Angelo Daniel A. Dela Paz

4CSC

Machine Problem 2

Cross using Lines and Quadratic Bezel

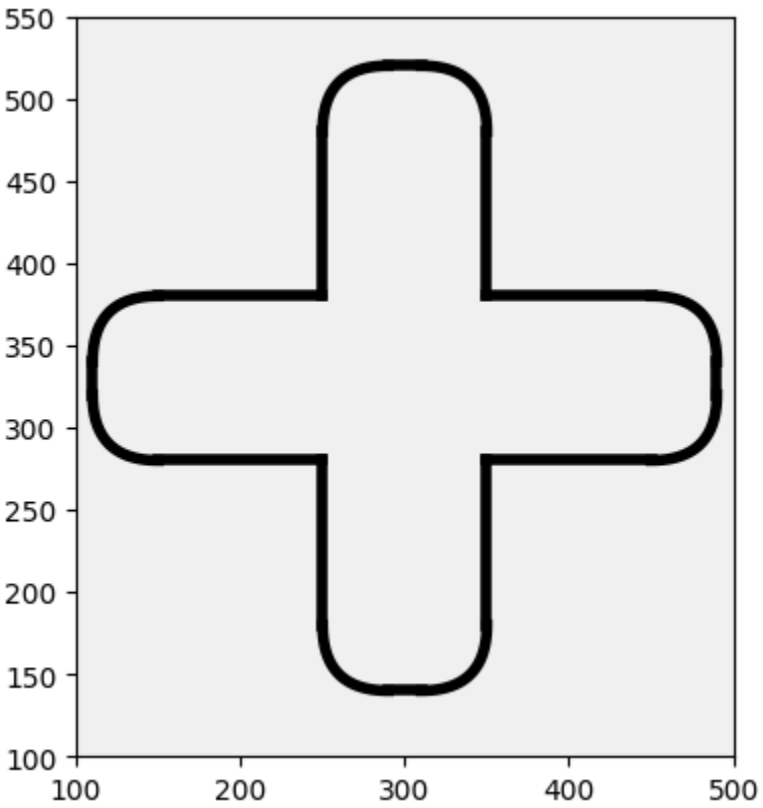
Python

```python
# Set up the plot
fig, ax = plt.subplots()
ax.set_aspect("equal")
ax.set_xlim(100, 500)
ax.set_ylim(100, 550)
ax.set_facecolor((240 / 255, 240 / 255, 240 / 255))

# Quadratic Bezier curve function
def quadratic_bezier(t, p0, p1, p2):
    x = (1 - t) ** 2 * p0[0] + 2 * (1 - t) * t * p1[0] + t**2 * p2[0]
    y = (1 - t) ** 2 * p0[1] + 2 * (1 - t) * t * p1[1] + t**2 * p2[1]
    return x, y

# Draw the cross using lines and quadratic Bezier curves

# Top part
ax.plot([250, 250], [280, 180], "k-", linewidth=4)
ax.plot([350, 350], [180, 280], "k-", linewidth=4)
ax.plot([290, 310], [140, 140], "k-", linewidth=4)
points = np.array(
    [
        quadratic_bezier(t, (250, 180), (250, 140), (290, 140))
        for t in np.linspace(0, 1, 101)
    ]
)
ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)
points = np.array(
    [
        quadratic_bezier(t, (310, 140), (350, 140), (350, 180))
        for t in np.linspace(0, 1, 101)
    ]
)
ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)
# Right part
```

```
ax.plot([350, 450], [280, 280], "k-", linewidth=4)
ax.plot([490, 490], [320, 340], "k-", linewidth=4)
ax.plot([350, 450], [380, 380], "k-", linewidth=4)
points = np.array(
    [
        quadratic_bezier(t, (450, 280), (490, 280), (490, 320))
        for t in np.linspace(0, 1, 101)
    ]
)
ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)
points = np.array(
    [
        quadratic_bezier(t, (490, 340), (490, 380), (450, 380))
        for t in np.linspace(0, 1, 101)
    ]
)
ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)

# Bottom part
ax.plot([250, 250], [380, 480], "k-", linewidth=4)
ax.plot([290, 310], [520, 520], "k-", linewidth=4)
ax.plot([350, 350], [380, 480], "k-", linewidth=4)
points = np.array(
    [
        quadratic_bezier(t, (250, 480), (250, 520), (290, 520))
        for t in np.linspace(0, 1, 101)
    ]
)
ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)
points = np.array(
    [
        quadratic_bezier(t, (310, 520), (350, 520), (350, 480))
        for t in np.linspace(0, 1, 101)
    ]
)
ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)

# Left part
ax.plot([250, 150], [280, 280], "k-", linewidth=4)
ax.plot([110, 110], [320, 340], "k-", linewidth=4)
ax.plot([250, 150], [380, 380], "k-", linewidth=4)
points = np.array(
    [
        quadratic_bezier(t, (150, 280), (110, 280), (110, 320))
        for t in np.linspace(0, 1, 101)
```

```
    ]
)
ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)
points = np.array(
    [
        quadratic_bezier(t, (110, 340), (110, 380), (150, 380))
        for t in np.linspace(0, 1, 101)
    ]
)
ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)

plt.show()
```

