实验课程名称：__软件工程基础实验_____

实验项目名称	结对编程			实验成绩	
实 验 者	蒋如星	专业班级	软件 1704	组　　别	
同 组 者	黄锐			实验日期	2019 年 4 月 24 日

第一部分：实验预习报告（包括实验目的、意义，实验基本原理与方法，主要仪器设备及耗材，实验方案与技术路线等）

一、实验目的

1）体验敏捷开发中的两人合作。

2）进一步提高个人编程技巧与实践。

二、实验内容及要求

1）选择一个程序实例，练习结对编程(pairprogramming)实践；

2）要求学生两人一组，自由组合。每组使用一台计算机，二人共同编码，完成实验要求。

3）要求在结对编程工作期间，两人的角色至少切换 4 次；

4）编程语言不限，版本不限。建议使用 Python 或 JAVA 进行编程。

三、示例程序问题描述

生命游戏

生命游戏是英国数学家约翰·何顿·康威在 1970 年发明的细胞自动机，它包括一个二维矩形世界，这个世界中的每个方格居住着一个活着的或死亡的细胞。一个细胞在下一个时刻生死取决于相邻八个方格中活着的或死了的细胞的数量。如果相邻方格活着的细胞数量过多，这个细胞会因为资源匮乏而在下一个时刻死去；相反，如果周围活细胞过少，这个细胞会因太孤单而死去。

游戏在一个类似于围棋棋盘一样的，可以无限延伸的二维方格网中进行。例如，设想每个方格中都可放置一个生命细胞，生命细胞只有两种状态：“生”或“死”。图中，用黑色的方格表示该细胞为“死”，其它颜色表示该细胞为“生”。游戏开始时，每个细胞可以随机地（或给定地）被设定为“生”或“死”之一的某个状态，然后，再根据如下生存定律计算下一代每个细胞的状态：

每个细胞的状态由该细胞及周围 8 个细胞上一次的状态所决定；

如果一个细胞周围有 3 个细胞为生，则该细胞为生，即该细胞若原先为死则转为生，若原先为生则保持不变；

如果一个细胞周围有 2 个细胞为生，则该细胞的生死状态保持不变；

在其它情况下，该细胞为死，即该细胞若原先为生则转为死，若原先为死则保持不变。

第二部分：实验过程记录（可加页）（包括实验原始数据记录，实验现象记录，实验过程发现的问题等）

1.主要算法

1) 数据的储存结构

```
private boolean[][] table = new boolean[SIZE][SIZE];//格子的邻居数目
private int[][] neighbors = new int[SIZE][SIZE];
```

使用二维数组来存储每一个细胞的状态

2) 界面的显示

```
private final int SIZE = 30;//二维游戏世界的大小,共SIZE*SIZE个格子
private final int CELL_Size =20;//每个格式的边长，Java 坐标系单位。
g.fillRect(x * CELL_Size, y * CELL_Size, CELL_Size-1 , CELL_Size-1 );  //位置+大小。全填充，无间隔
```

可以想象：x,y为直角坐标的中点，也就是正方形的额左上角，以该点为基准，向右下方来确定每个细胞的位置。并且，每一个细胞为一个小方格

3) 获取细胞周围 8个细胞上一次的状态

```
public void getNeighbors() {
   for (int r = 0; r < SIZE; r++){//row
      for (int c = 0; c < SIZE; c++){//col
          if(r-1 >= 0 && c-1 >= 0   && table[r-1][c1] )neighbors[r][c]++;
          if(r-1 >= 0 && table[r-1][c])            neighbors[r][c]++;
          if(r-1 >= 0 && c+1 < SIZE && table[r-1][c+1])neighbors[r][c]++;
          if(c-1 >= 0   && table[r][c-1]) neighbors[r][c]++;
          if(c+1 < SIZE && table[r][c+1]) neighbors[r][c]++;
          if(r+1 < SIZE && table[r+1][c]) neighbors[r][c]++;
          if(r+1 < SIZE && c+1 < SIZE && table[r+1][c+1])
          if(r+1 < SIZE && c-1 >=0 && table[r+1][c-1]) neighbors[r][c]++;
   }}     }
```

