# 案例7：碰撞的小球

2022年7月9日 8:40

一、碰撞后变大的优化：极端情况值的处理

1. 一个小球变大、另一个小球消失的代码：

```
if(isHit) {
    //碰撞之后将一个小球的半径变为0，另一个小球的半径加10(或者另一个小球的半径)
    allBalls[1].setR(allBalls[1].getR()+allBalls[0].getR());
    allBalls[0].setR(0);
    allBalls[0].setX(0);
    allBalls[0].setY(0);
}
```

因为其中一个小球对象并没有真正的消失，可能会导致另一个小球持续变大；

需要修改代码为将其中一个小球对象置为空null，然后在所有需要调用小球之处进行小球是否为空的判断

a. 现将碰撞之后的代码修改为其中一个对象置为null

```
if(isHit) {
    //碰撞之后将一个小球对象置空，另一个小球的半径加10(或者另一个小球的半径)
    allBalls[1].setR(allBalls[1].getR()+allBalls[0].getR());
    allBalls[0] = null;
}
```

b. paint方法的for循环中：

```
for (int i = 0; i < allBalls.length; i++) {
    if(allBalls[i] != null) {
        allBalls[i].draw(g);
    }
}
```

c. run方法的while循环里的for循环中：

```
for (int i = 0; i < allBalls.length; i++) {
    if(allBalls[i] != null) {
        allBalls[i].move();
        try {
            Thread.sleep(allBalls[i].getSpeed());
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
//              调用判断两个球相撞的方法
        boolean isHit = twoBallsHit(allBalls[0],allBalls[1]);
        if(isHit) {
//                  碰撞之后将一个小球对象置空，另一个小球的半径加10(或者另一个小球的半径)
            allBalls[1].setR(allBalls[1].getR()+allBalls[0].getR());
            allBalls[0] = null;
```

```
            }
        }
    }
```

a. twoBallsHit方法中：

```
        boolean isHit = false;
        if(ballOne !=null && ballTwo != null) {
//              定义一个标记位：未相撞
            int bxot = ballOne.getX() - ballTwo.getX();
            int byot = ballOne.getY() - ballTwo.getY();
            int rot = ballOne.getR() + ballTwo.getR();
//              判断小球碰撞
            double z = Math.sqrt(bxot*bxot+byot*byot);
            if(z <= rot) {
                isHit = true;//此时小球相撞
            }
        }
```

二、案例知识点总结

1. 面向对象思想：将具有相同特征和行为的事物抽象为一个类，在类中通过属性来定义特征、通过方法来定义行为；通过封装将属性私有化，再提供公开的赋值和取值方法，以便于在类的外部来使用属性
2. static关键字：类的、静态的；由它修饰的属性和方法可以通过类名直接调用；不能在静态方法中直接调用非静态的属性或方法、需要通过对象来调用才可。
3. final关键字：最终的、不可改变的；
   a. final修饰类：

```
public final class FinalAnimal {

    public int name;
    public int age;

    public void eat() {
    }

    public void slepp() {
    }
}
//此时会抛出异常提示：The type Dog cannot subclass the final class FinalAnimal
public class Dog extends FinalAnimal{

}
```

final类不能被继承，因为final类的成员变量或方法都没有机会被覆盖，默认都是final的；在设计一个类的时候，如果这个类不需要有子类，类的实现细节不允许被改变并且确定这个类不会再被扩展，那么就设计成final类。现阶段Java原生类库中最常用的final类是String

   b. final修饰方法：

如果一个类不允许其子类重写覆盖其自身的某个方法，则可以把这个方法声明为final方法：

第一个原因：锁定方法、防止任何继承的子类修改它的意义和实现

第二个原因：高效

c. final修饰变量（常量）：

用final修饰的成员变量表示常量！ 只能被赋值一次，赋值之后数值无法改变！

也即final修改的常量，只能声明的同时进行初始化！

- 可以修改静态变量：final static varName = val;

可以修改成员变量（类）：final varName=val;