OUTPUT



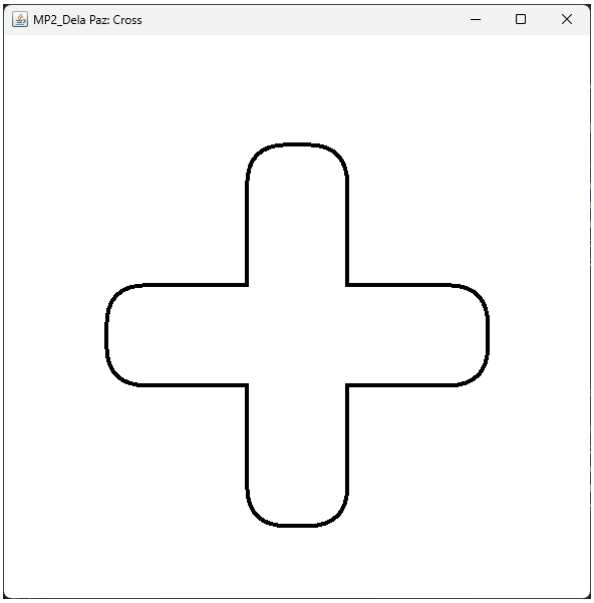
Java

```
import java.awt.*;
import java.awt.geom.QuadCurve2D;

public class MP2_DelaPaz extends Frame {

```

```
97.     private void QuadraticBezier(Graphics2D g2d, int x0, int y0, int x1,
    int y1, int x2, int y2) {
98.         QuadCurve2D q = new QuadCurve2D.Float();
99.         q.setCurve(x0, y0, x1, y1, x2, y2);
100.             g2d.draw(q);
101.         }
102.
103.         public void paint(Graphics g) {
104.             Graphics2D g2d = (Graphics2D) g;
105.
106.             // Set the stroke for drawing
107.             g2d.setStroke(new BasicStroke(4));
108.             g2d.setColor(Color.BLACK);
109.
110.             // Draw the cross using lines and quadratic Bezier curves
111.             // Upper Curve
112.             g2d.drawLine(250, 280, 250, 180);
113.             g2d.drawLine(350, 180, 350, 280);
114.             g2d.drawLine(290, 140, 310, 140);
115.             QuadraticBezier(g2d, 250, 180, 250, 140, 290, 140);
116.             QuadraticBezier(g2d, 310, 140, 350, 140, 350, 180);
117.
118.             // Right Curve
119.             g2d.drawLine(350, 280, 450, 280);
120.             g2d.drawLine(490, 320, 490, 340);
121.             g2d.drawLine(350, 380, 450, 380);
122.             QuadraticBezier(g2d, 450, 280, 490, 280, 490, 320);
123.             QuadraticBezier(g2d, 490, 340, 490, 380, 450, 380);
124.
125.             // Bottom Curve
126.             g2d.drawLine(250, 380, 250, 480);
127.             g2d.drawLine(290, 520, 310, 520);
128.             g2d.drawLine(350, 380, 350, 480);
129.             QuadraticBezier(g2d, 250, 480, 250, 520, 290, 520);
130.             QuadraticBezier(g2d, 310, 520, 350, 520, 350, 480);
131.
132.             // Left Curve
133.             g2d.drawLine(250, 280, 150, 280);
134.             g2d.drawLine(110, 320, 110, 340);
135.             g2d.drawLine(250, 380, 150, 380);
136.             QuadraticBezier(g2d, 150, 280, 110, 280, 110, 320);
137.             QuadraticBezier(g2d, 110, 340, 110, 380, 150, 380);
138.
139.         }
140.
```

```
141.         // Main Method
142.
143.         public static void main(String[] args) {
144.             Frame frame = new MP2_DelaPaz();
145.             frame.setTitle("MP2_Dela Paz: Cross");
146.             frame.setSize(600, 600);
147.             frame.setBackground(Color.WHITE);
148.             frame.setForeground(Color.BLACK);
149.             frame.setVisible(true);
150.         }
151.     }
152.
```

