案例10：冒泡的小球

2022年7月11日 8:39

一、案例需求

1. 需求：创建窗口，设置窗口的相应属性（大小、标题、默认关闭、可见）；创建画布、绘制一个小球；当鼠标的指针经过窗口上方时，小球会跟着光标移动，并且当点击鼠标的时候，小球会冒泡，单击一下冒一个泡，如果鼠标点击之后一直不松开就会一直冒泡
2. 分析：
   a. 创建窗体，在窗体上添加创建好的画布，并且在画布上绘制一个小球；
   b. 鼠标的光标移动时可以控制小球的移动；
   c. 鼠标单击时，先有的小球会开始冒泡（向上产生新的较小的小球），点击一下产生一个新的冒泡的小球、一直按压则一直产生新的冒泡的小球

二、案例具体实现步骤以及代码

1. 窗体以及画布：两个类：ShootBubbleFrame、ShootBubblePanel，参考之前的案例即可
2. 在画布ShootBubblePanel中完成绘制一个大的小球，并且可以通过移动鼠标的光标来控制小球的移动；添加到窗口上
   a. 绘制小球：

```
//    小球的x\y坐标
    private int x , y;
//    小球的半径
    private int r;

//    构造方法完成初始化
    public ShootBubblePanel() {
        x = 200;
        y = 180;
        r = 30;
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        this.setBackground(Color.DARK_GRAY);
        g.setColor(Color.pink);
        g.fillOval(x, y, 2*r, 2*r);
    }
```

   b. 小球跟随鼠标指针移动：ShootBubblePanel实现鼠标的动作接口MouseMotionListener，并重写其中的拖拽以及移动的方法：

```
@Override
public void mouseDragged(MouseEvent e) {
    x = e.getX();
```

```
        y = e.getY();
        repaint();
    }

    @Override
    public void mouseMoved(MouseEvent e) {
        x = e.getX();
        y = e.getY();
        repaint();
    }
```

然后在窗口上注册此鼠标监听器：

```
frame.addMouseMotionListener(panel);
```

实现了小球跟随鼠标移动之后，发现此时鼠标

实现了小球的移动之后，发现此时鼠标的指针和小球的位置有一定的偏差，纠正此偏差：

```
x = e.getX() - r;
y = e.getY() - r;
```

3. 鼠标单击时，先有的小球会开始冒泡（向上产生新的较小的小球），点击一下产生一个新的冒泡的小球、一直按压则一直产生新的冒泡的小球；此时需要不停的产生新的较小的用于冒泡的小球！所以将冒泡的小球抽取为一个小球实体类Bubble
   a. 创建好小球实体类Bubble：

```
/**
 *      小球实体类：
 *      属性：坐标、半径、颜色、移动方向（UP向上-4）、移动速度、所在画布
 *      方法：绘制、移动（扩展一个向上）
 * @author tarena
 *
 */
public class Bubble {

    public final static int LEFT_UP=0;
    public final static int LEFT_DOWN=1;
    public final static int RIGHT_UP=2;
    public final static int RIGHT_DOWN=3;
    public final static int UP=4;

    private int x,y;
    private int r;
    private Color color;
    private int orientation;
    private int speed;
    private JPanel panel;
```

```
public Bubble() {
}

public Bubble(int x, int y, int r, Color color, int orientation, int speed, JPanel panel) {
    this.x = x;
    this.y = y;
    this.r = r;
    this.color = color;
    this.orientation = orientation;
    this.speed = speed;
    this.panel = panel;
}

public int getX() {
    return x;
}

public void setX(int x) {
    this.x = x;
}

public int getY() {
    return y;
}

public void setY(int y) {
    this.y = y;
}

public int getR() {
    return r;
}

public void setR(int r) {
    this.r = r;
}

public Color getColor() {
    return color;
}

public void setColor(Color color) {
```

```
        this.color = color;
    }

    public int getOrientation() {
        return orientation;
    }

    public void setOrientation(int orientation) {
        this.orientation = orientation;
    }

    public int getSpeed() {
        return speed;
    }

    public void setSpeed(int speed) {
        this.speed = speed;
    }

    public JPanel getPanel() {
        return panel;
    }

    public void setPanel(JPanel panel) {
        this.panel = panel;
    }

    public void draw(Graphics g) {
        g.setColor(color);
        g.fillOval(x, y, 2*r, 2*r);
    }

    public void move() {
        switch(orientation) {
        case UP:
            y--;
            break;
        case LEFT_UP:
            x--;
            y--;
            break;
        case LEFT_DOWN:
```

```
                x--;
                y++;
                break;
            case RIGHT_UP:
                x++;
                y--;
                break;
            case RIGHT_DOWN:
                x++;
                y++;
                break;
            }
        }
    }
```

