



# 面向对象的软件构造导论

## 实验五：Swing和多线程

2023春

哈尔滨工业大学（深圳）



# 本学期实验总体安排

实验项目	一	二	三	四	五	六
学时数	2	2	2	2	4 (2+2)	4
实验内容	飞机大战 功能分析	单例模式 工厂模式	Junit 单元测试	策略模式 数据访问 对象模式	Swing 多线程	模板模式 观察者模式
分数	4	6	4	6	6	14
提交内容	UML类图、 代码	UML类图、 代码	单元测试 代码 测试报告	UML类图、 代码	代码	项目代码、 实验报告、 展示视频

实验课程共**16**个学时，**6**个实验项目，总成绩为**40**分。



# 目录

---

01

实验目的

---

02

实验任务

---

03

实验步骤

---

04

作业提交

---



# 实验目的

---

- 熟悉Java图形界面程序设计的基本方法，掌握Java Swing中的容器、常用组件和布局管理器的使用；
- 理解Java事件处理机制，掌握事件处理机制的基本用法；
- 理解Java多线程的概念和生命周期，掌握多线程的实现方法。



# 实验任务

---

1. 使用Java Swing类库完成以下界面：
  - ✓ 游戏难度选择、音效开关界面
  - ✓ 得分排行榜界面
  
2. 使用多线程完成以下功能：
  - ✓ 使用Runnable接口实现多线程，完善火力道具
  - ✓ 继承Thread类实现多线程，完成游戏音效控制

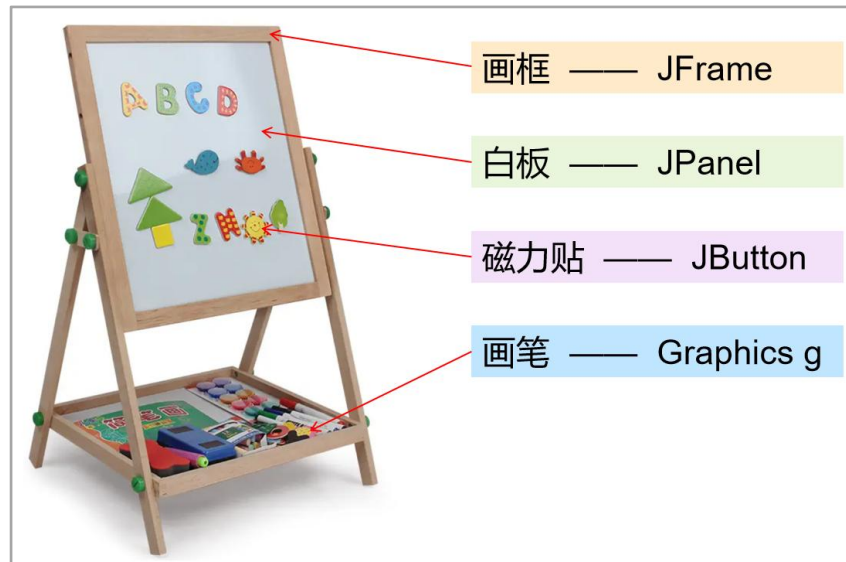
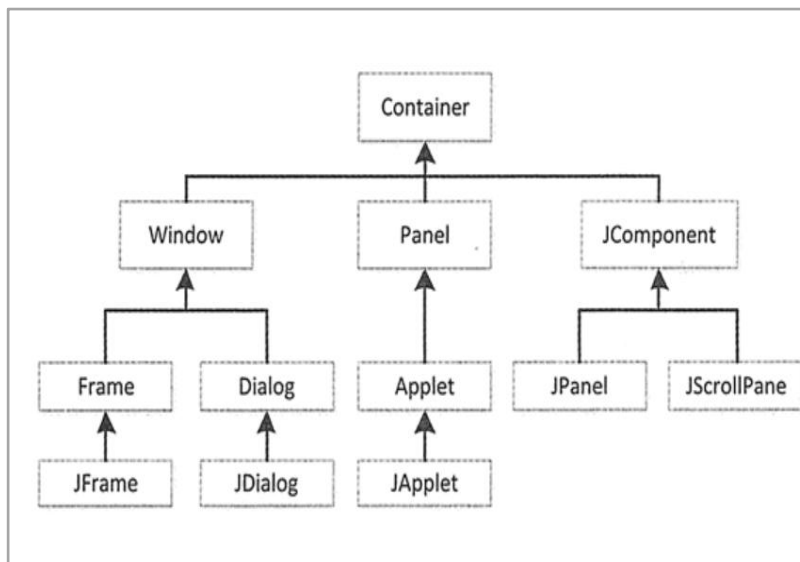