以该细胞为基准，检查周围8铬细胞（1.是否存在2.是否为活细胞）

4) 确定下一阶段生命界面活细胞分布情况

```
public void nextWorld() {
    for (int r = 0; r < SIZE; r++){//row
    for (int c = 0; c < SIZE; c++){//col
           if (neighbors[r][c] == 3){
               table[r][c] = true;
               }//if (neighbors[r][c] == 2) 不改变table[r][c]。
           if (neighbors[r][c] < 2)
               table[r][c] = false;
           if (neighbors[r][c] >= 4)
               table[r][c] = false;
           neighbors[r][c] = 0;      //格子周围活细胞数量置零
       }      }    }
```

* nextWorld()，世代交替。
* 生命游戏的核心是计算出下一代的 table，产生新一代的二维世界。
* 按照每一个 neighbors 元素

2.Applet 小程序实现界面

1) 初始状态

```
public void init()  {
  animator = new Thread(this);   //新建线程
  delay = 100;
  running = false;
  //setBackground(Color.yellow);
  setBackground(new Color(199,237,204));
  addMouseListener(this);
  addMouseMotionListener(this);
  addKeyListener(this);
  }
```

增加鼠标事件监听和空格事件监听，可以实现

a) 用鼠标对死细胞点击，复活
b) 点击鼠标并进行拖拽，可以批量复活细胞
c) 按压空格实现游戏的继续和暂停

2) 更新界面状态

```
public void paint(Graphics g) {
    update(g);}
public void update (Graphics g) {
   for (int x = 0; x < SIZE; x++)
    for (int y = 0; y < SIZE; y++) {
      g.setColor(table[x][y]?cell:space);
      g.fillRect(x * CELL_Size, y * CELL_Size, CELL_Size-1 ,CELL_Size-1 );  //
     位置+大小。全填充，无间隔}}
```

3) 线程运行run()

```
public void run() {
   long tm = System.currentTimeMillis();
   while (Thread.currentThread() == animator) {  //如果线程就是animator
       if (running == true) {
           getNeighbors();
           nextWorld();
           repaint();
       }
       try {
           tm += delay;
           Thread.sleep(Math.max(0, tm - System.currentTimeMillis()));
       } catch (InterruptedException e) {
           break;
       } }
} // run
```

调用 repaint(),刷新 applet 小程序，先清空程序空间，在调用 paint().

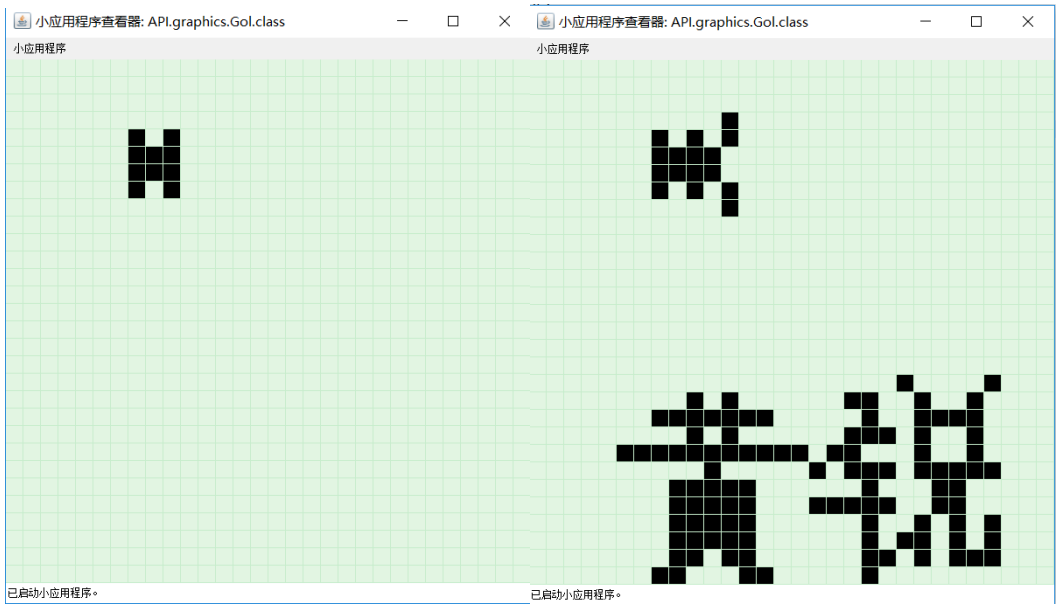
4) 事件

```
public void mousePressed(MouseEvent e){       //点击鼠标事件
    int cellX = e.getX()/CELL_Size;
    int cellY = e.getY()/CELL_Size;   //获取坐标
    table[cellX][cellY] = !e.isControlDown();
    //按ctrl时，活细胞变死(黑变白，1变0);无ctrl时，死细胞变活(白变黑，0变1)
    repaint();//更新界面 }
public void mouseDragged(MouseEvent e){    //鼠标拉拽事件
    this.mousePressed(e);
}
public void keyPressed(KeyEvent e){      //按压空格事件
    if(e.getKeyChar()==' '){
        running = !running;
        repaint();
    }}
public void keyReleased(KeyEvent e){}
}
```