b. 实现单击鼠标的时候，小球可以向上冒泡：

i. 首先在ShootBubblePanel类中声明一个存放冒泡小球对象的数组：

```
//      可以存放冒泡小球对象的数组；
        private Bubble [] allBubbles ;
```

然后在构造方法中实例化此数组对象：

```
allBubbles = new Bubble[length];
```

ii. 此时会存在一个问题：冒泡小球数组的初始大小应该为多少，设置一个固定值时可能会存在数组的容量不够或者数组容量过大的问题！

考虑将数组改为使用Vector集合（底层本质就是数组）来实现，集合可以存储任意类型的对象并且存储不定长的元素：

```
//      可以存放冒泡小球对象的集合属性，可以存储不同类型的任意个小球对象
        private Vector allBubbles;
//      构造方法完成初始化
        public ShootBubblePanel() {
            x = 200;
            y = 180;
            r = 30;
//          实例化此Vector集合对象：初始大小容量为10
            allBubbles = new Vector();
        }
```

iii. 分析：private Vector allBubbles= new Vector(); 的底层代码实现

1) 第一步调用无参构造：

```
    public Vector() {
      //默认的初始容量为10
       this(10);
    }
```

2) 第二步调用传入一个int类型参数的构造：

```
    public Vector(int initialCapacity) {
```

```
        this(initialCapacity, 0);
    }
```

3) 第三步调用传入两个int类型参数的构造：

```
    protected Object[] elementData;
    public Vector(int initialCapacity, int capacityIncrement) {
        super();
      //判断初始容量小于0，抛出异常
        if (initialCapacity < 0)
            throw new IllegalArgumentException("Illegal Capacity: "+
                                               initialCapacity);
      //创建一个传入容量大小（10）的Object类型数组，因此可以存储任意类型的对象
        this.elementData = new Object[initialCapacity];
        this.capacityIncrement = capacityIncrement;
    }
```

iv. 在画布类的paint方法中完成Vector集合中取出冒泡小球对象，并绘制出冒泡小球：

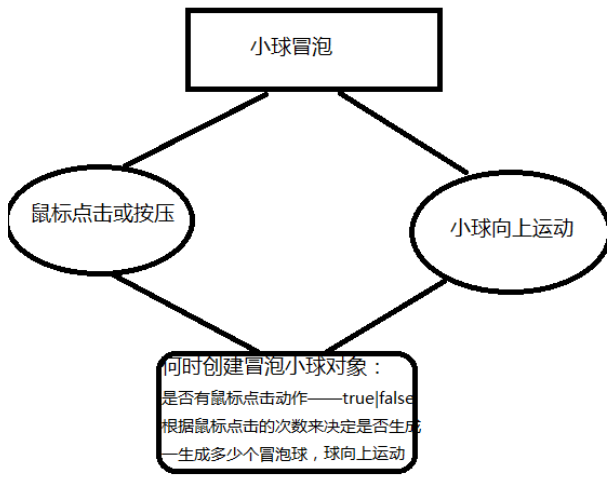
```
//              通过集合提供的size方法获取集合的元素的个数
        for (int i = 0; i < allBubbles.size(); i++) {
            Bubble bubble = (Bubble) allBubbles.elementAt(i);
            bubble.draw(g);
        }
```