# 实验步骤

## 1 Java Swing

### Swing

- 是 Java 为图形界面（GUI）应用开发提供的一组工具包；
- 包含构建图形界面的各种组件，可划分为容器组件和非容器组件；



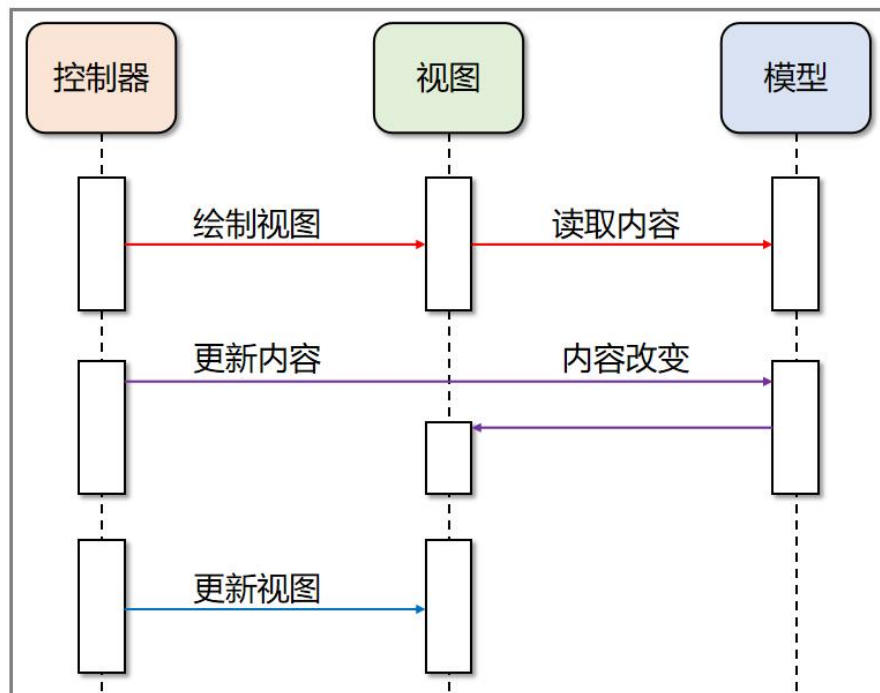


# 实验步骤

## 1 Java Swing

### Swing

- 采用MVC模式设计，实现GUI组件的显示逻辑和数据逻辑的分离；
- IntelliJ IDEA提供的Swing GUI Designer可方便进行图形界面编程。

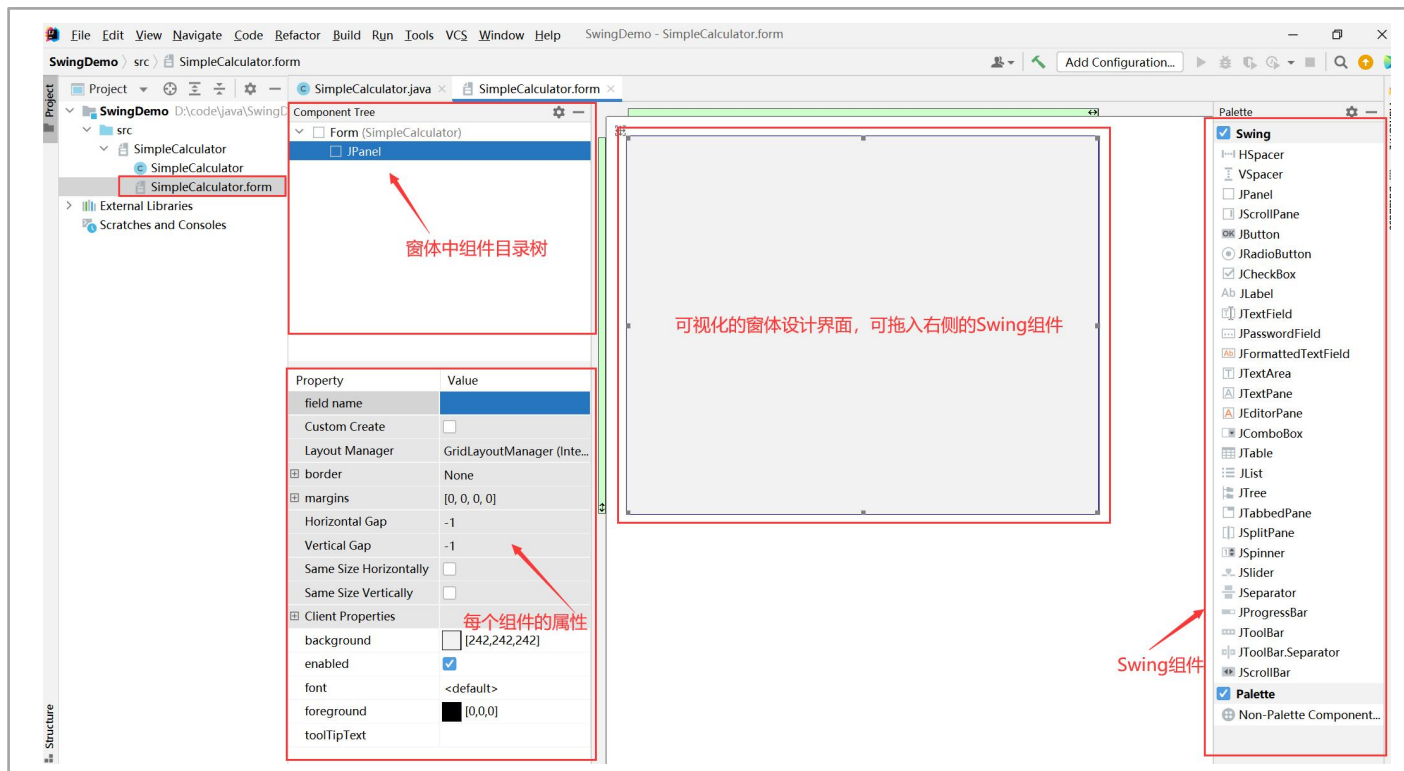




# 实验步骤

## 1 Java Swing - Swing GUI Designer

- ①选中项目src目录，右键新建一个**GUI Form**，命名为：SimpleCalculator。
- ②自动生成两个文件：**SimpleCalculator.form** 用于图形界面设计；  
**SimpleCalculator.java** 用于操作组件对象和运行。

