CompSci 230 S1 2020
Object Oriented Software Development

A1 Help

Graphical User Interface (GUI)
Implementing GUIs in Java

- The Java Foundation Classes (JFC) are a set of packages encompassing the following APIs:
  - AWT – Abstract Windows Toolkit (java.awt package)
    - The older version of the components
    - Rely on "peer architecture" : drawing done by the OS platform on which the application/applet is running
    - Considered to be "heavy-weight" components using native GUI system elements
  - Swing (Java 2, JDK 1.2+) (javax.swing package)
    - Newer version of the components
    - No "peer architecture" : components draw themselves
    - Most are considered to be "lightweight" that do not rely on the native GUI or OS

Graphical User Interface (GUI)
GUI elements

- **windows**: actual first-class citizens of desktop; also called top-level containers
  examples: frame, dialog box
- **components**: GUI widgets
  examples: button, text box, label
- **containers**: logical grouping for components
  example: panel

Graphical User Interface (GUI)
Swing component hierarchy

```
java.lang.Object
  +--java.awt.Component
        +--java.awt.Container
              |
              +--javax.swing.JComponent
              |     +--javax.swing.JButton
              |     +--javax.swing.JLabel
              |     +--javax.swing.JMenuBar
              |     +--javax.swing.JOptionPane
              |     +--javax.swing.JPanel
              |     +--javax.swing.JTextArea
              |     +--javax.swing.JTextField
              |
              +--java.awt.Window
                    +--java.awt.Frame
                          +--javax.swing.JFrame
```

Custom Painting

- Create an area for custom painting/drawing inside a JPanel
- Override the paintComponent method

```
public void paintComponent(Graphics g) {
  super.paintComponent(g);
```

  - Note: Call the superclass version of paintComponent as the first statement in the body of the overridden method to ensure that the component displays correctly.
- Note: We don't make a direct call to the paintComponent() method in our code.
  - This method is called **automatically** by the Java runtime whenever the JPanel area needs to be refreshed e.g.
    - when the JFrame is first created and displayed
    - on some platforms the JPanel area is covered (the user moves to another application) and comes back to the JFrame
    - when the user makes a change to the JFrame size

Custom Painting
Graphics & Graphics2D

- Old graphics context: java.awt.Graphics
  - Used in Java 1.0 and 1.1, now obsolete
- New graphics context: java.awt.Graphics2D
  - Part of Java 2D (in Java 1.2 and later)
  - Although paintComponent() takes a Graphics object, what you get is really a Graphics2D!
- Basic methods for painting (Graphics and Graphics2D):
  - drawLine()
  - clearRect(), drawRect(), draw3DRect(), fillRect(), fill3DRect()
  - drawArc(), fillArc(), drawOval(), fillOval()
  - drawPolygon(), fillPolygon(), drawPolyLine()
  - drawString()

```
public void paintComponent(final Graphics g) { ...
  final Graphics2D g2d = (Graphics2D) g; // Just cast it…
  // Use g2d
}
```


Custom Painting
Java 2D

- Support for arbitrary shapes
  - A single draw() method, a single fill()
  - Draws or fills anything implementing
    - Line2D, Rectangle2D, RoundRectangle2D
    - Arc2D, Ellipse2D
    - QuadCurve2D, CubicCurve2D
    - ...
- Pen styles implement the Stroke interface (BasicStroke)
  - Different line widths, patterns, join styles
  - Use setStroke()
- Fill patterns implement the Paint interface
  - Color: Solid fill, default color space sRGB (rgb + alpha)
    - Color.RED, Color.GREEN, Color.BLACK, ...

```
Color cyan2 = new Color(0, 255, 255); // Between 0 and 255
```

    - TexturePaint: Tiles a picture (repeats as necessary)
    - GradientPaint: A gradient between two colors
  - Use setPaint() or the older setColor()

Custom Painting
Drawing Shapes

- In order to draw in the JPanel area we use the Graphics object. The Graphics object is supplied by the Java runtime as a parameter to the paintComponent() method. The Graphics class contains many instance methods:

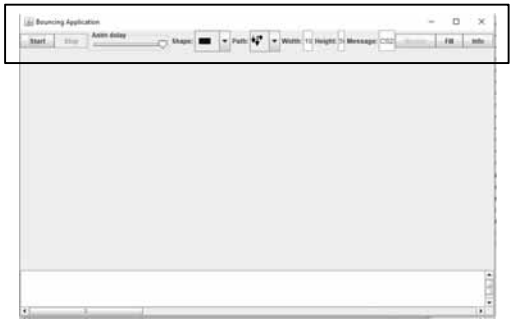
```
drawLine(int x1, int y1, int x2 , int y2)
drawRect(int x, int y, int width, int height)
drawOval(int x, int y, int width, int height)
fillRect(int x, int y, int width, int height)
fillOval(int x, int y, int width, int height)
drawString(String text, int x, int y)
```

```
setColor(Color color)
```

```
g.setColor(Color.YELLOW);
```

A1 Class

- Layout:
  - Top: toolsPanel
    - stores the current properties: e.g. shape type, fill colour, border colour..
    - controls the animation, eg. start, stop, adjust the speed
  - Middle: AnimationPanel
    - Shapes bouncing area



- Note: You don't need to make any changes to the A1 class!

The Bouncing program - Background

- Animation:
  - animationThread.start()
  - execute run()
  - execute repaint()
  - execute paintComponent()
    - Loop through the shapes and execute the move() and draw() method
      - move() of the MovingRectangle
        - call path.move of a path
        - change the x and y position (i.e. top-left corner)
        - Note: MovingPath is an Inner class of the shape, it can access and change the x, y coordinates
      - draw() of the MovingRectangle
        - call the draw method
        - draw the shape and handles if selected

```
public void paintComponent(Graphics g) {
   for (MovingShape currentShape: shapes) {
     currentShape.move();
     currentShape.draw(g);
   }
}
```

The Bouncing program - Background

- Adding a new shape
  - mouse click within the AnimationPanel area
  - Fire the mouseClicked event
    - If not selected
      - createNewShape(e.getX(), e.getY()) – at mouse point
        - Get all current values: shape, path, width, height …
        - Create a new instance and add it to the shapes array
    - If selected
      - Set the selected boolean to true

```
public void mouseClicked( MouseEvent e ) {
 ...
  if (!found)
    createNewShape(e.getX(), e.getY());
}
```

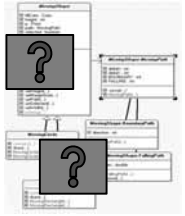
```
protected void createNewShape(int x, int y) {
. . .
  switch (currentShapeType) {
    case 0: {
      shapes.add( new MovingRectangle(x, y, . . .);
      break;
    }
```

Tasks

- Task 1: MovingRectangle (6 marks): 4 (CR)+ 2 marks
- Task 2: MovingSquare (6 marks): 4 (CR)+ 2
- Task 3: MovingEllipse (6 marks): 4 (CR)+ 2
- Task 4: MovingStarsMap (6 marks): 4 (CR)+ 2
- Task 5: MovingSpinningCircle (6marks): 4 (CR)+ 2
- Task 6: The message property (8 marks): 4 (CR) + 4
- Task 7: The BoundaryPath (5 marks): 2 (CR) + 3
- Task 8: Sorting (5 marks): 5 (CR)
- Include your name, UPI and a comment at the beginning of ALL YOUR FILES: 0 (CR) + 2
- Total = 50 marks

Task 1 - MovingRectangle

- Create a new Class
  - The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice.
    - Extends …
  - Implement THREE Constructors
  - draw(): draws a rectangle shape
  - contains(): checks if a mouse point is within the rectangle
  - getArea(): returns the area of a rectangle
  - Add a new case in the createNewShape() method in AnimationPanel



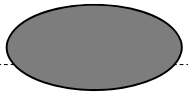
- Check the following:
  - New shape is drawn with the current values.
  - Users should be able to change the properties of selected shapes.