考虑此时集合中目前还不存在冒泡小球对象，先来采用碰撞小球的方式实现向集合中存放冒泡小球：

```
//通过集合提供的capacity()方法获取集合的初始容量
        for (int i = 0; i < allBubbles.capacity(); i++) {
//采用之前碰撞小球案例中的实例化创建小球的方式
Bubble bubble =
                        new Bubble(100, 100, 10, Color.orange, Bubble.UP, 10, this);
            allBubbles.add(bubble);
        }
```

? 此时存在的问题：还未进行任何鼠标点击或按压等动作就已经画出了冒泡小球并且生成的所有的小球都在同一个位置处；不符合需求

v. 条件解析：



1) 在ShootBubblePanel类中添加一个标记符isMao，初始状态为false

```
//    小球是否是冒泡状态：
    private boolean isMao;
//    构造方法完成初始化
    public ShootBubblePanel() {
        x = 200;
        y = 180;
        r = 30;
//          实例化此Vector集合对象：初始大小容量为10
        allBubbles = new Vector();
//          初始应为false；
        isMao = false;
    }
```

2) 再让ShootBubblePanel类实现MouseListener接口，并重写其中的鼠标点击、按压、释放等方法

```
public class ShootBubblePanel extends JPanel implements
MouseMotionListener,MouseListener{
    @Override
    public void mouseClicked(MouseEvent e) {
//          点击并冒泡，修改冒泡状态为true
        isMao = true;
    }

    @Override
    public void mousePressed(MouseEvent e) {
//          按压并冒泡，修改冒泡状态为true
        isMao = true;
    }

    @Override
```

```
    public void mouseReleased(MouseEvent e) {
//            释放并停止冒泡，修改为false
        isMao = false;
    }

    @Override
    public void mouseEntered(MouseEvent e) {
    }

    @Override
    public void mouseExited(MouseEvent e) {
    }
}
```

添加此鼠标监听器到窗口上：

```
frame.addMouseListener(panel);
```

3) 再让ShootBubblePanel类实现Runnable接口 ，重写run方法，在其中根据鼠标动作标记符来决定是否创建冒泡小球，完成小球的创建以及属性初始化

```
public class ShootBubblePanel extends JPanel implements Runnable,
MouseMotionListener,MouseListener{
    //    创建冒泡小球：属性初始化以及冒泡小球向上运动
        @Override
        public void run() {
            while(true) {
    //                isMao=true表示鼠标点击或按压，完成冒泡小球创建
                if(isMao) {
                    Bubble bubble = new Bubble();
                    bubble.setR(10);
                    bubble.setColor(Color.orange);
                    bubble.setOrientation(Bubble.UP);
    //                    冒泡小球应该在大的小球的上方产生
                    bubble.setX(x+r-bubble.getR());
                    bubble.setY(y);
                    bubble.setPanel(this);
    //                    每生成一个冒泡小球就存储到集合allBubbles当中
                    allBubbles.add(bubble);
                }
    //                遍历冒泡小球，并向上运动：size()表示此时集合中存储的元素个数
                for (int i = 0; i < allBubbles.size(); i++) {
                    Bubble b = (Bubble) allBubbles.elementAt(i);
                    b.move();
                }
    //                每个冒泡小球运动间隔20毫秒
```

```
                    try {
                        Thread.sleep(20);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            }
        }
```

在ShootBubbleFrame中完成线程的启动：

```
Thread t = new Thread(panel);
t.start();
```

4) 此时运行程序，可以实现鼠标点击按压然后产生冒泡小球的需求；但是此时产生的冒泡小球没入窗体边框的顶端之后，依旧存在于allBubbles集合当中，会导致后续程序中遍历代码越跑越慢；考虑通过在for循环中判断冒泡小球是否越界，如果越界了就从集合中移除该冒泡小球，未越界则继续向上移动：

```
for (int i = 0; i < allBubbles.size(); i++) {
    Bubble b = (Bubble) allBubbles.elementAt(i);
    //判断当前冒泡小球的y轴坐标是否小于等于0，是则移除
    if(b.getY() <= 0) {
        allBubbles.remove(i);
    }
    b.move();
}
```