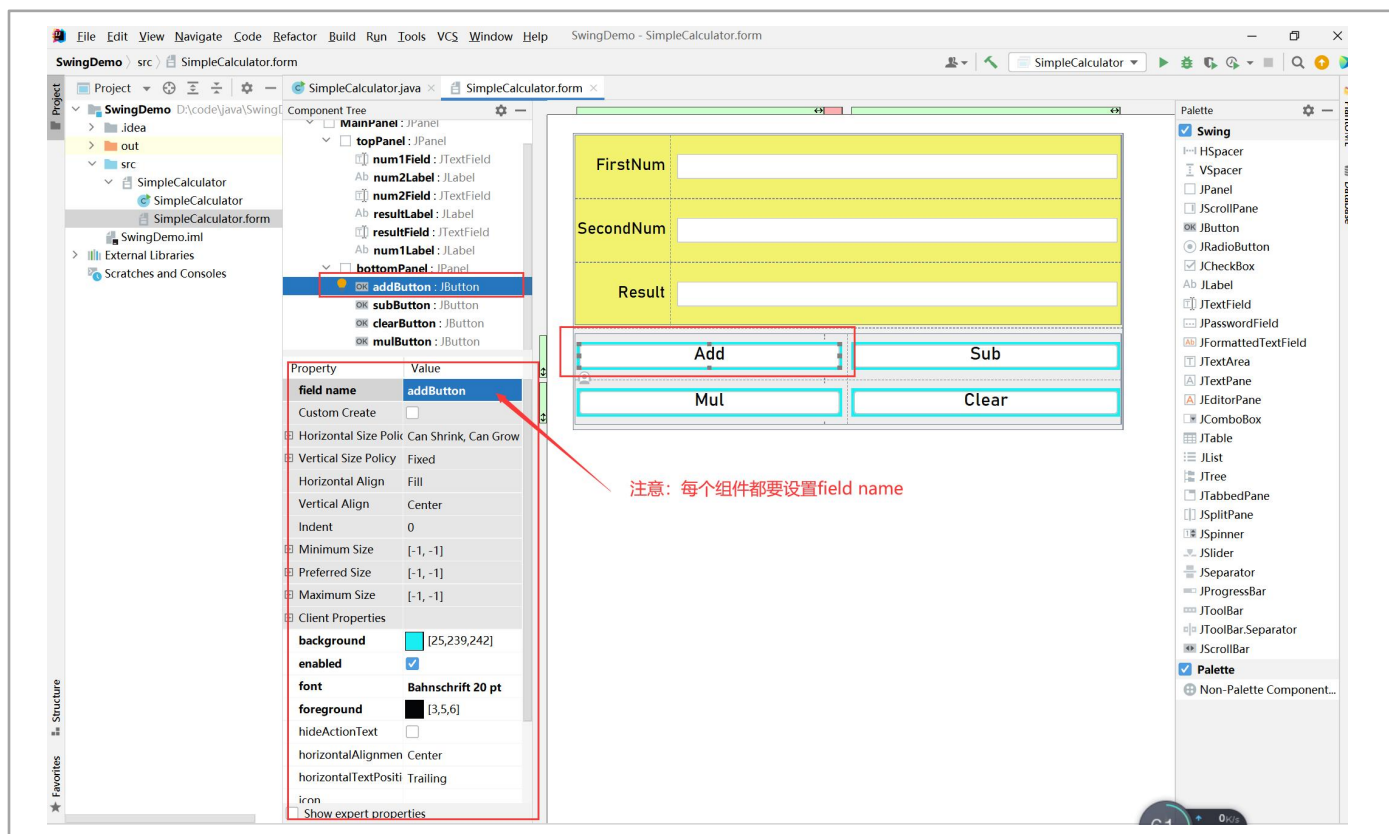




# 实验步骤

## 1 Java Swing - Swing GUI Designer

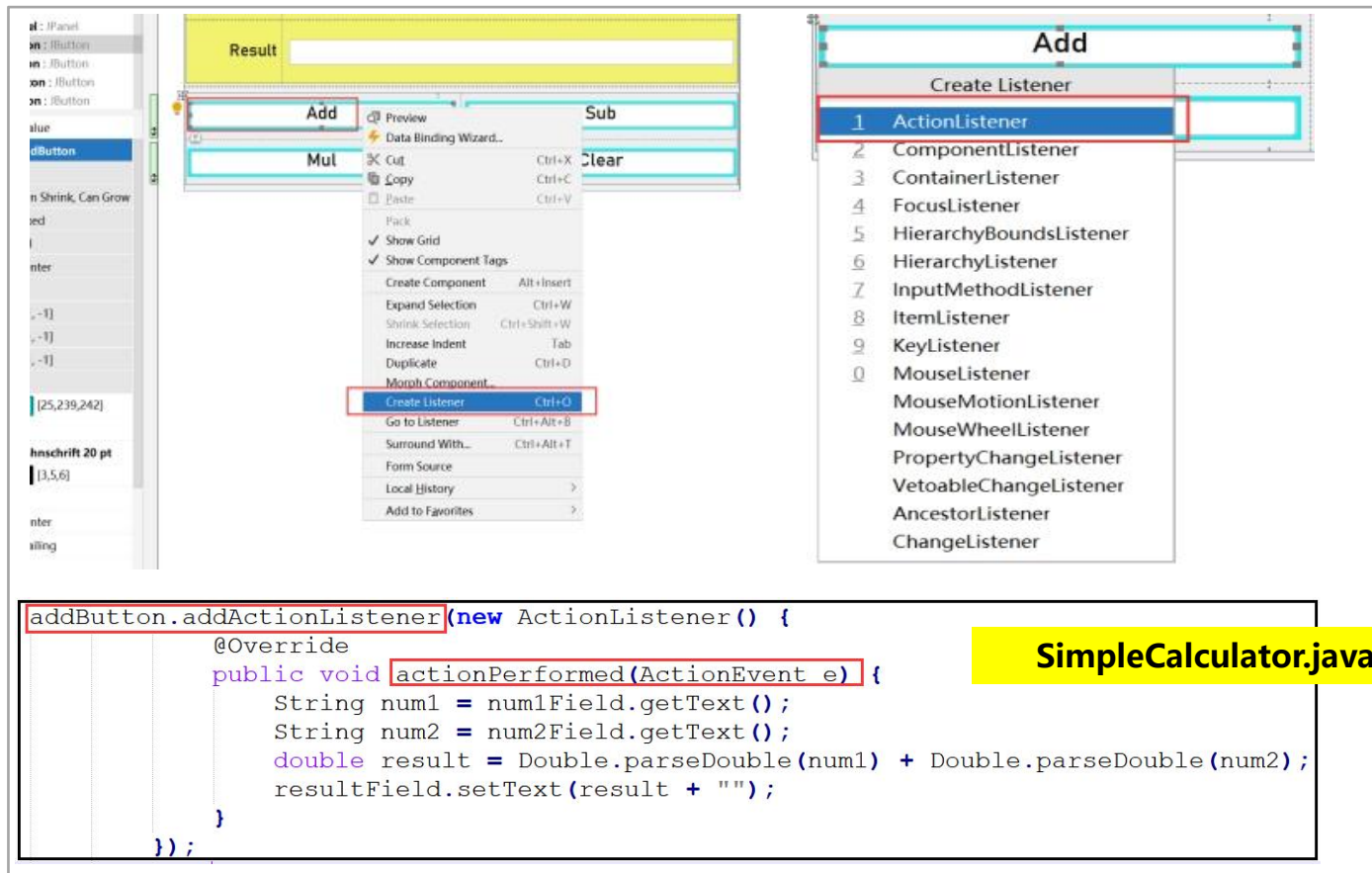
③ 在窗口设计界面中拖入所需要的Swing组件并合理布局。**注意每个组件都要设置field name。**右键选择Preview选项可查看窗体设计的效果。



# 实验步骤

## 1 Java Swing - Swing GUI Designer

④ 添加按钮事件，右键Add按钮，选择Create Listener选项，为其添加 **ActionListener**。在actionPerformed()函数体中添加事件处理代码。



The screenshot illustrates the process of adding an event listener to a button in the Swing GUI Designer. The 'Add' button is selected, and the context menu is open, with 'Create Listener' highlighted. The 'Create Listener' dialog is shown, listing various listener interfaces, with 'ActionListener' selected. Below the dialog, the Java code for the ActionListener is shown, with the 'addButton.addActionListener' line highlighted.

```
addButton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        String num1 = num1Field.getText();  
        String num2 = num2Field.getText();  
        double result = Double.parseDouble(num1) + Double.parseDouble(num2);  
        resultField.setText(result + "");  
    }  
});
```

