Computer Programming Fundamentals

CS 152
Professor: Leah Buechley
TAs: Melody Horn, Noah Garcia, Andrew Geyko, Juan Ormaza
Time: MWF 10:00-10:50am
https://handandmachine.cs.unm.edu/classes/CS152_Fall2021/

MIDTERM GRADED

HIGHEST GRADE: 100
LOWEST GRADE: 0
AVERAGE: 86
(doesn’t include 0s)

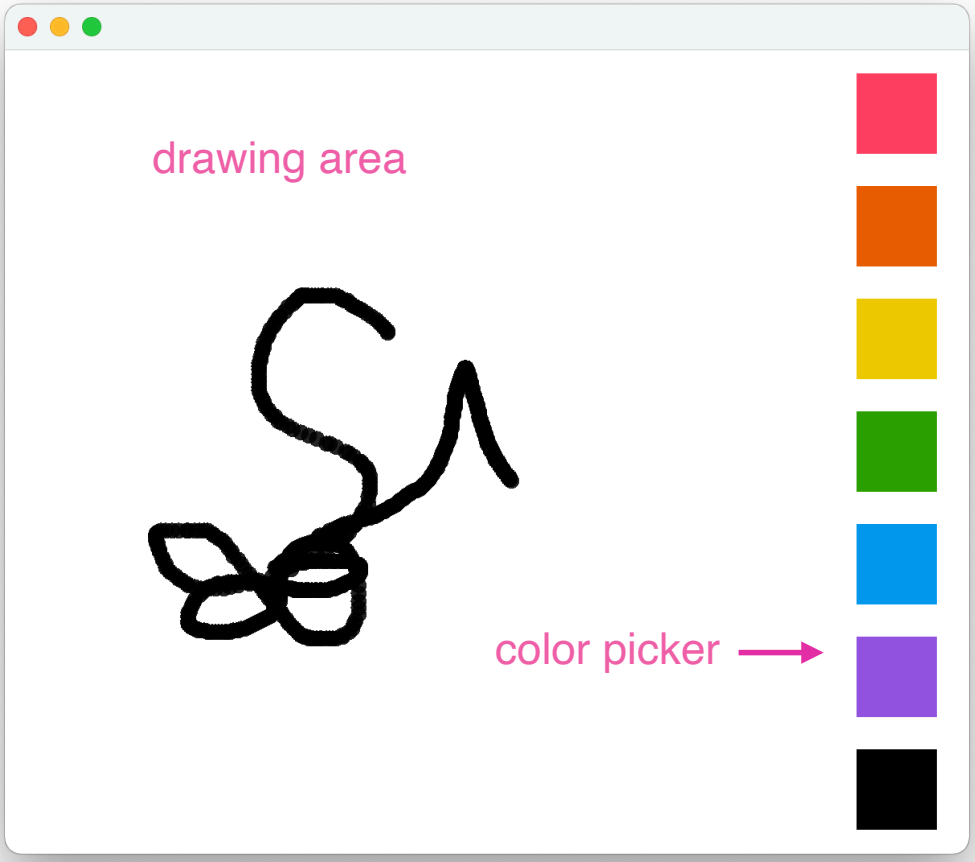
CLASS GRADE SO FAR BY MONDAY

questions?

OPEN UP PROJECT FROM LAST CLASS

LET’S MAKE A DRAWING PROGRAM

A SIMPLE DRAWING PROGRAM



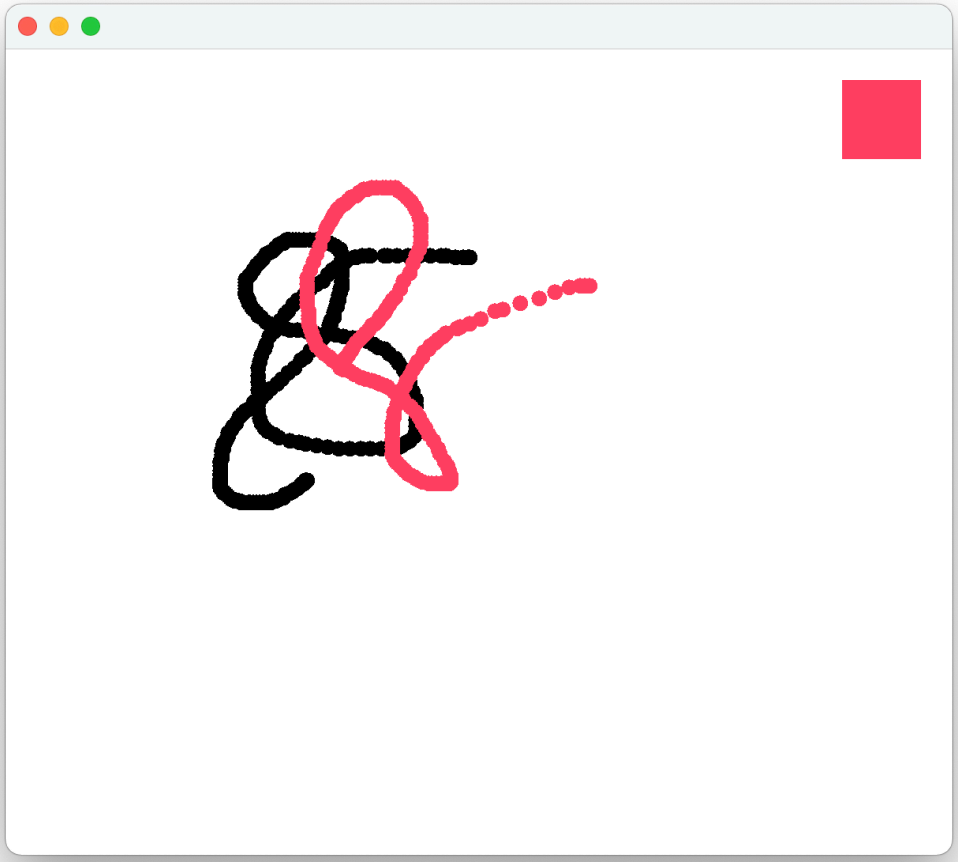
- click on a color square to change drawing color
- press spacebar to clear drawing (but not color picker)
- draw only when mouse is dragged
- don’t draw on color picker