Output



Cross using General Path

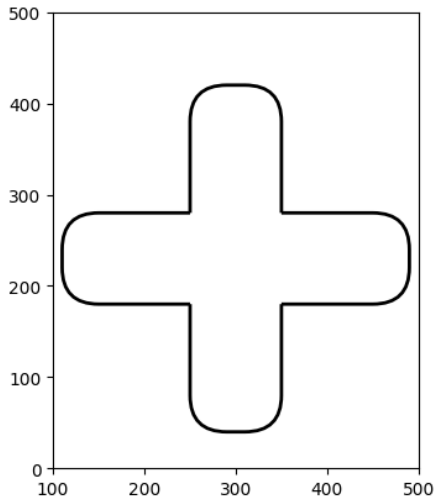
Python

```
153.     verticesA = [
154.         (250, 280),  # Starting point
155.         (250, 380),  # Straight line
156.         (250, 420),  # Curve 1
157.         (290, 420),  # Curve 1
158.         (310, 420),  # Straight line
159.         (350, 420),  # Curve 2
160.         (350, 380),  # Curve 2
161.         (350, 280),  # Ending point
```

```
    ]

    verticesB = [
        (350, 280),  # Starting point
        (450, 280),  # Straight line
        (490, 280),  # Curve 1
        (490, 240),  # Curve 1
        (490, 220),  # Straight line
        (490, 180),  # Curve 2
        (450, 180),  # Curve 2
        (350, 180),  # Ending point
    ]

    verticesC = [
        (350, 180),  # Starting point
        (350, 80),
        (350, 40),
        (310, 40),
        (290, 40),
        (250, 40),
        (250, 80),
        (250, 180),  # Ending point
    ]

    verticesD = [
        (250, 180),  # Starting point
        (150, 180),
        (110, 180),
        (110, 220),
        (110, 240),
        (110, 280),
        (150, 280),
        (250, 280),  # Ending point
    ]

    codes = [
        path.Path.MOVETO,
        path.Path.LINETO,
        path.Path.CURVE3,
        path.Path.CURVE3,
        path.Path.LINETO,
        path.Path.CURVE3,
        path.Path.CURVE3,
        path.Path.LINETO,
    ]
```

```
207.
208.      pathA = path.Path(verticesA, codes)
209.      pathB = path.Path(verticesB, codes)
210.      pathC = path.Path(verticesC, codes)
211.      pathD = path.Path(verticesD, codes)
212.
213.      patchA = patches.PathPatch(pathA, facecolor="none", lw=2)
214.      patchB = patches.PathPatch(pathB, facecolor="none", lw=2)
215.      patchC = patches.PathPatch(pathC, facecolor="none", lw=2)
216.      patchD = patches.PathPatch(pathD, facecolor="none", lw=2)
217.
218.      fig, ax = plt.subplots()
219.
220.      ax.add_patch(patchA)
221.      ax.add_patch(patchB)
222.      ax.add_patch(patchC)
223.      ax.add_patch(patchD)
224.
225.      ax.set_xlim(100, 500)
226.      ax.set_ylim(0, 500)
227.      plt.gca().set_aspect("equal", adjustable="box")
228.      plt.show()
```

