Tutorial of Swing

> source code are based on lab materials of "sustech-teaching group"
>
> Document designed by ZHU Yueming in 2023. May. 6th

Objective

- Can use the painting mechanism of swing
- Can use the mouse event in swing
- Understand addActionListener in swing

Introduction

1. painting mechanism

All subclasses of `JComponent` can override a method `paintComponent`, in which you can paint anything you want in current component.

```java
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
}
```

The method cannot be invoked directly by us, while you can invoke `repaint()` method to execute the `paintComponent` method.

For example: when we invoke the `shrink()` method, it repaint a smaller circle.

```java
   /**
     * Enlarge the circle
     */
    public void shrink() {
        radius = (int) (radius * 0.9);
        this.repaint();
    }

    /**
     * when doing repaint() method, execute paintComponent method
     */
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(this.color);
        g.drawString(String.format("Radius: %d",this.radius),10,15);
        g.fillOval(this.getWidth() / 2 - radius, this.getHeight() / 2 - radius,
2 * radius, 2 * radius);
    }
```

2. mouse event

If one component has been add mouse click event, when it being clicked by mouse, the method processMouseEvent can execute immediately.

Add following code in component, which means the mouse click event:

```
enableEvents(AWTEvent.MOUSE_EVENT_MASK);
```

Design following method in the same component above, which mean the action would be processed as soon as the component is clicked:

```
 @Override
    protected void processMouseEvent(MouseEvent e) {
        super.processMouseEvent(e);
        if (e.getID() == MouseEvent.MOUSE_PRESSED) {
            color = new Color(random.nextInt(255), random.nextInt(255),
random.nextInt(255));
            System.out.println(color);
            repaint();
        }
    }
```

3. Listener in button

The `jbtEnlarge` and `jbtShrink` are `JButton` components.

```
JButton jbtEnlarge = new JButton("Enlarge");
JButton jbtShrink = new JButton("Shrink");
//add click listener into button: jbtEnlarge
jbtEnlarge.addActionListener(l -> {
      canvas.enlarge();//when the button clicked, do enlarge method.
});
//add click listener into button: jbtShrink
jbtShrink.addActionListener(l -> {
      canvas.shrink();//when the button clicked, do shrink method.
});
```

All statements that you want to execute after clicking the button should be written in `{}`.

```
btn.addActionListener(l -> {
      //todo: writing all statements that you want to execute after clicking
this btn
});
```

Exercise

Similar to the Mouse Event, add KeyEvent to process the functions below:

- Press UP key to enlarge the circle
- Press DOWN key to shrink the circle

Hint: The KeyEvent should be added in `CirclePanel` class, and we should set the `CirclePanel` class to be focusable. You can use following statement in constructor in `CirclePanel`:

```
this.setFocusable(true);
```

Solution 1: enableEvents(AWTEvent.KEY_EVENT_MASK);

Step 1: Add enable Key Event in constructor in `CirclePanel`

```
enableEvents(AWTEvent.KEY_EVENT_MASK);
```

Step 2: Implement method in `CirclePanel`

```
 @Override
    protected void processKeyEvent(KeyEvent e) {
        super.processKeyEvent(e);
        if(e.getID()==KeyEvent.KEY_PRESSED) {
            //todo: finish the exercise
        }
    }
```

Solution 2: Add KeyListener

Step 1: Add keyListener in constructor in `CirclePanel`

```java
this.addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                super.keyPressed(e);
                //todo:: finish the method
            }
        });
```