THINGS WE NEED TO DO

- Draw color picker squares along edge
- Implement color picking
  - create variable to store current color information
  - determine which square is being clicked on
  - change color when a different square is clicked
- Implement drawing when dragged
  - draw when mouse is dragged
  - don’t draw on color picker
- Implement screen clearing
  - clear screen when space bar is pressed
  - don’t clear color picker

WHERE WE ARE

1 COLOR PICKER SQUARE



THINGS WE NEED TO DO

- ~~Draw 1 color picker square along edge~~
- ~~Implement color picking~~
  - ~~create variable to store current color information~~
  - ~~determine which square is being clicked on~~
  - ~~change color when a different square is clicked~~
- ~~Implement drawing when dragged~~
  - ~~draw when mouse is dragged~~
  - ~~don't draw on color picker~~
- Implement screen clearing
  - clear screen when space bar is pressed
  - don't clear color picker

THINGS WE NEED TO DO

- ~~Draw 1 color picker square along edge~~
- ~~Implement color picking~~
  - ~~create variable to store current color information~~
  - ~~determine which square is being clicked on~~
  - ~~change color when a different square is clicked~~
- ~~Implement drawing when dragged~~
  - ~~draw when mouse is dragged~~
  - ~~don't draw on color picker~~
- Implement screen clearing
  - clear screen when space bar is pressed
  - don't clear color picker

DO WE NEED TO DO ANYTHING TO IMPLEMENT SCREEN CLEARING?

ALREADY DONE!
WHY DOES IT WORK?

IN keyTyped()

```
public void keyTyped(KeyEvent e) {
    keyPressed = e.getKeyChar();
    System.out.println(keyPressed);
    //clear screen if spacebar is pressed
    if (keyPressed==' ')
        repaint();
}
```

IN paintComponent()

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(Color.WHITE);
    //color picker variables
    int x,y;
    x = width-squareSize-spacing;
    y = spacing;
    //draw color picker
    Color color = new Color(238, 82, 101);
    g.setColor(color);
    g.fillRect(x, y, squareSize,squareSize);
}
```

questions?

THINGS WE NEED TO DO

- Draw all color picker squares along edge
- Implement color picking
  - create variable to store current color information
  - determine which square is being clicked on
  - change color when a different square is clicked
- Implement drawing when dragged
  - draw when mouse is dragged
  - don’t draw on color picker
- Implement screen clearing
  - clear screen when space bar is pressed
  - don’t clear color picker

CREATE A VARIABLE FOR ALL COLORS

```
public class MouseInputPractice extends JPanel implements KeyListener, MouseMotionListener, MouseListener {
    int width;
    int height;
    char keyPressed;
    int keyCode;
    int mouseX, mouseY;
    Graphics g;
    Color drawingColor;
    //color picker variables
    final int squareSize = 50;
    final int spacing = 20;
    final Color[] colors;
```

will store all 7 colors in the color picker
a constant

CREATE A VARIABLE FOR ALL COLORS

```
public class MouseInputPractice extends JPanel implements KeyListener, MouseMotionListener, MouseListener {
    int width;
    int height;
    char keyPressed;
    int keyCode;
    int mouseX, mouseY;
    Graphics g;
    Color drawingColor;
    //color picker variables
    final int squareSize = 50;
    final int spacing = 20;
    final Color[] colors = {new Color(238, 82, 101),
            new Color(238, 147, 82),
            new Color(233, 203, 27),
            new Color(119, 187, 95),
            new Color(82, 163, 238),
            new Color(134, 82, 238),
            new Color(0, 0, 0)};
```

hard code all 7 colors

questions?

USE OUR NEW VARIABLE IN paintComponent()

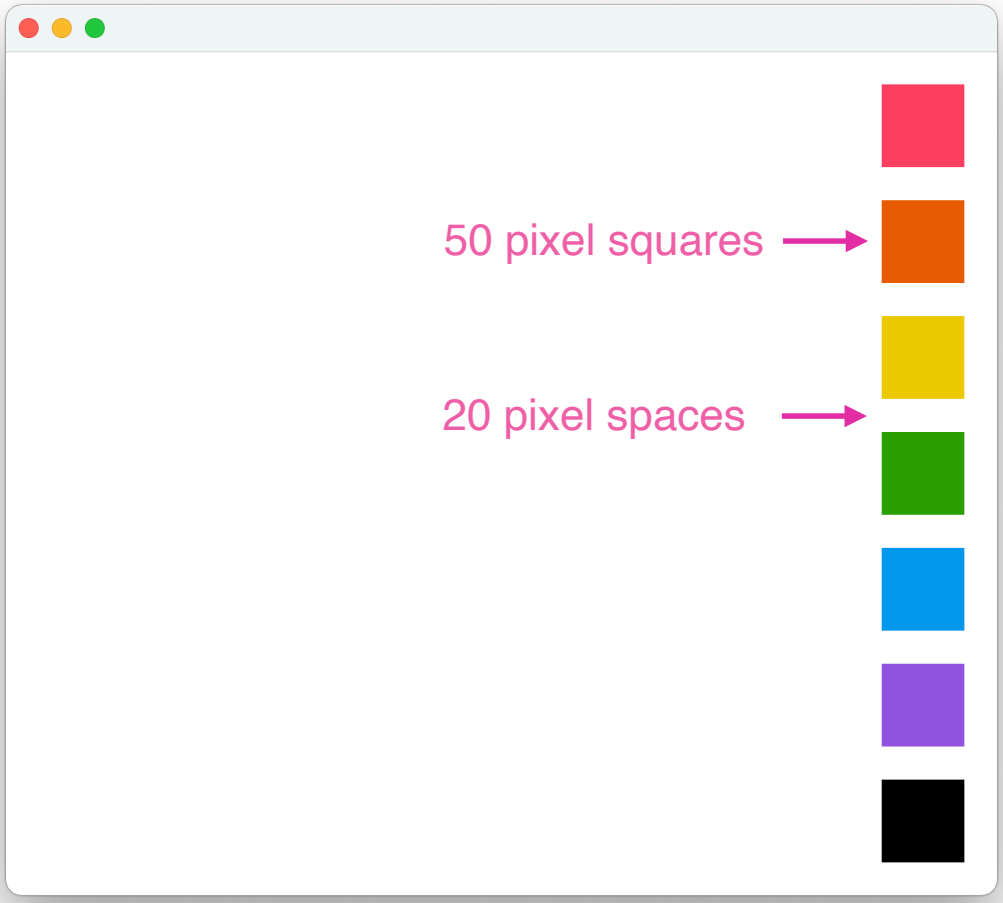
```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    //do your drawing here
    setBackground(Color.WHITE);
    //draw color picker
    Color color = new Color(228, 68, 59);
    g.setColor(color);
    g.fillRect(width-squareSize-spacing,spacing,squareSize,squareSize);
}
```

USE OUR NEW VARIABLE IN paintComponent()

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    //do your drawing here
    setBackground(Color.WHITE);
    //draw color picker
    g.setColor(colors[0]);
    g.fillRect(width-squareSize-spacing,spacing,squareSize,squareSize);
}
```