OUTPUT



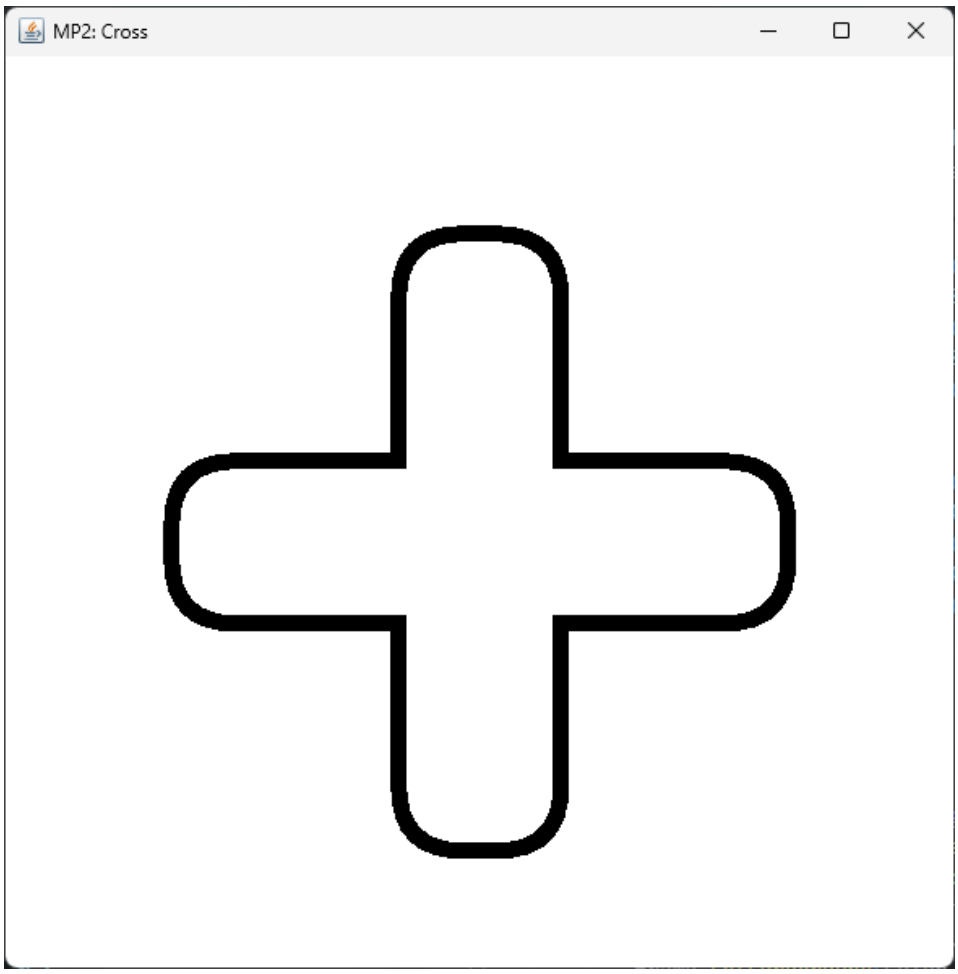
Java

```
229.
230.      import java.awt.*;
231.      import java.awt.geom.GeneralPath;
```

```
232.
233.     public class MP2_Cross extends Frame {
234.
235.         public void paint(Graphics g) {
236.
237.             Graphics2D g2d = (Graphics2D) g;
238.             BasicStroke bs = new BasicStroke(10.0f);
239.             g2d.setStroke(bs);
240.
241.             GeneralPath cross = new GeneralPath();
242.
243.             // Start
244.             cross.moveTo(250, 280); // A
245.
246.             // Left Arm
247.             cross.lineTo(150, 280);
248.             cross.quadTo(110, 280, 110, 320);
249.             cross.lineTo(110, 340);
250.             cross.quadTo(110, 380, 150, 380);
251.             cross.lineTo(250, 380);
252.
253.             // Bottom Arm
254.             cross.lineTo(250, 480);
255.             cross.quadTo(250, 520, 290, 520);
256.             cross.lineTo(310, 520);
257.             cross.quadTo(350, 520, 350, 480);
258.             cross.lineTo(350, 380);
259.
260.             // Right Arm
261.             cross.lineTo(450, 380);
262.             cross.quadTo(490, 380, 490, 340);
263.             cross.lineTo(490, 320);
264.             cross.quadTo(490, 280, 450, 280);
265.             cross.lineTo(350, 280);
266.
267.             // Top Arm
268.             cross.lineTo(350, 180);
269.             cross.quadTo(350, 140, 310, 140);
270.             cross.lineTo(290, 140);
271.             cross.quadTo(250, 140, 250, 180);
272.             cross.lineTo(250, 280);
273.
274.             g2d.draw(cross);
275.
276.         }
```

```
277.
278.         public static void main(String[] args) {
279.             Frame frame = new MP2_Cross();
280.             frame.setTitle("MP2: Cross");
281.             frame.setSize(600, 600);
282.             frame.setBackground(Color.WHITE);
283.             frame.setForeground(Color.BLACK);
284.             frame.setVisible(true);
285.         }
286.     }
287.
```