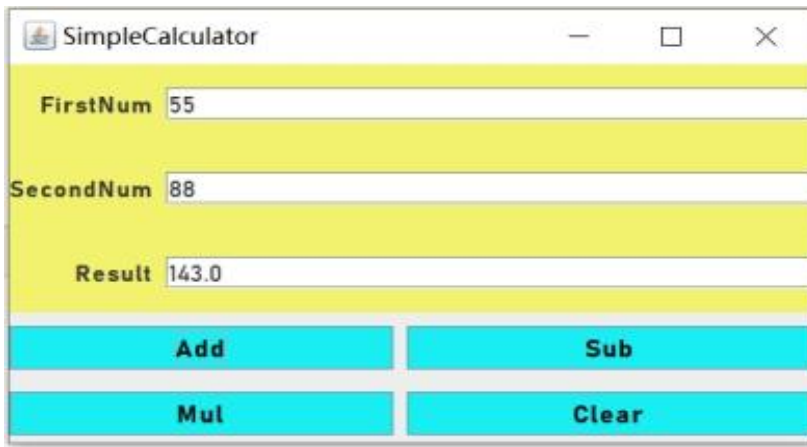
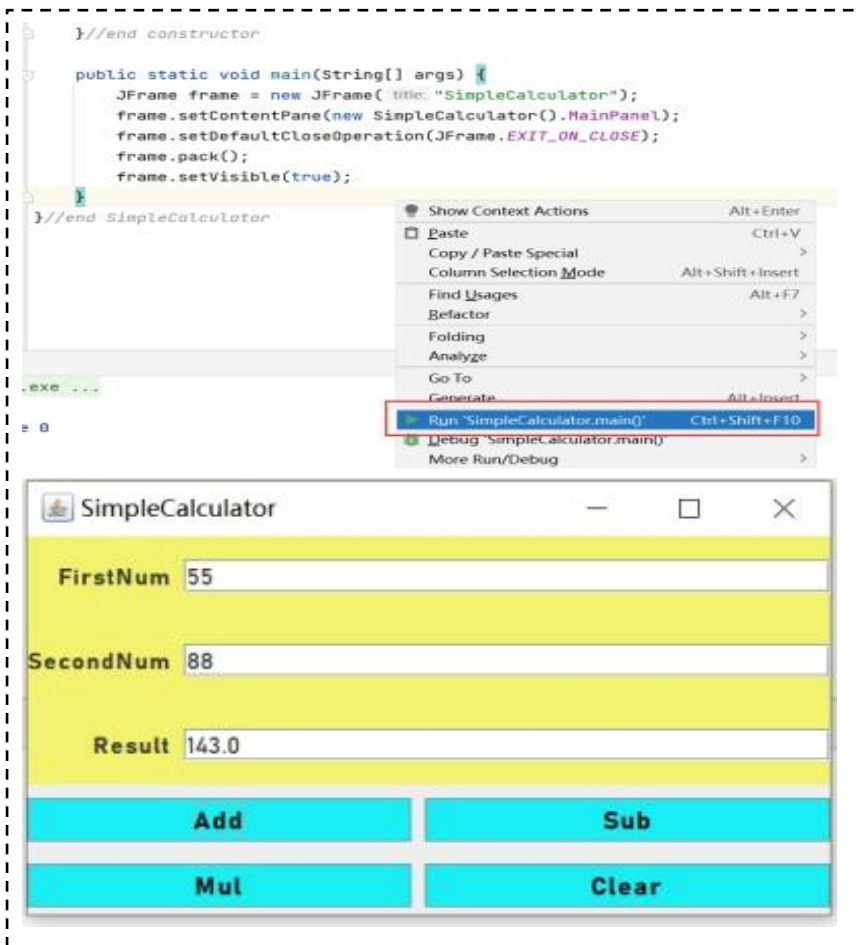
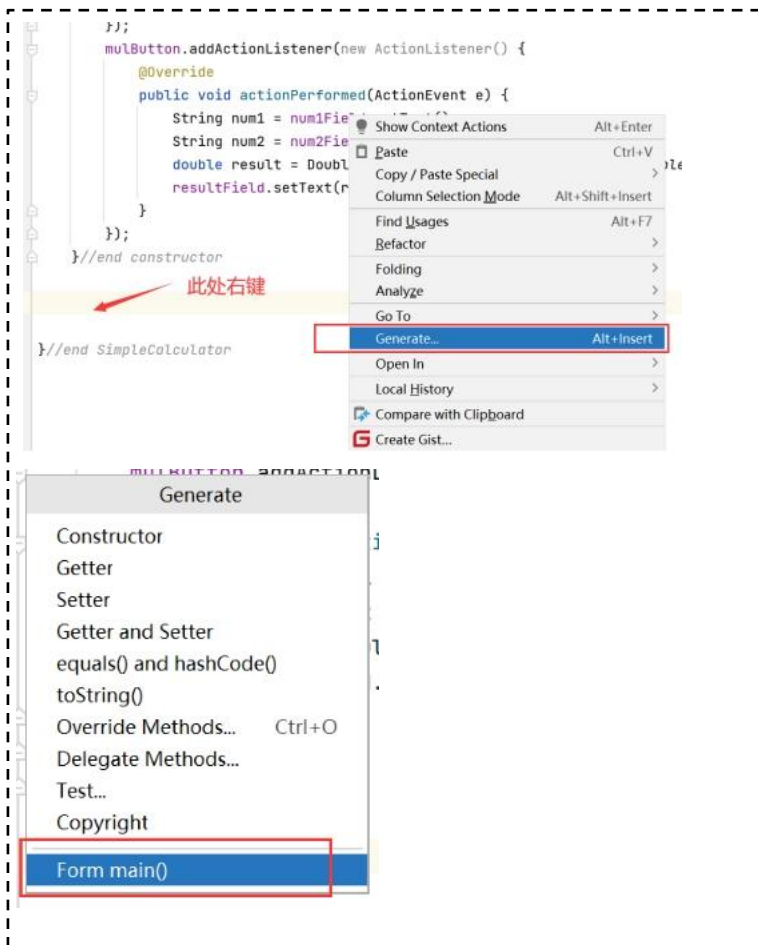
**SimpleCalculator.java**