Task 2 - MovingSquare

- Create a new Class
  - The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice.
    - Should you extends MovingShape or extends MovingRectangle?
    - Should you implement the draw()/contains()/getArea() in MovingSquare?
  - Implement THREE constructors
  - Add a new case in the createNewShape() method in AnimationPanel

- Check the following:
  - New shape is drawn with the current values (i.e. min of width and height)
  - Users should be able to change the properties of selected shapes.
    - Eg. if a 50x50 square is selected and we change the height to 100. you should have a 100x100 square now

Task 3: MovingEllipse



- Create a new Class
  - The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice.
    - Should you extends MovingShape or MovingRectangle or MovingSquare
    - Should you implement the draw()/contains()/getArea()?
  - Implement THREE constructors
  - Add a new case in the createNewShape() method in AnimationPanel

- Check the following
  - New shape is drawn with the current values.
  - Users should be able to change the properties of selected shapes.

Task 4: MovingStarsMap

- Create a new Class
  - The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice.
    - Instance variable: ArrayList<Point> points
    - Should you extends MovingShape or MovingRectangle or MovingSquare
    - Should you implement the draw()/contains()/getArea()
  - Implement THREE constructors
    - create 3 point objects randomly in the constructor
  - Add a new case in the createNewShape() method in AnimationPanel

- Check the following
  - New shape is drawn with the current values
  - Users should be able to change the properties of selected shapes.

Task 4: MovingSpinningCircle

- Create a new Class
  - The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice.
    - Instance variable: startAngle = 0
    - constant: ROTATION_SPEED = 20
    - Should you extends MovingShape or MovingRectangle or MovingEllipse
    - Should you implement the draw()/contains()/getArea()
  - Implement THREE constructors
  - Add a new case in the createNewShape() method in AnimationPanel
- Check the following
  - New shape is drawn with the current values (i.e. min of width and height)
  - Users should be able to change the properties of selected shapes.
    - Eg. if a 50x50 circle is selected and we change the height to 100. you should have a 100x100 circle now

The message field

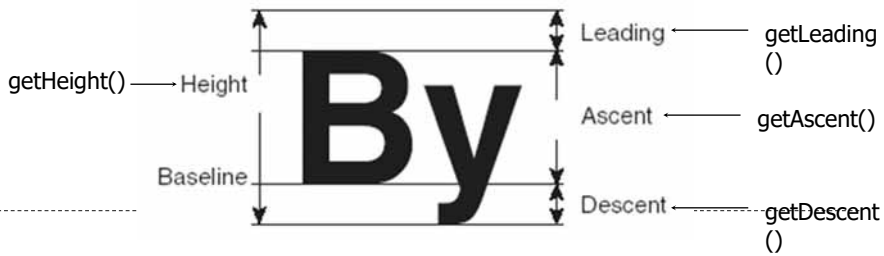
CS230

- draw a message at the centre
- MovingShape:
  - add an instance field, add the get/set methods
  - add the drawMessage() method
  - modify all constructors in the MovingShape class
- AnimationPanel:
  - add an instance field: currentMessage and the get/set methods
  - modify the createNewShape method
- Modify all constructors in all subclasses

Using the FontMetrics Class

- Use the FontMetrics class to measure the exact width and height of the string for a particular font.
  - FontMetrics fm = g2d.getFontMetrics();

- public int getAscent()
- public int getDescent()
- public int getLeading()
- public int getHeight()
- public int stringWidth(String str)



Task 5: The BoundaryPath

- Create a new Inner Class
  - The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice.
    - Extends …
  - Implement constructor
  - Override the move method
    - boundary: marginHeight, marginWidth
    - Four direction: downward, upward, left & right
  - Add a new case in the setPath() method in MovingShape
- Check the following:
  - New shape is bouncing using the new path.
  - Modify selected shapes to be bounced using the new path.

Sorting

- provide information on a list of **sorted** MovingShape objects.
- implement Comparable<MovingShape> interface
- implement compareTo(MovingShape other) method
  - based on the area() of MovingShape object