DRAWING ALL THE SQUARES

THINK ABOUT SPACING



- 7 colors
- size: 50 pixels
- spacing between:
- 20 pixels
- spacing from edge:
  20 pixels

SPACING BETWEEN SQUARES

what is this distance?

d = squareSize + spacing

DRAWING ALL THE SQUARES IN paintComponent()

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(Color.WHITE);
    //color picker variables
    int x,y;
    x = width - squareSize - spacing;
    y = spacing;
    for (int i=0;i<colors.length;i++) {
        g.setColor(colors[i]);

    }
}
```

DRAWING ALL THE SQUARES IN paintComponent()

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(Color.WHITE);
    //color picker variables
    int x,y;
    x = width - squareSize - spacing;
    y = spacing;
    for (int i=0;i<colors.length;i++) {
        g.setColor(colors[i]);
        g.fillRect(x, y, squareSize,squareSize);

    }
}
```

does x change?

does y change?

how does y change?

DRAWING ALL THE SQUARES IN paintComponent()

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(Color.WHITE);
    //color picker variables
    int x,y;
    x = width - squareSize - spacing;
    y = spacing;
    for (int i=0;i<colors.length;i++) {
        g.setColor(colors[i]);
        g.fillRect(x, y, squareSize,squareSize);
        y = y + squareSize + spacing;
    }
}
```

SPACING BETWEEN SQUARES

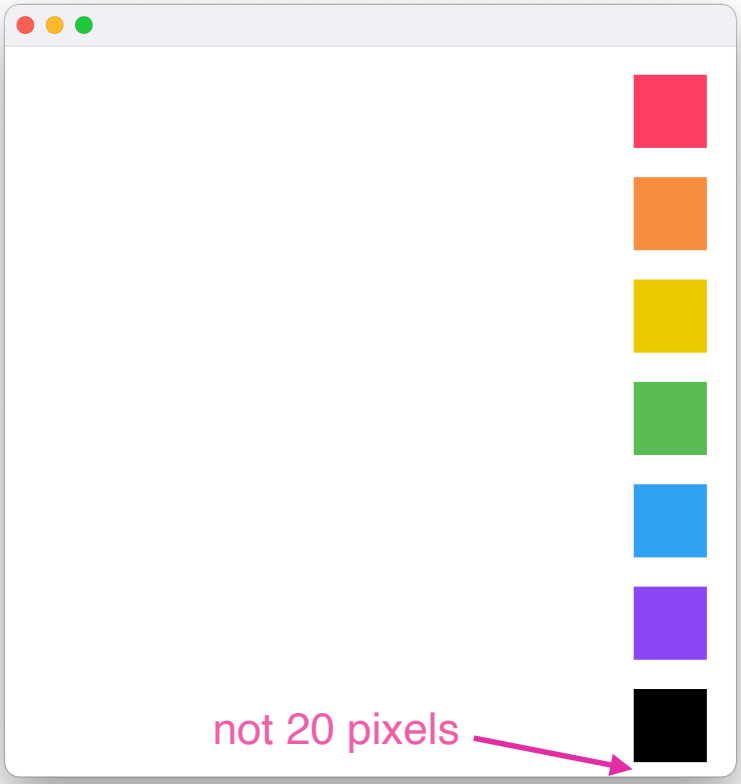
what is this distance?

d = squareSize + spacing

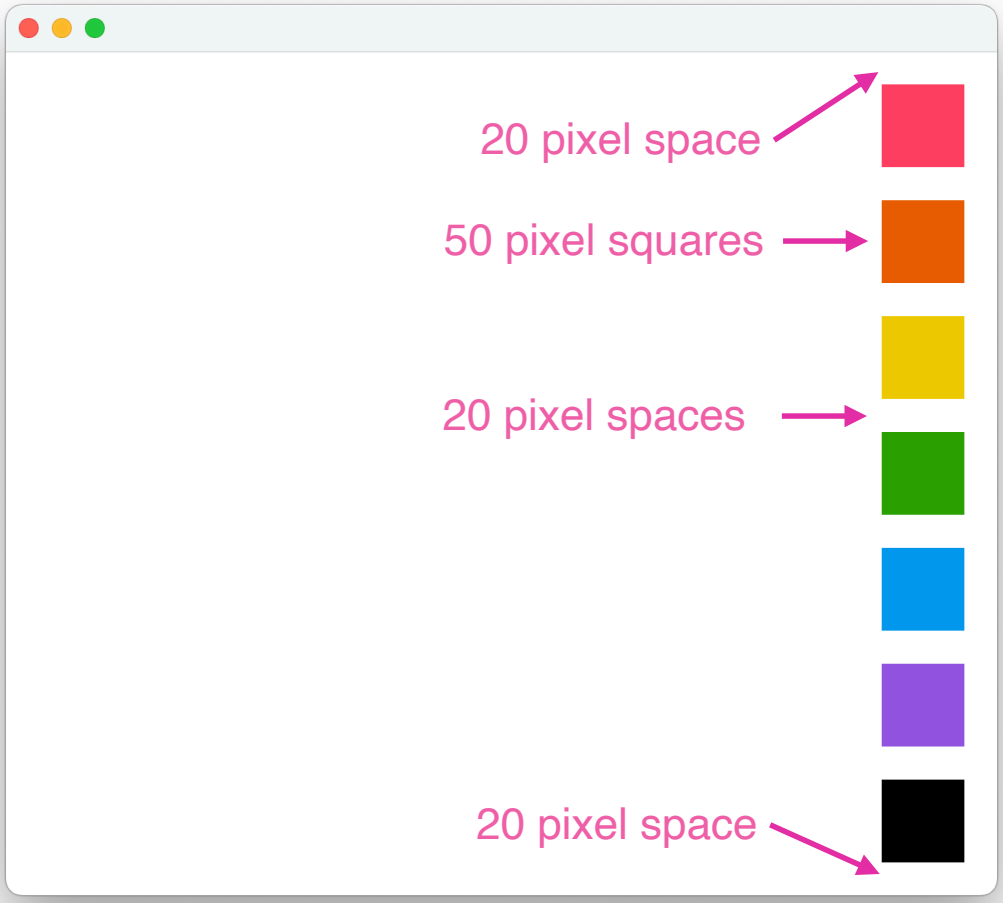
questions?

COMPILE AND RUN

A SMALL PROBLEM



GOOD WIDOW SIZE?