# 实验步骤

## 1 Java Swing - Swing GUI Designer

⑤ 添加Form Main函数，并运行程序。



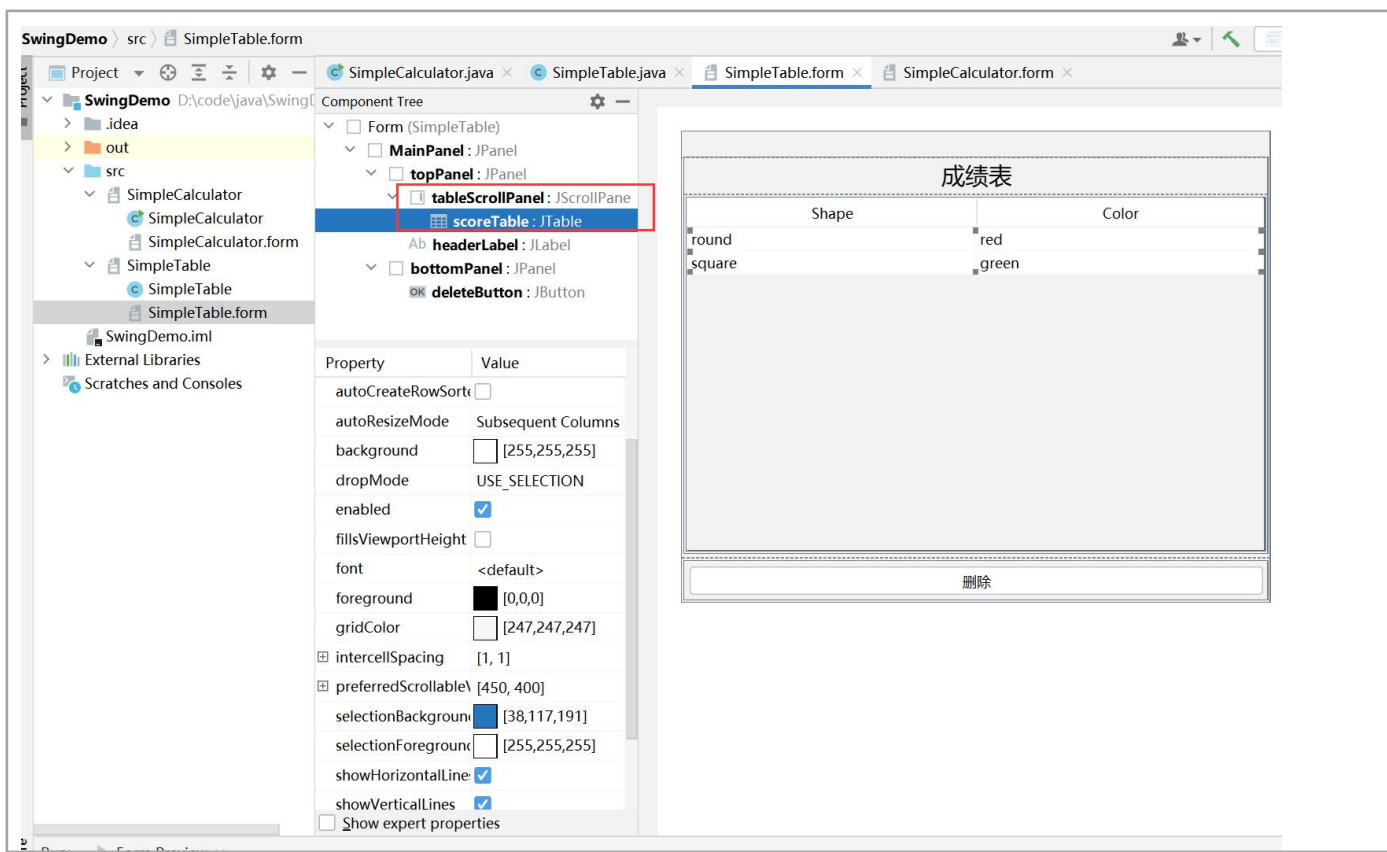


# 实验步骤

## 1 Java Swing - JTable

**JTable** 是将数据以表格的形式显示给用户看的一种组件，它包括行和列。

① 添加所需组件 (**JScrollPane** 和 **JTable**) 并合理布局。





# 实验步骤

## 1 Java Swing - JTable

② 在构造函数中创建DefaultTableModel对象，装载数据。

```
public SimpleTable() {  
  
    String[] columnName = {"学号", "姓名", "成绩"};  
    String[][] tableData = {{ "001", "Lily", "78"}, {"002", "Jane", "89"}, {"003", "Alex", "67"},  
                             {"004", "Macy", "83"}, {"005", "Nancy", "66"}, {"006", "John", "99"} };  
  
    // 表格模型  
    DefaultTableModel model = new DefaultTableModel(tableData, columnName) {  
        @Override  
        public boolean isCellEditable(int row, int col) {  
            return false;  
        }  
    };  
  
    // 从表格模型那里获取数据  
    scoreTable.setModel(model);  
    tableScrollPane.setViewportViewView(scoreTable);  
  
}
```

JTable表格与数据通过TableModel分离，JTable并不存储自己的数据，而是从表格模型那里获取它的数据。



# 实验步骤

## 1 Java Swing - JTable

③ 添加“删除”按钮事件和Form Main函数。

```
deleteButton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        int row = scoreTable.getSelectedRow();  
        System.out.println(row);  
        if (row != -1) {  
            model.removeRow(row);  
        }  
    }  
});
```

运行程序：

学号	姓名	成绩
001	Lily	78
002	Jane	89
003	Alex	67
004	Macy	83
005	Nancy	66
006	John	99

