```
import java.awt.*;
import java.applet.*;

/// = Amad Ansari's annotations

class FlagBuilder extends Object
{
  private static int flagCount = 0;
  int x,y,w,h;
  Applet parapp;


  public FlagDescription getFlagDesc(String country)
  {
    FlagDescription fd = FlagDescription.getFlagDesc(country);
    return fd;
  }


  public FlagBuilder(int x, int y, int w, int h, Applet parapp)     //
size of flag
  {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.parapp = parapp;
  }


  public Element makeFlagElement(ElementDescription ED)   ///takes in
ElementDescription object, outputs Element object
  {
    if (ED != null) {
      String flagpartname = ED.countryName + " " + ED.elementName;
      Element F = null;
      int shapeX = x+(int)((ED.x*.001)*w);
      int shapeY = y+(int)((ED.y*.001)*h);
      int shapeW = (int)((ED.w*.001)*w);
      int shapeH = (int)((ED.h*.001)*h);
      //System.out.println("MakeFlagElement flag=" + flagpartname);
      if (ED.type.startsWith("circle")) {  ///checks
ElementDescription "type" argument, which holds type of shape
        int diameter = shapeW;
        F = new ElementOval( ///runs ElementOval function with given
specifications
          flagpartname,
//          x+(int)((w/2)-(diameter/2)),
//          y+(int)((h/2)-(diameter/2)),
          x+(int)((shapeX)-(diameter/2)),
```

```
          y+(int)((shapeY)-(diameter/2)),
          diameter  );
      }
      else if (ED.type.startsWith("arc")) {
        int diameter = shapeW;
        F = new ElementArc( ///ElementArc function found in
ElementArc.java
          flagpartname,
          shapeX,
          shapeY,
          diameter,diameter,
          ED.var1, ED.var2);    // start and end of arc
      }
      else if (ED.type.startsWith("star")) { ///new shape, does not
have its own java file and all calculations are made in
FlagBuilder.java
        int starWidth = shapeW;
        int starHeight = calcStarHeight(starWidth);
        int xpos = shapeX;
        int ypos = shapeY;
        // Star: var1 is degrees rotation
        Polygon starp =
makeStarPoints(xpos,ypos,starWidth,ED.var1,ED.type);
        Polygon starp1 =
makeStarPoints(xpos,ypos,starWidth,ED.var1,ED.type);
        starp1.translate(1,1);
        F = new ElementPolygon(flagpartname, starp, starp1);
      }
      else if (ED.type.startsWith("triangle")) {
        // Triangle wide on the left, point to right
        Polygon p = new Polygon();
        p.addPoint(shapeX, shapeY);
        p.addPoint(x+shapeW, y+(shapeH/2));
        p.addPoint(shapeX, y+shapeH);
        Polygon p1 = new Polygon();
        p1.addPoint(shapeX, shapeY);
        p1.addPoint(x+shapeW, y+(shapeH/2));
        p1.addPoint(shapeX, y+shapeH);
        p1.translate(1,1);
        F = new ElementPolygon(flagpartname, p, p1);
      }
      else if (ED.type.startsWith("poly")) {
        // X
        //System.out.println("makeflagelement:  polygon");
        int left = shapeX;
        int top = shapeY;
        int width = shapeW;
        int height = shapeH;
        Polygon p = new Polygon();
        for (int i=0; i < ED.numPoints*2; i+=2) {
```

```
          p.addPoint(x+(int)((ED.polyPoints[i]*.001)*w), y+(int)
((ED.polyPoints[i+1]*.001)*h)); ///uses polypoints function in
ElementDescription
        }
        F = new ElementPolygon(flagpartname, p, p);
      }
      else if (ED.type.startsWith("cross")) {
        // Cross
        int left = shapeX;
        int top = shapeY;
        int width = shapeW+2;   // !!! add a pixel
        int height = shapeH+2;
        int barwidth = (int)(((ED.var1*.001)*w)/2);
        int hpos = (int)((ED.var2*.001)*w);
        int vpos = (int)((ED.var3*.001)*h);
        Polygon p = new Polygon();
        p.addPoint(left+hpos-barwidth, top);               //     __
        p.addPoint(left+hpos+barwidth, top);               //
        p.addPoint(left+hpos+barwidth, top+vpos-barwidth); //       |
        p.addPoint(left+width, top+vpos-barwidth);         //
---
        p.addPoint(left+width, top+vpos+barwidth);         //
|