```
50 x 7 = 350
20 x 7 = 140
  + 20 =  20
       = 510
```

EDIT WINDOW SIZE

```
public static void main(String[] args) {
    MouseInputPractice panel = new MouseInput(600, 510);
    MyFrame f = new MyFrame(panel);
    //panel.animate(60);
}
```

IS THERE A BETTER WAY?

A BETTER WAY: CHANGE CONSTRUCTOR

```
MouseInput() {
    width = 600;
    height = 510;
    drawingColor = Color.BLACK;
    Dimension d = new Dimension(width, height);
    setPreferredSize(d);
    addKeyListener(this);
    addMouseMotionListener(this);
    addMouseListener(this);
    setFocusable(true);
    requestFocusInWindow();
    setVisible(true);
    System.out.println("The initial value of mouseX is:" + mouseX);
    System.out.println("The initial value of mouseY is:" + mouseY);
}
```

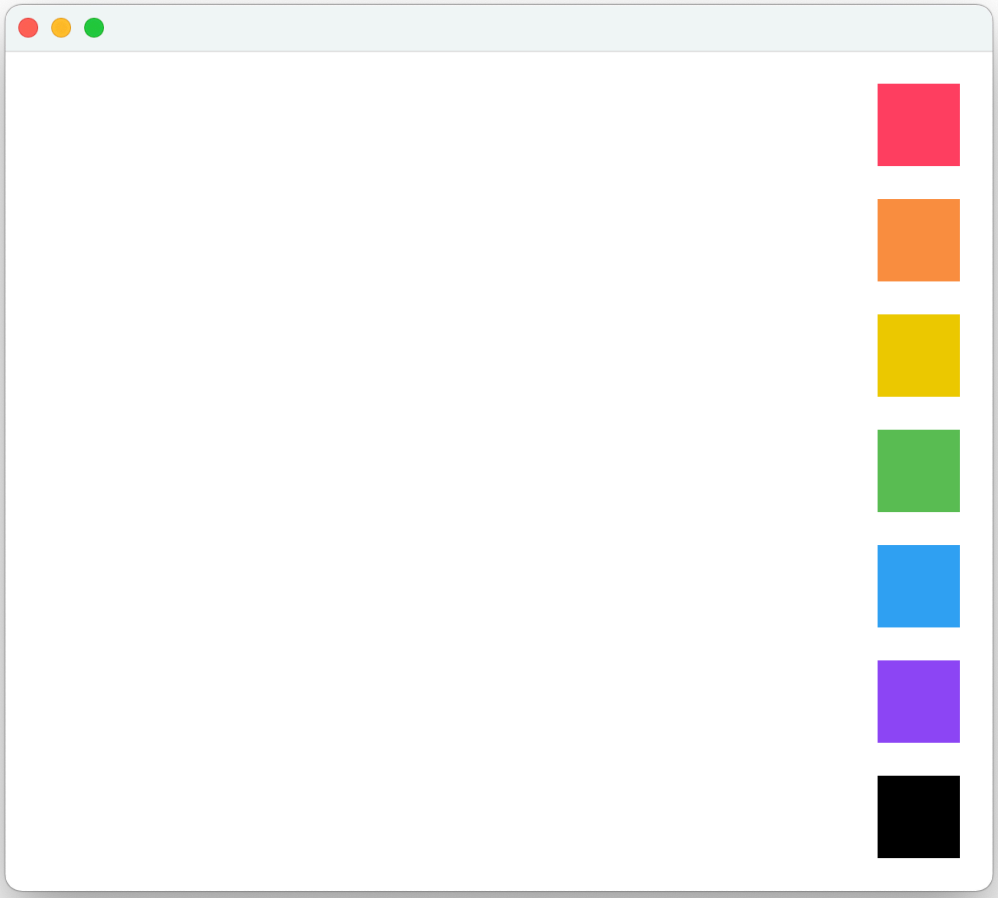
ALSO CHANGE MAIN

```
public static void main(String[] args) {
    MouseInputPractice panel = new MouseInput();
    MyFrame f = new MyFrame(panel);
    //panel.animate(60);
}
```

WHY IS THIS BETTER?

- we are creating a specific application
- the window has a specific size
- to fit our color picker
- don’t want window size to change
- don’t want window size to be a variable

A BETTER WINDOW



THINGS WE NEED TO DO

- ~~Draw all color picker squares along edge~~
- Implement color picking
  - ~~create variable to store current color information~~
  - determine which square is being clicked on
  - change color when a different square is clicked
- Implement drawing when dragged
  - draw when mouse is dragged
  - don’t draw on color picker
- Implement screen clearing
  - clear screen when space bar is pressed
  - don’t clear color picker

WHERE IN OUR CODE DO WE CHECK WHICH SQUARE IS BEING CLICKED ON?

IN mousePressed()

```
public void mousePressed(MouseEvent e) {
    mouseX=e.getX();
    mouseY=e.getY();
    System.out.println("Mouse pressed.");
    g = getGraphics();
    int minX,maxX;
    int minY,maxY;
    minX = width-spacing-squareSize;
    maxX = width-spacing;
    minY = spacing;
    maxY = spacing+squareSize;
    //determine which square is being clicked on
    if (mouseX>minX && mouseX<maxX && mouseY>minY && mouseY<maxY)
        drawingColor = new Color(238, 82, 101);
}
```

IN mousePressed()