OUTPUT



Christmas Tree

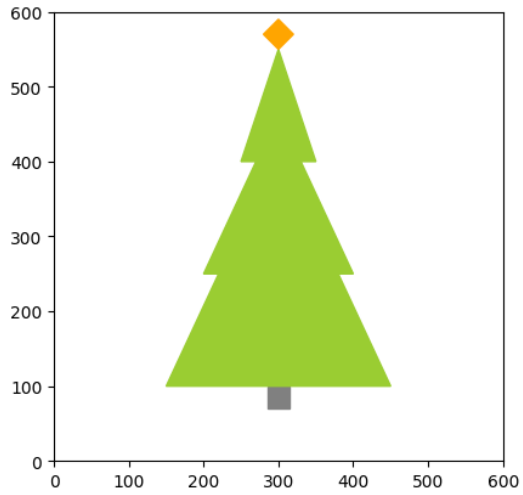
Python

```
3. verticesBase = [(285, 100), (315, 100), (315, 70), (285, 70), (285,
   100)]
```

```
4.
5. verticesTree = [
6.     (300, 100),
7.     (150, 100),
8.     (220, 250),
9.     (200, 250),
10.    (270, 400),
11.    (250, 400),
12.    (300, 550),
13.    (350, 400),
14.    (330, 400),
15.    (400, 250),
16.    (380, 250),
17.    (450, 100),
18.    (300, 100),
19.]
20.
21.verticesStar = [(300, 550), (280, 570), (300, 590), (320, 570), (300,
   550)]
22.
23.codesTrees = [
24.    path.Path.MOVETO,
25.    path.Path.LINETO,
26.    path.Path.LINETO,
27.    path.Path.LINETO,
28.    path.Path.LINETO,
29.    path.Path.LINETO,
30.    path.Path.LINETO,
31.    path.Path.LINETO,
32.    path.Path.LINETO,
33.    path.Path.LINETO,
34.    path.Path.LINETO,
35.    path.Path.LINETO,
36.    path.Path.CLOSEPOLY,
37.]
38.
39.codesBase = [
40.    path.Path.MOVETO,
41.    path.Path.LINETO,
42.    path.Path.LINETO,
43.    path.Path.LINETO,
44.    path.Path.CLOSEPOLY,
45.]
46.
47.codesTree = [
```

```
    path.Path.MOVETO,
    path.Path.LINETO,
    path.Path.LINETO,
    path.Path.LINETO,
    path.Path.LINETO,
    path.Path.LINETO,
]

codesStar = [
    path.Path.MOVETO,
    path.Path.LINETO,
    path.Path.LINETO,
    path.Path.LINETO,
    path.Path.CLOSEPOLY,
]

pathBase = path.Path(verticesBase, codesBase)
pathTree1 = path.Path(verticesTree, codesTrees)
pathStar = path.Path(verticesStar, codesStar)

patchBase = patches.PathPatch(pathBase, edgecolor="Gray",
   facecolor="Gray", lw=1)

patchTree1 = patches.PathPatch(
    pathTree1, edgecolor="YellowGreen", facecolor="YellowGreen", lw=1
)

patchStar = patches.PathPatch(pathStar, edgecolor="Orange",
   facecolor="Orange", lw=1)

fig, ax = plt.subplots()

ax.add_patch(patchBase)
ax.add_patch(patchTree1)
ax.add_patch(patchStar)

ax.set_xlim(0, 600)
ax.set_ylim(0, 600)
plt.gca().set_aspect("equal", adjustable="box")
plt.show()
```

OUTPUT

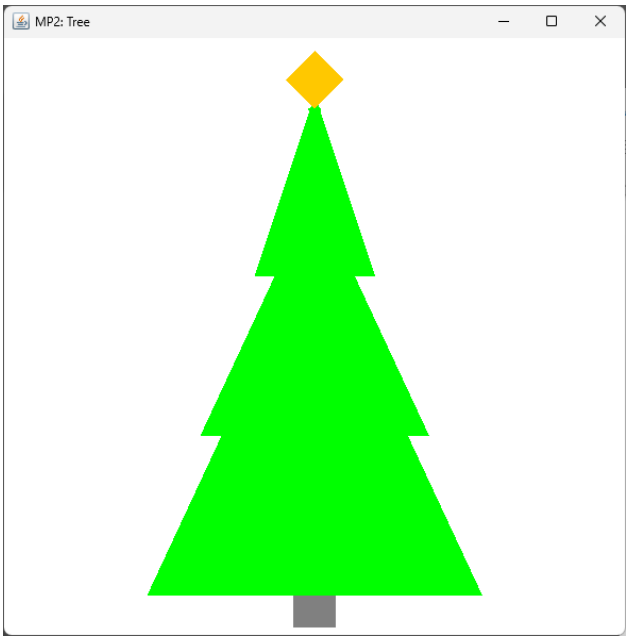


a. Java

```
86.import java.awt.*;
87.import java.awt.geom.Area;
88.import java.awt.geom.GeneralPath;
89.
90.public class MP2_Tree extends Frame {
91.    public void paint (Graphics g){
92.        Graphics2D g2d = (Graphics2D) g;
93.        BasicStroke bs = new BasicStroke(10.0f);
94.        g2d.setStroke(bs);
95.
96.        GeneralPath base = new GeneralPath();
97.        base.moveTo(300, 550);
98.        base.lineTo(315, 550);
99.        base.lineTo(315, 580);
100.           base.lineTo(285, 580);
101.           base.lineTo(285, 550);
102.           base.lineTo(300, 550);
103.           g2d.setPaint(Color.GRAY);
104.           g2d.fill(base);
105.           g2d.draw(base);
106.
107.           GeneralPath tree = new GeneralPath();
108.           tree.moveTo(300, 100);
109.           tree.lineTo(250, 250);
110.           tree.lineTo(270, 250);
111.           tree.lineTo(200, 400);
112.           tree.lineTo(220, 400);
113.           tree.lineTo(150, 550);
114.           tree.lineTo(300, 550);
115.           tree.moveTo(300, 100);
```

```
116.            tree.lineTo(350, 250);
117.            tree.lineTo(330, 250);
118.            tree.lineTo(400, 400);
119.            tree.lineTo(380, 400);
120.            tree.lineTo(450, 550);
121.            tree.lineTo(300, 550);
122.            g2d.setPaint(Color.GREEN);
123.            g2d.fill(tree);
124.            g2d.draw(tree);
125.
126.
127.            GeneralPath star = new GeneralPath();
128.            star.moveTo(300, 90);
129.            star.lineTo(280, 70);
130.            star.lineTo(300, 50);
131.            star.lineTo(320, 70);
132.            star.lineTo(300, 90);
133.            g2d.setPaint(Color.ORANGE);
134.            g2d.fill(star);
135.            g2d.draw(star);
136.        }
137.
138.        public static void main (String[] args){
139.            Frame frame = new MP2_Tree();
140.            frame.setTitle("MP2: Tree");
141.            frame.setSize(600, 600);
142.            frame.setBackground(Color.WHITE);
143.            frame.setForeground(Color.BLACK);
144.            frame.setVisible(true);
145.        }
146.    }
```