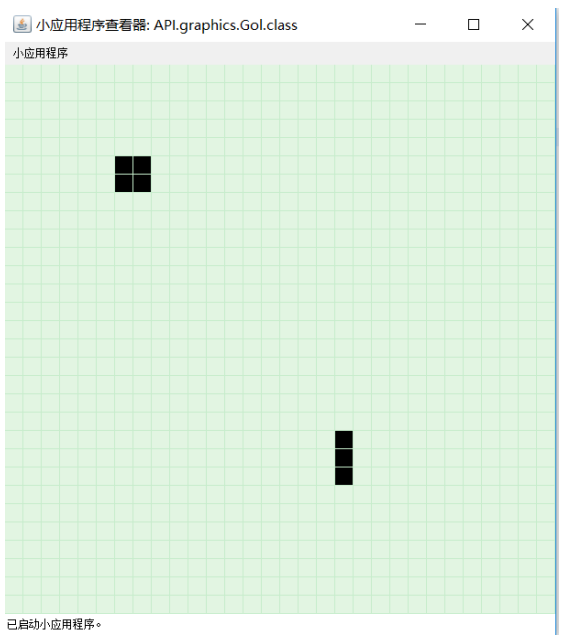
第三部分　结果与讨论（可加页）

一、实验结果分析（包括数据处理、实验现象分析、影响因素讨论、综合分析和结论等）

1)　初始状态人为确定，使用鼠标设计初始状态图案（鼠标事件）

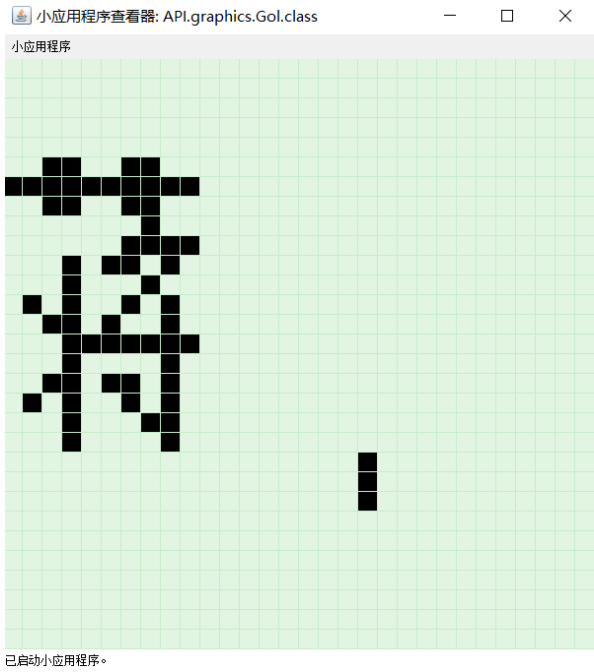


2)　有 space 空格键来执行运行和停止



运行结果

3)　可以在运行过程中更改图形状态：使用空格暂停，相当于初始状态，此时可以使用鼠标更改状态



4) 可以使用鼠标拖拽实现批量更改，并且可以实现更改颜色

二、实验小结及体会

本次实验程序的代码是由黄锐和我共同编写完成的，在以往的实验中，所有的工作都是由一个人独立进行的，通过结对编程，我们在过程中互相发现并改正了自己的不足之处，成功编写了 lifegame 程序，在之后的实验中，也希望能够增强我们的合作水平。

成绩评定表：

序号	评分项目	满分	实得分
1	**实验报告格式规范**	**2**	
2	**实验报告过程清晰，内容详实**	**4**	
3	**实验报告结果正确性**	**2**	
4	**实验分析与总结详尽**	**2**	
	总得分	**10**	