```
public void mousePressed(MouseEvent e) {
    mouseX=e.getX();
    mouseY=e.getY();
    System.out.println("Mouse pressed.");
    g = getGraphics();
    int minX,maxX;
    int minY,maxY;
    minX = width-spacing-squareSize;
    maxX = width-spacing;
    minY = spacing;
    maxY = spacing+squareSize;
    //determine which square is being clicked on
    if (mouseX>minX && mouseX<maxX && mouseY>minY && mouseY<maxY)
        drawingColor = colors[0]; //red
}
```

IN mousePressed()

```
public void mousePressed(MouseEvent e) {
    mouseX=e.getX();
    mouseY=e.getY();
    System.out.println("Mouse pressed.");
    g = getGraphics();
    int minX,maxX;
    int minY,maxY;
    minX = width-spacing-squareSize;
    maxX = width-spacing;
    minY = spacing;
    maxY = spacing+squareSize;
    //determine which square is being clicked on
    if (mouseX>minX && mouseX<maxX && mouseY>minY && mouseY<maxY)
        drawingColor = colors[0]; //red
    else if (mouseX>minX && mouseX<maxX && mouseY>(minY+maxY) && mouseY<(maxY+maxY))
        drawingColor = colors[1]; //orange
}
```

IN mousePressed()

```
public void mousePressed(MouseEvent e) {
    mouseX=e.getX();
    mouseY=e.getY();
    System.out.println("Mouse pressed.");
    g = getGraphics();
    int minX,maxX;
    int minY,maxY;
    minX = width-spacing-squareSize;
    maxX = width-spacing;
    minY = spacing;
    maxY = spacing+squareSize;
    //determine which square is being clicked on
    if (mouseX>minX && mouseX<maxX && mouseY>minY && mouseY<maxY)
        drawingColor = colors[0]; //red
    else if (mouseX>minX && mouseX<maxX && mouseY>(minY+maxY) && mouseY<(maxY+maxY))
        drawingColor = colors[1]; //orange
    else if (mouseX>minX && mouseX<maxX && mouseY>(minY+maxY*2) && mouseY<(maxY+maxY*2))
        drawingColor = colors[2]; //yellow
}
```

IN mousePressed()

```
public void mousePressed(MouseEvent e) {
    mouseX=e.getX();
    mouseY=e.getY();
    System.out.println("Mouse pressed.");
    g = getGraphics();
    int minX,maxX;
    int minY,maxY;
    minX = width-spacing-squareSize;
    maxX = width-spacing;
    minY = spacing;
    maxY = spacing+squareSize;
    //determine which square is being clicked on
    if (mouseX>minX && mouseX<maxX && mouseY>minY && mouseY<maxY)
        drawingColor = colors[0]; //red
    else if (mouseX>minX && mouseX<maxX && mouseY>(minY+maxY) && mouseY<(maxY+maxY))
        drawingColor = colors[1]; //orange
    else if (mouseX>minX && mouseX<maxX && mouseY>(minY+maxY*2) && mouseY<(maxY+maxY*2))
        drawingColor = colors[2]; //yellow
    else if (mouseX>minX && mouseX<maxX && mouseY>(minY+maxY*3) && mouseY<(maxY+maxY*3))
        drawingColor = colors[3]; //green
}
```

IS THERE A BETTER WAY?

USE A LOOP!

IN mousePressed()

```
public void mousePressed(MouseEvent e) {
    mouseX=e.getX();
    mouseY=e.getY();
    System.out.println("Mouse pressed.");
    g = getGraphics();
    int minX,maxX;
    int minY,maxY;
    minX = width-spacing-squareSize;
    maxX = width-spacing;
    minY = spacing;
    maxY = spacing+squareSize;
    //determine which square is being clicked on
    for (int i=0;i<colors.length;i++) {
        if (mouseX>minX && mouseX<maxX && mouseY>(minY+maxY*i) && mouseY<(maxY+maxY*i)) {
            drawingColor = colors[i];
        }
    }
}
```

IN mousePressed()

```
public void mousePressed(MouseEvent e) {
    mouseX=e.getX();
    mouseY=e.getY();
    System.out.println("Mouse pressed.");
    g = getGraphics();
    int minX,maxX;
    int minY,maxY;
    minX = width-spacing-squareSize;
    maxX = width-spacing;
    minY = spacing;
    maxY = spacing+squareSize;
    //determine which square is being clicked on
    for (int i=0;i<colors.length;i++) {
        if (mouseX>minX && mouseX<maxX && mouseY>(minY+maxY*i) && mouseY<(maxY+maxY*i)) {
            drawingColor = colors[i];
            break;
        }
    }
}
```

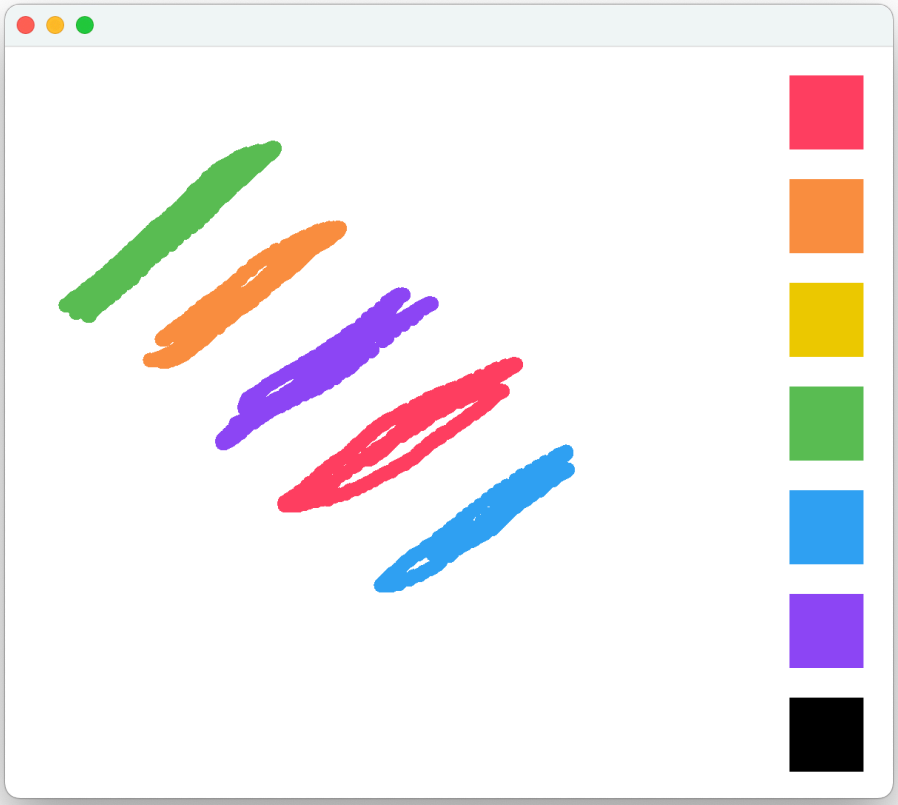
JAVA KEYWORD BREAK