学号	姓名	成绩
001	Lily	78
002	Jane	89
003	Alex	67
005	Nancy	66
006	John	99





# 实验步骤

## 1 Java Swing - CardLayout

**CardLayout** 布局管理器以时间而非空间来管理它里面的组件，能够让多个组件共享同一个显示空间。

① 初始化卡片布局对象**cardLayout** 以及使用它的容器**cardPanel**。

```
public class CardLayoutDemo {  
  
    static final CardLayout cardLayout = new CardLayout( hgap: 0, vgap: 0);  
    static final JPanel cardPanel = new JPanel(cardLayout);  
  
    public static void main(String[] args) {  
  
        JFrame frame = new JFrame( title: "CardLyaout Demo");  
        frame.setSize( width: 800, height: 1024);  
        frame.setResizable(false);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
        frame.add(cardPanel);  
  
        StartMenu start = new StartMenu();  
        cardPanel.add(start.getMainPanel());  
        frame.setVisible(true);  
    }  
}
```



# 实验步骤

## 1 Java Swing - CardLayout

- ② 新建窗体JFrame，并将容器cardPanel加入其中。新建StartMenu页面，加入容器cardPanel并显示。

```
public class CardLayoutDemo {  
  
    static final CardLayout cardLayout = new CardLayout( hgap: 0, vgap: 0);  
    static final JPanel cardPanel = new JPanel(cardLayout);  
  
    public static void main(String[] args) {  
  
        JFrame frame = new JFrame( title: "CardLayout Demo");  
        frame.setSize( width: 800, height: 1024);  
        frame.setResizable(false);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
        frame.add(cardPanel);  
  
        StartMenu start = new StartMenu();  
        cardPanel.add(start.getMainPanel());  
        frame.setVisible(true);  
    }  
}
```





# 实验步骤

## 1 Java Swing - CardLayout

- ③ 为StartMenu的两个按钮添加事件，将要跳转的页面加入容器cardPanel，并使用cardLayout实现页面切换。

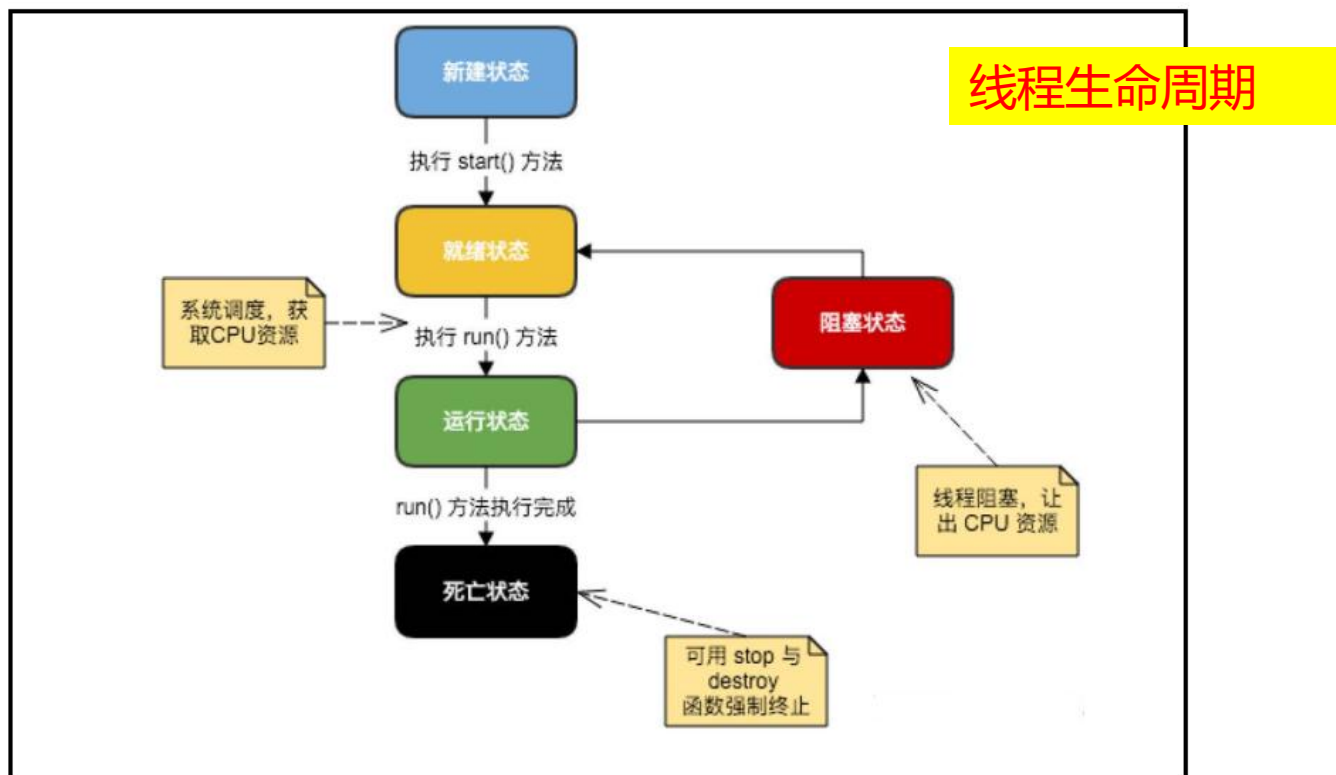
```
public StartMenu() {  
    simpleCalculatorButton.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            CardLayoutDemo.cardPanel.add(new SimpleCalculator().getMainPanel());  
            CardLayoutDemo.cardLayout.last(CardLayoutDemo.cardPanel);  
        }  
    });  
    simpleTableButton.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            CardLayoutDemo.cardPanel.add(new SimpleTable().getMainPanel());  
            CardLayoutDemo.cardLayout.last(CardLayoutDemo.cardPanel);  
        }  
    });  
}
```



