

## Case Study 2: A Drawing Program

Write a program that can be used to create freehand drawing using the mouse. The user must be able to change the width of the pen used in the drawings. Allow the user to change the pen size using a menu. Drawing will occur while the mouse button is held down.

28

---

---

---

---

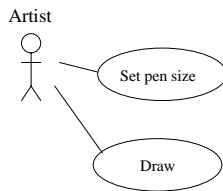
---

---

---

---

## Use Case Diagram



29

---

---

---

---

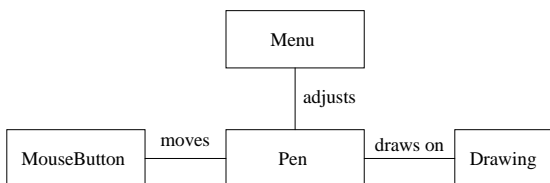
---

---

---

---

## A Class Diagram



30

---

---

---

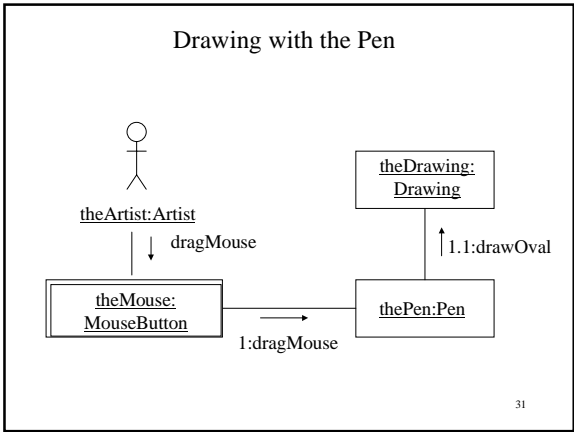
---

---

---

---

---




---

---

---

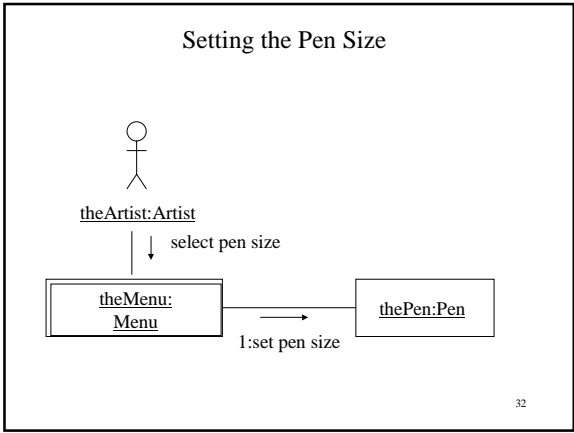
---

---

---

---

---




---

---

---

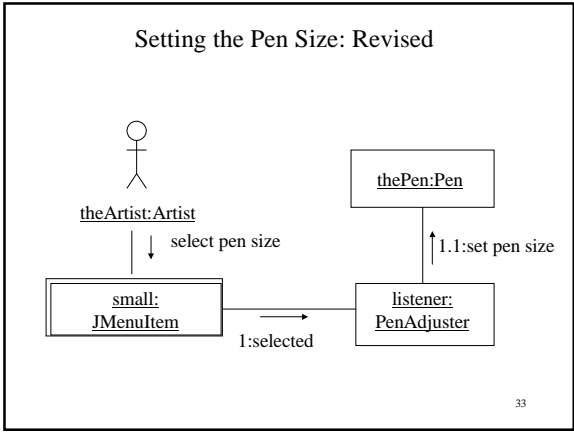
---

---

---

---

---




---

---

---

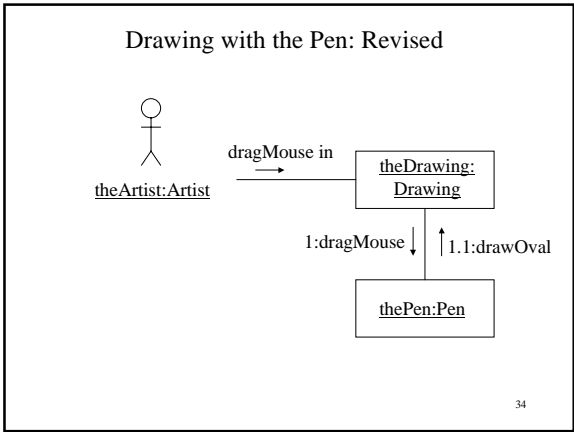
---

---

---

---

---




---

---

---

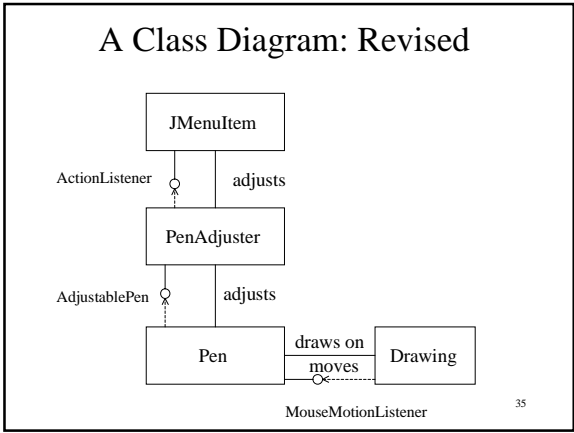
---

---

---

---

---




---

---

---

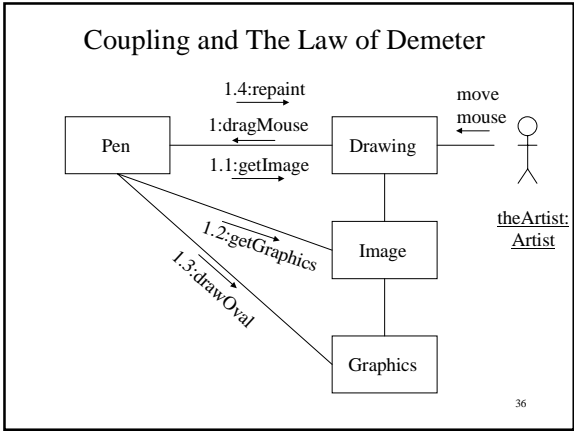
---

---

---

---

---




---

---

---

---

---

---

---

---

```
// Pen.java - paints on a Drawing
import java.awt.*;
import java.awt.event.*;

public class Pen implements MouseMotionListener,
    AdjustablePen
{
    public void mouseDragged(MouseEvent e) {
        Drawing canvas = (Drawing)e.getSource();
        canvas.fillOval(e.getX() - radius,
            e.getY() - radius,
            diameter, diameter);
    }
    public void mouseMoved(MouseEvent e){}
```

37

---

---

---

---

---

---

---

---

```
public void setPenSize(String size) {
    if (size.equals("Small")) {
        radius = 0;
        diameter = 1;
    }
    else if (size.equals("Medium")) {
        radius = 3;
        diameter = radius * 2;
    }
    else if (size.equals("Large")) {
        radius = 6;