```
for (int i=0;i<colors.length;i++) {
    if (mouseX>minX && mouseX<maxX && mouseY>(minY+maxY*i) && mouseY<(maxY+maxY*i)) {
        drawingColor = colors[i];
        break;
    }
}
```

- “break” breaks out of a loop
- when the keyword break is encountered, program jumps out of current loop and continues executing

questions?

YAY!



THINGS WE NEED TO DO

- ~~Draw all color picker squares along edge~~
- ~~Implement color picking~~
  - ~~create variable to store current color information~~
  - ~~determine which square is being clicked on~~
  - ~~change color when a different square is clicked~~
- Implement drawing when dragged
  - draw when mouse is dragged
  - don't draw on color picker
- Implement screen clearing
  - clear screen when space bar is pressed
  - don't clear color picker

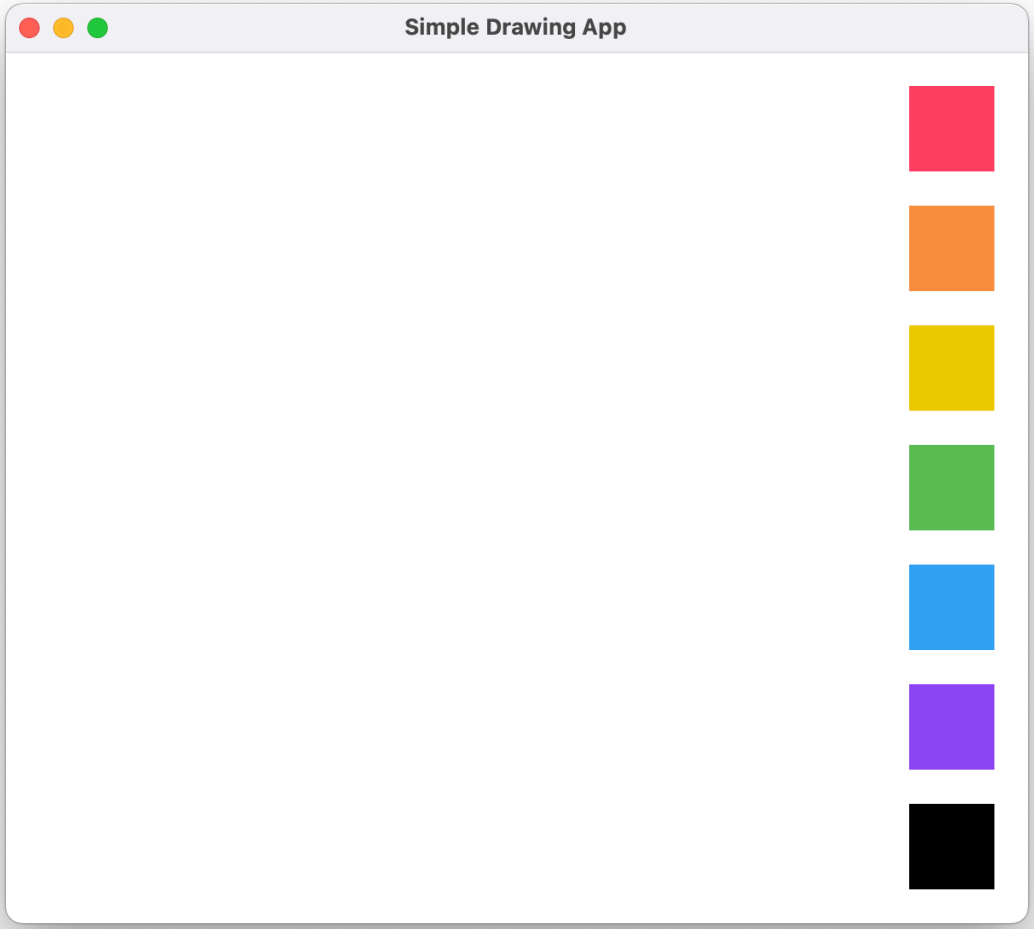
already done!

POLISH UP OUR APPLICATION

ADD A TITLE

IN MyFrame CLASS

```
class MyFrame {
    private JFrame j;
    private JPanel panel;

