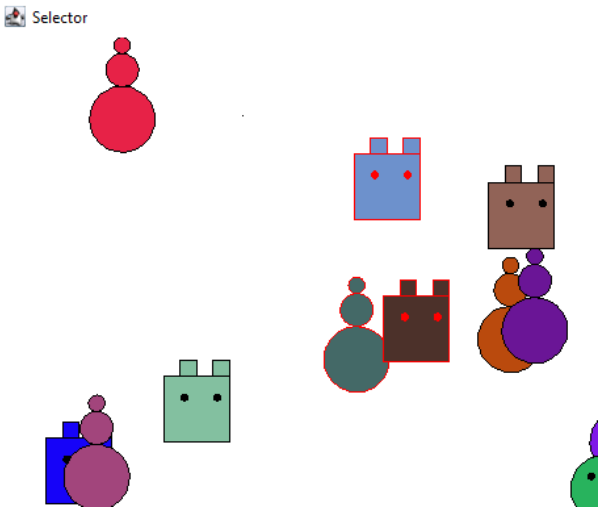
Application Start



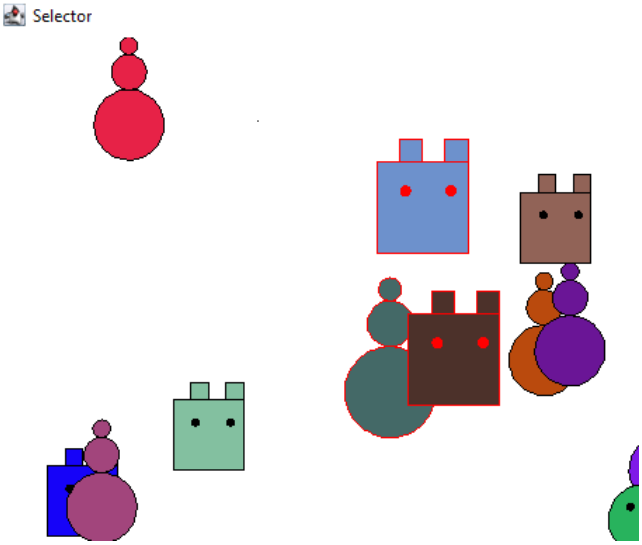
Selected Some Shapes



Pressed “A”



Pressed “S” a Few Times



Selector in Action (Note that left clicks start a new click and drag and deselects everything)