OUTPUT

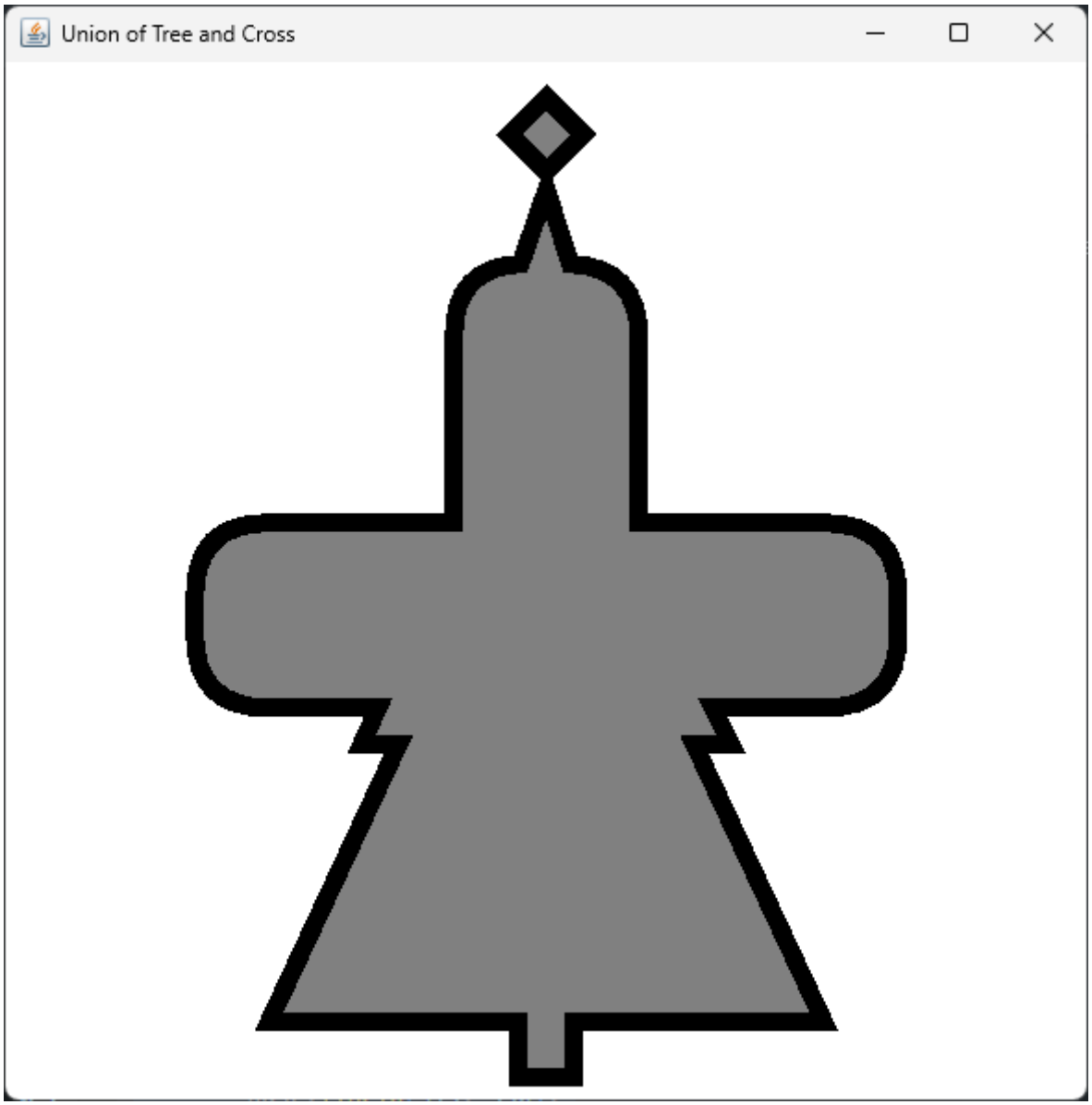


Conversion using Areas in Java

Union (Using the code from above)

```
  // Convert cross to Area
      Area crossArea = new Area(cross);

      // Perform the union operation
      Area unionArea = new Area(treeArea);
      unionArea.add(crossArea);

      // Draw the union of the tree and cross
      g2d.setPaint(Color.GRAY);
      g2d.fill(unionArea);
      g2d.setPaint(Color.BLACK);
      g2d.draw(unionArea);
  }
```