        diameter = radius * 2;
    }
}

protected int radius = 3;
protected int diameter = radius * 2;
}
```

38

---

---

---

---

---

---

---

---

```
// PenAdjuster.java - pass pen adjustment requests
// to the AdjustablePen
import java.awt.event.*;

class PenAdjuster implements ActionListener {
    private AdjustablePen pen;
    PenAdjuster(AdjustablePen thePen) {
        pen = thePen;
    }
    public void actionPerformed(ActionEvent e) {
        pen.setPenSize(e.getActionCommand());
    }
}

// AdjustablePen.java -
interface AdjustablePen {
    public void setPenSize(String size);
}
```

39

---

---

---

---

---

---

---

---

```
//Drawing.java - remember drawing operations,  
// using an offscreen image, only supports fillOval()  
import java.awt.*;  
import javax.swing.*;  
  
class Drawing extends JComponent {  
    private static final int SIZE = 500;  
    private Image offscreenImage;  
    private Graphics offscreenGraphics;  
  
    // transfer the offscreen image to the screen  
    public void paint(Graphics g) {  
        if (offscreenImage != null)  
            g.drawImage(offscreenImage,0,0,SIZE,SIZE,null);  
    }  
}
```

40

---

---

---

---

---

---

---

---

```
// draw on an offscreen image, displayed by paint()  
public void fillOval(int left, int top,  
                    int width, int height)  
{  
    if (offscreenImage == null) {  
        offscreenImage = createImage(SIZE, SIZE);  
        offscreenGraphics = offscreenImage.getGraphics();  
    }  
    offscreenGraphics.fillOval(left,top,width,height);  
    repaint();  
}  
public Dimension getMinimumSize() {  
    return new Dimension(SIZE, SIZE);  
}  
public Dimension getPreferredSize() {  
    return new Dimension(SIZE, SIZE);  
}  
}
```

41

---

---

---

---

---

---

---

---