        p.addPoint(left+hpos+barwidth, top+vpos+barwidth); //
---
        p.addPoint(left+hpos+barwidth, top+height);        //       |
        p.addPoint(left+hpos-barwidth, top+height);        //     --
        p.addPoint(left+hpos-barwidth, top+vpos+barwidth); //   |
        p.addPoint(left, top+vpos+barwidth);               // --
        p.addPoint(left, top+vpos-barwidth);               //|
        p.addPoint(left+hpos-barwidth, top+vpos-barwidth); // --
        p.addPoint(left+hpos-barwidth, top);               //   |
        F = new ElementPolygon(flagpartname, p, p);
      }
      else if (ED.type.startsWith("image")) {   //angola_tn.gif    ///
if gif image (coat of arms), locate that image in the website's files
        String imageName = ED.countryName.toLowerCase().replace('
','_') + "_image.gif";
        Image coatOfArms =
parapp.getImage(parapp.getDocumentBase(),"images/" + imageName);
        //System.out.println("Load image base=" +
parapp.getDocumentBase() + " name=" + ("images/" + imageName));
        if (waitForImage(coatOfArms,parapp)) {
          F = new ElementImage(
            flagpartname,
            shapeX,
            shapeY,
            shapeW,
            shapeH,
            coatOfArms  );
```

```
        }
        else {
          System.out.println("ERROR: could not load image:" +
imageName);
          F = null;
        }
      }
      else if (ED.type.startsWith("rect")) {
        F = new Element(
          flagpartname,
          shapeX,
          shapeY,
          shapeW+2,      // !!!! add a pixel to fill gap between
stripes
          shapeH+2  );
      }
      else {
        System.out.println("FlagBuilder.makeFlagElement(): unknown
shape type: " + ED.type);
      }
      // Set some element member vars
      if (F != null) {
        F.setColor(ED.r,ED.g,ED.b);
        F.country = ED.countryName;
        F.element = ED.elementName;
        F.countryId = ED.countryId;
        F.elementId = ED.elementId;
        F.type = ED.type;   // rect,poly,etc
      }
      return F;
    }
    return null;
  }


  public Element makeFlagElement(String country, String element)  ///
overloaded makeFlagElement functiom, this one takes in country and
element strings,
  {                                                                ///
not an ElementDescription object
    FlagDescription flagD = getFlagDesc(country);  ///assigns
flagDescription
    if (flagD != null) {
      ElementDescription ed = flagD.getElement(element); ///assigns
elementDescripton,
      if (ed != null) {
        return makeFlagElement(ed);                    ///then calls
makeFlagElement function with the ElementDescription, which is the
function
      }                                                ///on line 29
```

```
    }
    System.out.println("MakeflagElement(): Cant't find " + country  +
" " + element);
    return null;
  }


  public Flag makeFlag(String countryname, Flag flag)
  {
    FlagDescription flagD = getFlagDesc(countryname);
    if (flagD == null) {
      System.out.println("makeFlag(" + countryname + "): getFlagDesc()
returned null");
    }
    else {
      ElementDescription ED;
      Element F;
      for (int i=0; i<flagD.getNumElements(); i++) {
        ED = flagD.getElement(i);
        if (ED == null) {
          System.out.println("Flag Element " + i + " is NULL!");
        }
        else {
          //System.out.println("MakeFlag() flag="+flagD.countryName +"
eleId=" + i + " x=" + ED.x + " y=" + ED.y + " w=" + ED.w + " h=" +
ED.h);
          F = makeFlagElement(ED);
          flag.add(F);
        }
      }
    }
    return flag;
  }


  public int calcStarHeight(int starWidth)  {
      return (int) (.959 * starWidth);
  }

  public Polygon makeStarPoints(int xpos, int ypos, int starWidth, int
degreesRotation, String style)
  {
    if (style.equals("star5")) {
      return makeStar(xpos, ypos, starWidth, .380F, 5,
degreesRotation);  ///checks what kind of star it is from
ElementDescription, makes calculations
    }
///based on it
    else if (style.equals("starz")) {