OUTPUT



Intersection (Using the code from above)

```
// Perform the intersection operation
    Area intersectionArea = new Area(treeArea);
    intersectionArea.intersect(crossArea);
```

// Draw the intersection of the tree and cross

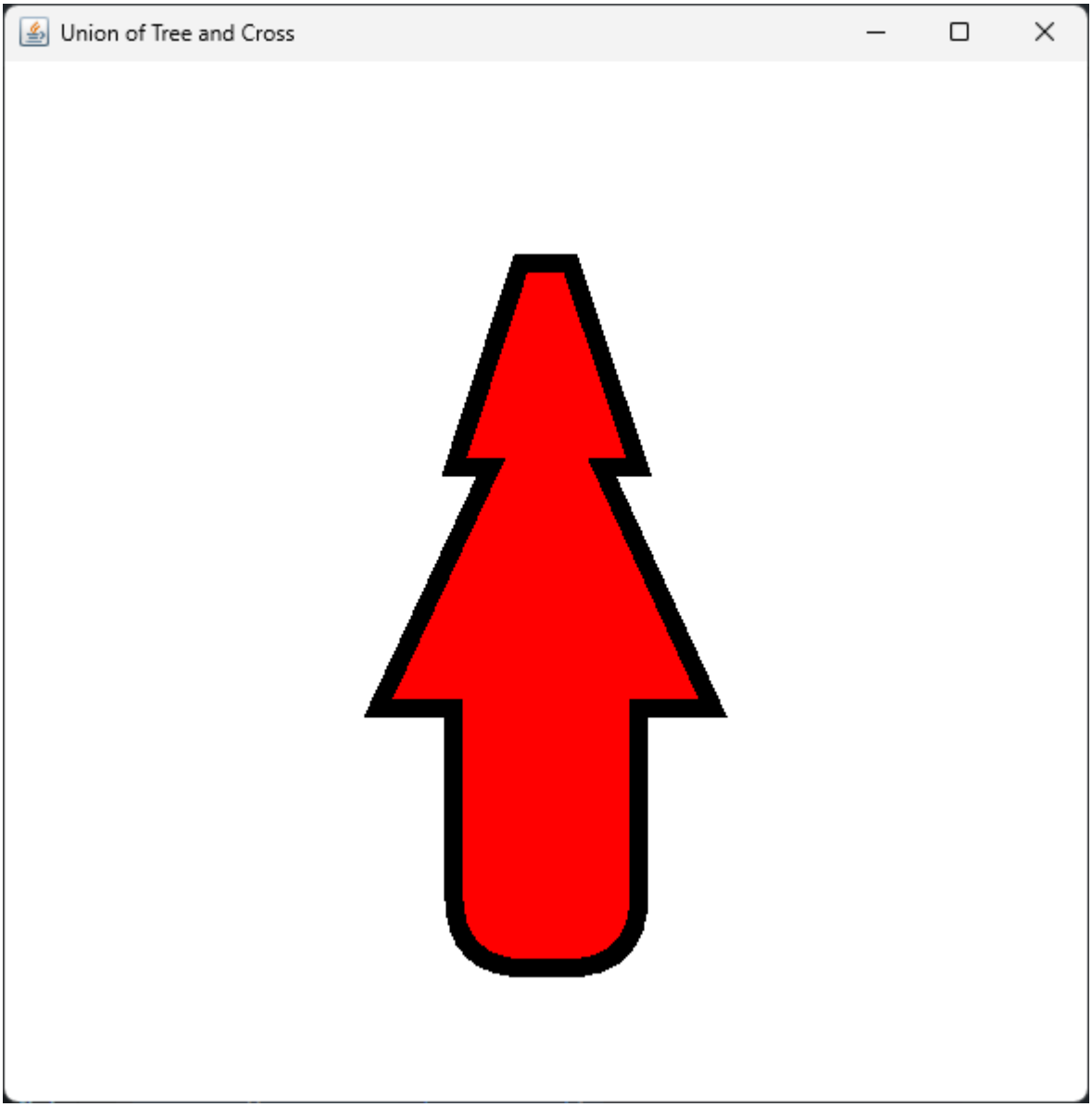
g2d.setPaint(Color.RED);

g2d.fill(intersectionArea);

g2d.setPaint(Color.BLACK);

g2d.draw(intersectionArea);