    MyFrame (JPanel p) {
        panel = p;
        j = new JFrame();
        j.setTitle("Simple Drawing App");
        j.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        j.add(panel);
        j.pack();
        j.setVisible(true);
    }
}
```

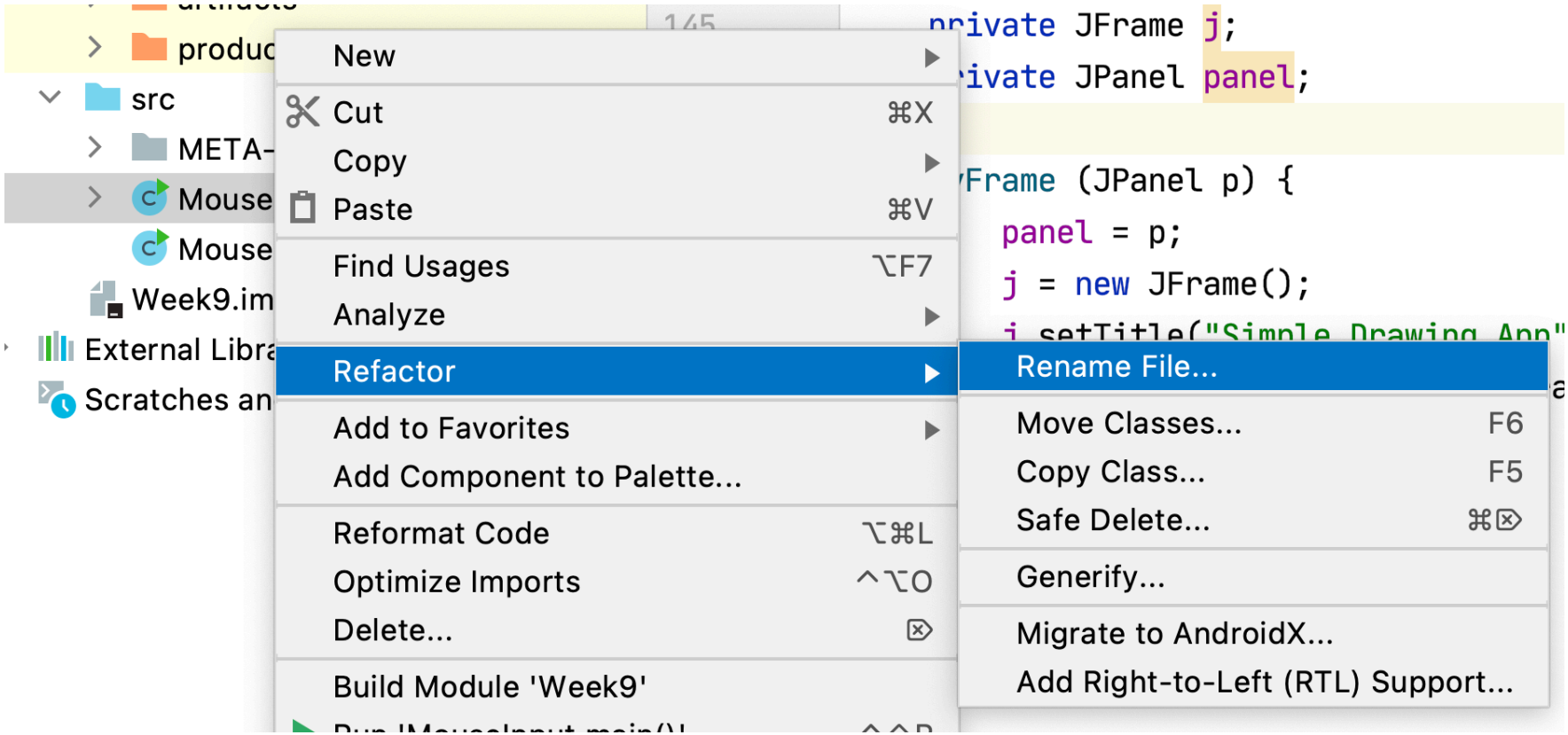

Simple Drawing App

PREVENT RE-SIZING OF WINDOW

IN MyFrame CLASS

```
class MyFrame {
    private JFrame j;
    private JPanel panel;

    MyFrame (JPanel p) {
        panel = p;
        j = new JFrame();
        j.setTitle("Simple Drawing App");
        j.setResizable(false);
        j.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        j.add(panel);
        j.pack();
        j.setVisible(true);
    }
}
```

BETTER CLASS NAME



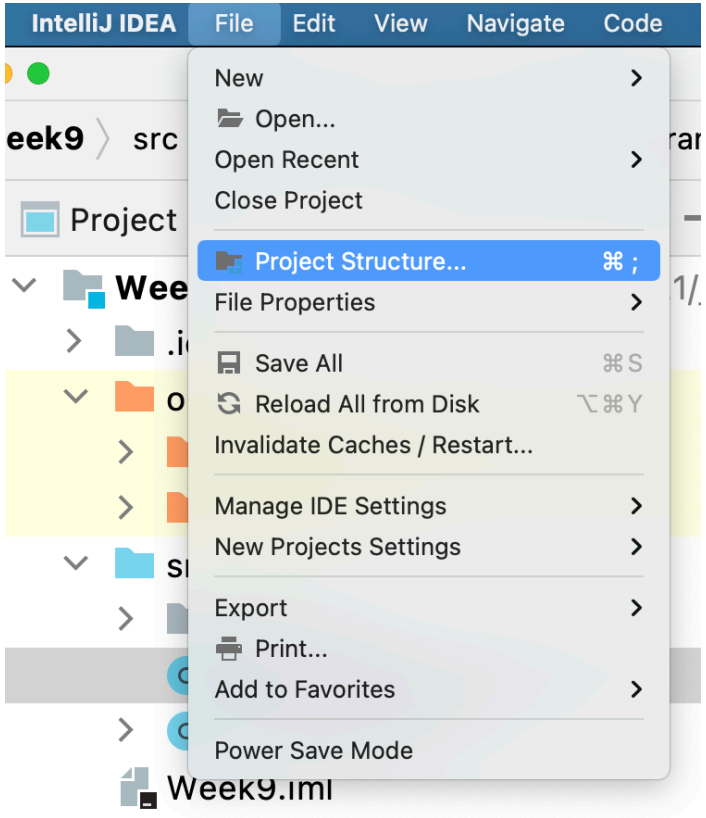
DrawingApplication.java

```
public class DrawingApplication extends JPanel implements KeyListener, MouseMotionListener, MouseListener {
    int width;
    int height;
    char keyPressed;
    int keyCode;
    int mouseX, mouseY;
    Graphics g;
    Color drawingColor;
    //color picker variables
    final int squareSize = 50;
    final int spacing = 20;
    final Color[] colors = {new Color(238, 82, 101),
            new Color(238, 147, 82),
            new Color(233, 203, 27),
            new Color(119, 187, 95),
            new Color(82, 163, 238),
            new Color(134, 82, 238),
            new Color(0, 0, 0)};