可以修改局部变量（方法）：final varName=val;

# 案例8：用键盘控制小球

2022年7月9日 9:35

一、案例需求

1. 需求：窗口--画布；在画布上绘制一个小球，然后实现玩家按键盘上的上下左右键的时候，小球将向箭头指向的方法移动，按键松开之后小球停止移动
2. 分析：
   a. 创建窗口、画布、绘制小球
   b. 通过上下左右方向键控制小球的移动

二、具体实现步骤以及代码

1. 创建窗口、画布、绘制小球，参考之前案例实现窗口创建、画布创建、小球绘制即可：
   a. 窗口类 ControlBallFrame
   b. 画布类 ControlBallPanel
2. 实现通过键盘的方向键控制小球：需要让程序想办法获得键盘的输入信息，Java类库提供了一个键盘监听器接口：KeyListener，需要实现此接口以及其中的几个重要方法：
   public void keyTyped(KeyEvent e); 当键入某个键的时候调用此方法
   public void keyPressed(KeyEvent e);当按下某个键的时候调用此方法
   public void keyReleased(KeyEvent e);当释放某个键的时候调用此方法
   让ControlBallPanel添加一个实现类public class ControlBallPanel extends JPanel implements KeyListener，按照提示来实现上述的三个方法：
   a. 第一个方法keyPressed
      实现按下方向键的时候小球向对应的方向移动，也即获得某个方向修改小球的坐标：

```
//     KeyEvent类提供一系列的键盘输入事件常量，将此时的键盘输入与常量进行比较
//     e表示玩家输入的键盘操作
@Override
public void keyPressed(KeyEvent e) {
    if(e.getKeyCode() == KeyEvent.VK_UP) {
        y--;
        repaint();
    }else if(e.getKeyCode() == KeyEvent.VK_DOWN) {
        y++;
        repaint();
    }else if(e.getKeyCode() == KeyEvent.VK_LEFT) {
        x--;
        repaint();
    }else if(e.getKeyCode() == KeyEvent.VK_RIGHT) {
        x++;
        repaint();
    }
```

}

b. 此时运行发现键盘无法控制让小球动起来，小球空有移动的能力，但是不会使用；需要通过注册监听器让小球可以被键盘控制在窗口上运动：

```
//              打通小球的任督二脉，为控件注册监听
            frame.addKeyListener(panel);
```

二、案例知识点总结

1. 添加键盘事件监听的步骤：
   a. 实现对应的键盘监听器接口：KeyListener
   b. 重写接口中的方法：获取键盘按下的事件
   c. 将对象的键盘监听器接口注册到窗口上
2. 练习：获取键盘输入的数字，然后让小球移动相应的步数

# 案例9：用鼠标控制小球

2022年7月9日 14:33

一、案例需求

1. 需求：创建一个图形化窗体，显示；创建画布，在上面绘制一个小球，并将画布添加到窗体上，然后实现当玩家在窗体上按下鼠标时可以拖着小球移动
2. 分析：
   a. 创建窗体、画布、并绘制一个小球
   b. 通过鼠标来控制小球的移动，鼠标可以拖动小球来回移动

二、具体实现步骤以及代码

1. 创建窗体、画布，完成小球的绘制；参考之前的案例的实现即可
2. 实现鼠标可以拖动小球来回移动，事件监听器提供了三个鼠标的监听器，其中最贴合需求是MouseMotionListener监听器，实现其中的两个方法：
   public void mouseDragged(MouseEvent e);//按压然后拖拽
   public void mouseMoved(MouseEvent e);//跟随鼠标指针移动
   在MouseCtrlPanel中完成鼠标控制小球移动的代码：

```
//    让小球跟随鼠标的按压拖拽移动而移动，
//    通过MouseEvent来获取到当前鼠标在画布上的坐标，
//    并将此坐标赋值给小球
    @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();
    }
```

三、案例知识点：

1. 鼠标动作的事件监听器：移动（按压并拖拽、指针）鼠标，小球随之移动
2. 练习：MouseListener