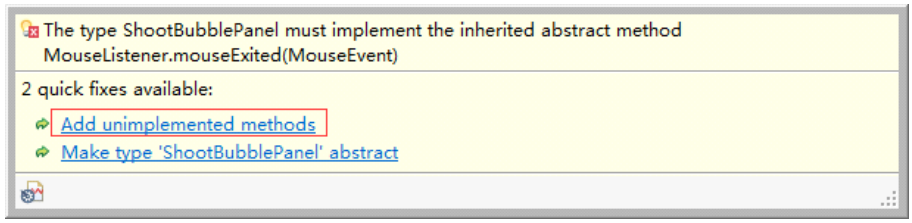
5) 最终优化：
   a) 生成冒泡小球的间隔优化，定义一个计数器，处理为count为25的倍数时才生成冒泡小球：
      i) 在run方法中定义一个int count = 0;
      ii) 在while循环的if判断中增加一个count值的判断：if(isMao & count%25 ==0)
      iii) 在while循环的代码块当中添加计数器递增：count++;
   b) 当点击一次之后再释放鼠标按钮时，应该是冒泡小球不再跟随鼠标移动而生成：
      i) 在mouseMove方法中将鼠标动作标记符设置为false：isMao = false;
      ii) 在mouseDragged方法将鼠标动作标记符设置为true：isMao = true;

```
    @Override
    public void mouseDragged(MouseEvent e) {
//      拖拽的同时会生成冒泡小球
        isMao = true;
        x = e.getX() - r;
        y = e.getY() - r;
        repaint();
    }

    @Override
    public void mouseMoved(MouseEvent e) {
//      无按压仅仅是鼠标指针移动时，只有大球跟随，而不生成冒泡小球
        isMao = false;
        x = e.getX() - r;
        y = e.getY() - r;
        repaint();
    }
```

三、案例知识点总结：

1. 鼠标动作接口：MouseMotionListener、MouseListener
   a. 画布类先实现：implements MouseMotionListener,MouseListener
   b. 画布类名字上根据提示直接添加未实现的方法即可



   c. 保存，然后再去编辑方法中的业务逻辑
   d. 在窗口类当中添加上鼠标监听器接口：
   frame.addMouseMotionListener(panel);
   frame.addMouseListener(panel);
2. 线程接口：Runnable
   a. 画布类先实现：implements Runnable
   b. 根据提示直接添加未实现的run方法即可
   c. 保存，然后再去编辑方法中的业务逻辑
   d. 在窗口类当中添加上线程的启动：
   Thread t = new Thread(panel);
   t.start();
3. 集合的一个实现类：Vector；具有可以存储任意类型的元素，也即底层创建了一个存储Object类型元素的数组、并且长度不固定会跟随所存入的元素个数而增加
   ★a. 分析：private Vector allBubbles= new Vector(); 的底层代码实现
      1) 第一步调用无参构造：

```
public Vector() {
  //默认的初始容量为10
    this(10);
}
```

      2) 第二步调用传入一个int类型参数的构造：

```
public Vector(int initialCapacity) {
    this(initialCapacity, 0);
}
```

3) 第三步调用传入两个int类型参数的构造：

```
protected Object[] elementData;
public Vector(int initialCapacity, int capacityIncrement) {
    super();
   //判断初始容量小于0，抛出异常
    if (initialCapacity < 0)
        throw new IllegalArgumentException("Illegal Capacity: "+
                                           initialCapacity);
   //创建一个传入容量大小（10）的Object类型数组，因此可以存储任意类型的对象
    this.elementData = new Object[initialCapacity];
    this.capacityIncrement = capacityIncrement;
}
```

b. 方法：

1) elementAt(int index)：根据所传入的下标值获取对应位置处的元素；在案例10当中获取到的元素类型是Object，要其复制给小球类Bubble对象就需要进行强转
2) size()：获取集合的大小，也即获取集合中元素的个数
3) capacity()：获取集合的初始容量 10
4) add(Object obj)：存储传入的某个类型的元素对象；在案例10 当中传入小球类Bubble对象
5) remove(int index)：根据所传入的下标值移除对应位置处的元素