    DrawingApplication() {
        width = 600;
        height = 510;
        drawingColor = Color.BLACK;
        Dimension d = new Dimension(width, height);
        setPreferredSize(d);
        addKeyListener(this);
        addMouseMotionListener(this);
        addMouseListener(this);
        setFocusable(true);
        requestFocusInWindow();
        setVisible(true);
        System.out.println("The initial value of mouseX is:" + mouseX);
        System.out.println("The initial value of mouseY is:" + mouseY);
    }

    public static void main(String[] args) {
        DrawingApplication panel = new DrawingApplication();
        MyFrame f = new MyFrame(panel);
        //panel.animate(60);
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        setBackground(Color.WHITE);
        //color picker variables
        int x,y;
        x = width - squareSize - spacing;
        y = spacing;
        for (int i=0;i<colors.length;i++) {
            g.setColor(colors[i]);
            g.fillRect(x, y, squareSize,squareSize);
            y = y + squareSize + spacing;
        }
    }

    void delay(int time) {
        try {
            Thread.sleep(time);
        } catch (Exception exc) {
        }
    }

    void animate(int framerate) {
        int delay = 1000 / framerate;
        while (true) {
            repaint();
            delay(delay);
        }
    }

    @Override
    public void keyTyped(KeyEvent e) {
        keyPressed = e.getKeyChar();
        System.out.println(keyPressed);
        //clear screen if spacebar is pressed
        if (keyPressed == ' ') {
            repaint();
        }
    }
```

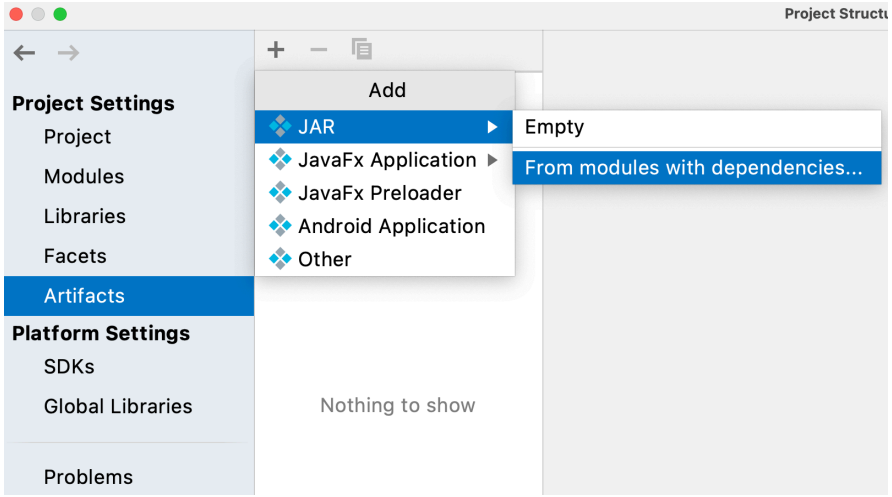
COMPLETE CODE

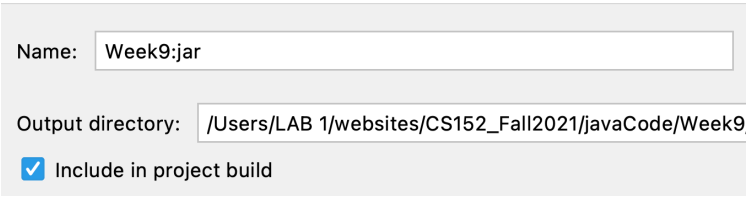
EXPORT AN APPLICATION

OPEN PROJECT STRUCTURE



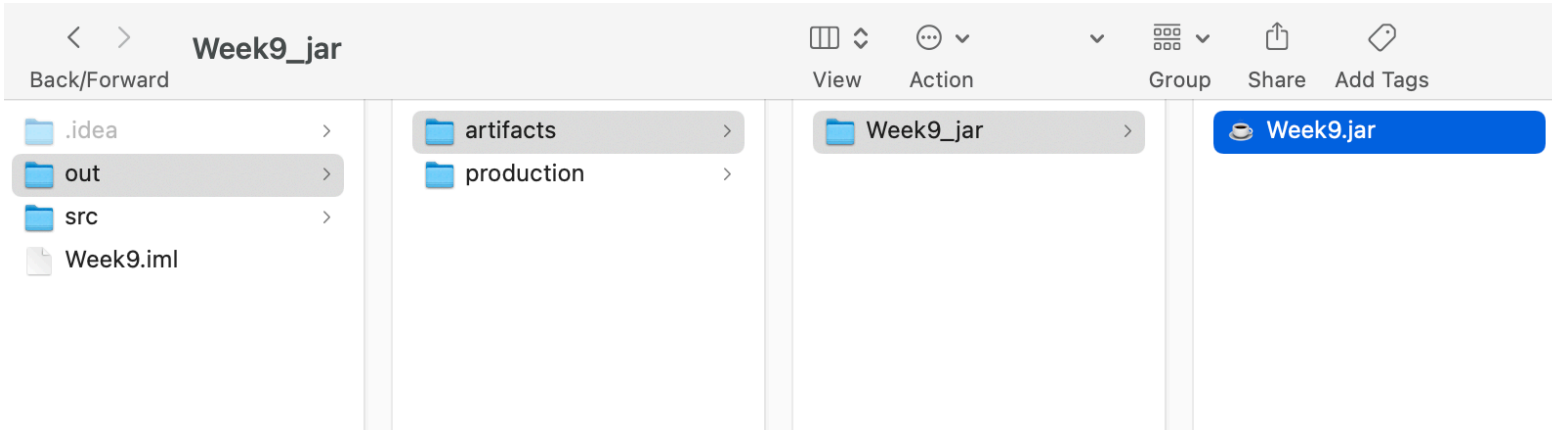
IN PROJECT STRUCTURE





- Under “Artifacts” click +
- Choose JAR —> From modules with dependencies…
- Choose “DrawingApplication” as Main Class
- Check “Include in project build”

BROWSE TO PROJECT FOLDER ON COMPUTER



- Double click on Week9.jar file to run your application

questions?

Thank you!

CS 152
Professor: Leah Buechley
TAs: Melody Horn, Noah Garcia, Andrew Geyko, Juan Ormaza
Time: MWF 10:00-10:50am
https://handandmachine.cs.unm.edu/classes/CS152_Fall2021/