Learning Objectives: (i) Graphical Sprites as Objects. (ii) Getters and Setters. (iii) A list/stack/queue of things (primitives & objects).

Reminder Photoscoop is due tonight 8pm.

```
class Ghost {
// Class variables ("static in memory") are not part of each
Ghost object – just happen to be defined in the same java file
   private static Color DEFAULT_COLOR = Color.ORANGE;

// Instance variables - part of each object:
   private int x, y, direction;
   private Color color = DEFAULT_COLOR;

// Instance methods - call these on an object:
   public void setColor(Color c) { this.color = c;}
   public int getX() { return this.x; }
   public int getY() { return this.y; }

   public void setX(int x) {                    }

   public void setY(         ) {                    }

   public ______ getColor()      {                     }

   public void setXY(int xx, int yy) { x = xx; y = yy;}

   public boolean isTouching(Pacman p) {
// return true if Pacman is within 16 units in X
direction and within 16 units in the Y direction
// Hint: use p.getX(), p.getY(), Math.abs


   }
   public void move() {
      if (direction == 0) this.x += 1;
      else if (direction == 1) this.x -= 1;


   }
   public void paint(Graphics g) {
      g.setColor(color);
      g.fillRect(this.x-8, this.y-8, 16, 16);
       //later when the art is ready…
      //g.drawImage(this.ghostImage, x - 8, x - 8, null);
   }
```

```
public class SimpleList {
  private double[] data ;

  public int length() { _____________________________}
// Add a value to end of the list
  public void add(double value) {









  }
  public double removeAt(int index) {









}
  public double removeFromFront() {
// add to the end and always remove from
// the front: our list can be used as a queue!

}
  public double removeFromEnd() {
// our list can be used as a stack!

}
```

Complete the *'GhostList'* such that the following code compiles and works correctly. Your *GhostList* should have **two** instance variables - an array of ghost references and a count of ghosts in the array. The array may not be completely full

```
Ghost g1 = new Ghost();
Ghost g2 = new Ghost();
g1.setColor(Color.BLUE);
g1.setXY(100, 50);
g2.setColor(Color.GREEN);
g2.setXY(100, 80);

GhostList myGhosts = new GhostList();
myGhosts.add(g1);
myGhosts.add(g2);

int count = myGhosts.getCount()
while (true) {
   myGhosts.moveAll(); // move each ghost
   Graphics2D g = Zen.getBufferGraphics();
   myGhosts.paintAll(g);
   Zen.flipBuffer();
   Ghost blueG = myGhosts.findFirst(Color.BLUE);
   blueGhost.move(); // blue ghost moves twice.
}
```

// **Each GhostList object includes a pointer to its own array and an integer counter – the number of valid entries in the array. The array is an array of pointers to Ghosts.**
So `array[0].setX(8)` will set the x position of the first ghost.

```
public class GhostList {

   private Ghost[] array = new Ghost[100];     // array of pointers to Ghosts

   int count = 0;

  public void add(Ghost g) {
// Simplifying assumption: there will never be more than one hundred ghosts




}

public void paintAll() {




}
// Hint: Color objects implement an "equals" method.
public _______ findFirst(Color c) {




}
```