      return makeStar(xpos, ypos, starWidth, .5F, 5, 0);
```

```
    }
    else if (style.equals("star7")) {
      return makeStar(xpos, ypos, starWidth, .45F, 7, 0);
    }
    else if (style.equals("star14")) {
      return makeStar(xpos, ypos, starWidth, .45F, 14, 0);
    }
    else {
      return makeStar(xpos, ypos, starWidth, .5F, 5, 0);
    }

  }

/*
  public Polygon makeStarMask(int xpos,int ypos, int x, int y, int w,
int h, int starWidth, String style)
  {
      Polygon star = makeStarPoints(xpos,ypos,starWidth,style);
        // back to top of star
         star.addPoint((int) (xpos + (.5 * starWidth)), ypos);
         // to top of ground
         star.addPoint( (int) (xpos + (.5 * starWidth)), y);
         // to upper left corner and around ground
         star.addPoint( x, y);
         star.addPoint( x, y+h);
         star.addPoint( x+w, y+h);
         star.addPoint( x+w, y);
         // back to top of ground above star
         star.addPoint( (int) (xpos + (.5 * starWidth)), y);
      return star;
  }
*/

        public Polygon makeStar(int x1, int y1,
  float d1, float spikyness1, int points1, int rotation1)
  {
                // params : x, y, diameter, spikyness, points,
rotation
    // (0= point at top) in degrees

                int x = (int)(x1 + (d1/2));
                int y = (int)(y1 + (d1/2));
                int points = points1;

                float rotation = (float)((rotation1/360F) * (2 *
Math.PI));

                // spikyness is the amount of point.  it goes
    // from 0 to 1. At 0,
                // the interior point is at the origin
```

```
                // at 1, it is on the star's bounding circle.

                float spikyness = spikyness1;
                float diameter = d1;
                float radius = (diameter / 2);

                Polygon dummy = new Polygon();

                int p1;
                int p2;

                // this is the amount we rotate.  It's double the
amount
    // of points on the star,
                // because we rotate halfway to put inthe interior
points
                // the code works by defining an exterior point
                // then an interior point
                float angleDifference = (float)((2F*Math.PI)/
(points*2F));

                // 'cause we want the top of the star to be a point:
                float angle = (float)((6*Math.PI/4F) + rotation);

                for  (int i=0; i < (points*2); i++){
                        if (i%2 == 0 || i == 0){
                                p1 = (int)(x +
radius*Math.cos(angle));
                                p2 = (int)(y +
radius*Math.sin(angle));
                        } else {
                                p1 = (int)(x +
spikyness*radius*Math.cos(angle));
                                p2 = (int)(y +
spikyness*radius*Math.sin(angle));
                        }
                        dummy.addPoint(p1, p2);
                        angle = angle + angleDifference;
                }

                return dummy;
  }


/*
  public static boolean waitForImage(Image img, Component c)
  {

        int imgW=-1, imgH=-1, counter=200;
```

```
        while ( (imgW=img.getWidth(c)) < 0 && counter > 0)
            try {
            counter--; Thread.sleep(100);
          }
          catch( InterruptedException e ) {
            System.out.println(e);
          }
        while ( (imgH=img.getHeight(c)) < 0 && counter > 0 )
            try {
            counter--; Thread.sleep(100);
          }
          catch( InterruptedException e ) {
            System.out.println(e);
          }
        return (counter > 0);
        //System.out.println("TOOLBAR IMGW=" + imgW + " IMGH=" +
imgH );
  }

*/


    public static boolean waitForImage(Image image, Component
component)
    {
        MediaTracker tracker = new MediaTracker(component);
        try {
            tracker.addImage(image, 0);
            tracker.waitForID(0);
        }
        catch(InterruptedException e) {
            e.printStackTrace();
            return false;
        }
        return true;
    }

}
```