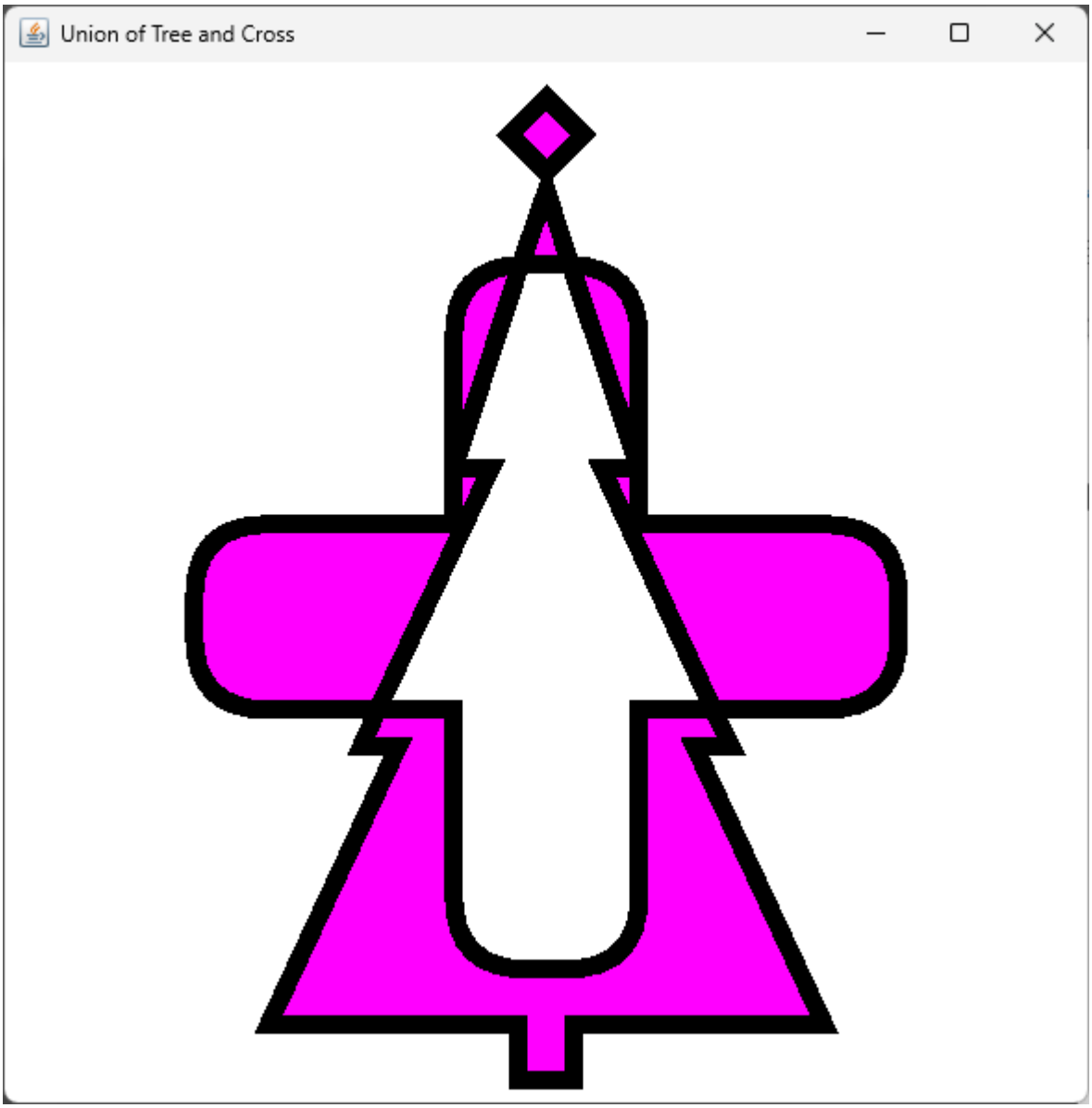
OUTPUT



Symmetric Difference (Using the code from above)

```
// Perform the symmetric difference operation
    Area symmetricDifferenceArea = new Area(treeArea);
    symmetricDifferenceArea.add(crossArea);
    Area intersectionArea = new Area(treeArea);
    intersectionArea.intersect(crossArea);
    symmetricDifferenceArea.subtract(intersectionArea);

    // Draw the symmetric difference of the tree and cross
    g2d.setPaint(Color.MAGENTA);
    g2d.fill(symmetricDifferenceArea);
    g2d.setPaint(Color.BLACK);
    g2d.draw(symmetricDifferenceArea);
```

OUTPUT



Relative Difference (Using the code from above)

```
// Convert cross to Area
    Area crossArea = new Area(cross);


    // Perform the relative difference operation (Tree - Cross)
    Area relativeDifferenceArea = new Area(treeArea);
    relativeDifferenceArea.subtract(crossArea);
```

```
// Draw the relative difference of the tree and cross
g2d.setPaint(Color.CYAN);
g2d.fill(relativeDifferenceArea);
g2d.setPaint(Color.BLACK);
g2d.draw(relativeDifferenceArea);
```

OUTPUT