# 实验步骤

## 2 Java 多线程编程

飞机大战游戏中，**火力道具生效**、**音效多处控制**等功能需要用多线程来完成。Java 给多线程编程提供了内置的支持。线程是一个动态执行的过程，它也有一个从产生到死亡的过程。





# 实验步骤

## 2

## Java 多线程编程

- ① 使用 Runnable 接口实现多线程, 由于Runnable 是函数式接口, 可使用 lambda表达式简写。

```
public class RunnableTest {  
    public static void main(String[] args) {  
  
        Runnable r = () -> {  
            try {  
                for (int i = 0; i < 3; i++) {  
                    System.out.println("【" + Thread.currentThread().getName() +  
                        "】线程执行, 当前的循环次数: " + i);  
  
                    Thread.sleep(2000);  
                }  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        };  
  
        // 启动线程  
        new Thread(r, "线程1").start();  
        new Thread(r, "线程2").start();  
    }  
}
```

【线程 1】线程执行, 当前的循环次数: 0

【线程 2】线程执行, 当前的循环次数: 0

【线程 1】线程执行, 当前的循环次数: 1

【线程 2】线程执行, 当前的循环次数: 1

【线程 1】线程执行, 当前的循环次数: 2

【线程 2】线程执行, 当前的循环次数: 2



# 实验步骤

## 2

## Java 多线程编程

- ② 继承 Thread 类实现多线程，继承类必须重写 run() 方法，该方法是新线程的入口点。

本实验已提供 **MusicThread** 类，该类继承 Thread 类，重写了 run() 方法，用于启动音频播放。

```
@Override
public void run() {
    InputStream stream = new ByteArrayInputStream(samples);
    play(stream);
}
```



# 实验步骤

## 2

## Java 多线程编程

- ② 继承 Thread 类实现多线程，继承类必须重写 run() 方法，该方法是新线程的入口点。

使用方法：

```
public class ThreadTest {  
    public static void main(String[] args) {  
        new MusicThread("src/bgm.wav").start();  
    }  
}
```

**注意：**

在飞机大战游戏中，还需实现循环播放、停止播放音频的功能。



# 本次实验的目标（1）

- ✓ 游戏开始显示**难度选择**和**音效设置**界面，根据玩家选择显示相应难度的**游戏地图**；
- ✓ 游戏结束后显示**得分排行榜**界面，要求可记录玩家该局得分，并可删除玩家历史得分；

**注意：**

本实验无需  
细化三种不  
同游戏难度，  
实验六进一  
步完善即可。

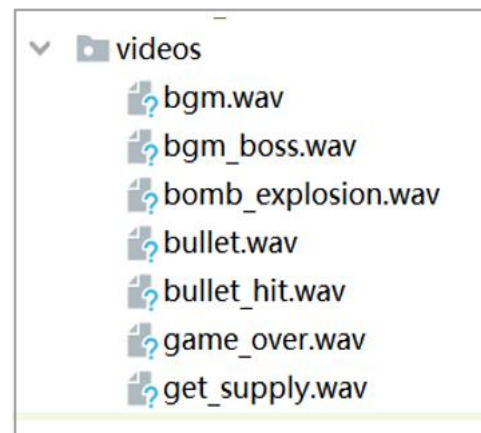
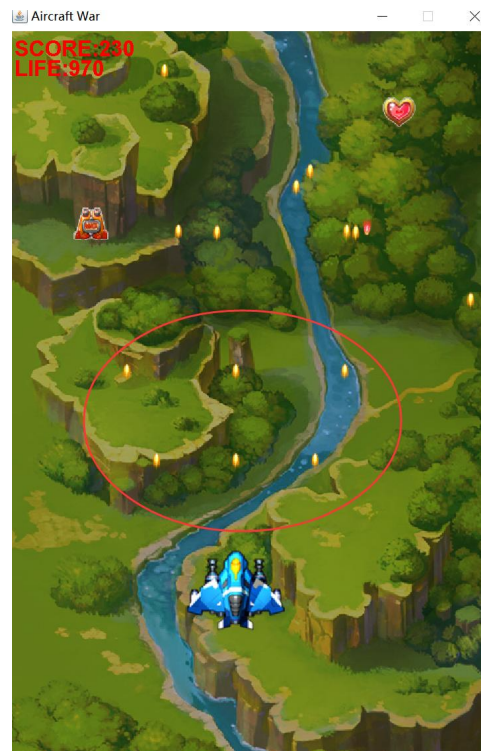


界面与范例接近即可。



## 本次实验的目标（2）

- ✓ 火力道具生效时，英雄机由直射切换为散射弹道并持续一段时间，结束后恢复直射状态；
- ✓ 若音效开启，游戏中循环播放游戏背景音乐，游戏结束后停止播放；
- ✓ 子弹击中敌机、炸弹爆炸、道具生效、游戏结束时会有相应的音效。
- ✓ Boss敌机出场时循环播放其背景音乐，坠毁或游戏结束后停止播放。





# 作业提交

---

- 提交内容

- ① 整个项目压缩包

- 截止时间

实验课后一周内提交至HITsz Grader 作业提交平台，具体截止日期参考平台发布。

登录网址：： <http://grader.tery.top:8000/#/login>





---

**同学们  
请开始实验吧！**