广州大学学生实验报告

开课实验室：计算机科学与工程实验（电子楼 418A）　　2023 年 11 月 15 日

学院	计算机科学与网络工程学院	年级、专业、班	软件 221	姓名	何骐光	学号	32206300076
实验课程名称	JAVA 语言实验					成绩	
实验项目名称	Java 语言图形图像和多线程运用					指导老师	樊志平

一、实验目的

- 熟悉 Java 图形界面的基本设计。
- 熟悉 Java 界面的菜单使用方法。
- 熟悉 Java 的多线程应用程序开发方法。

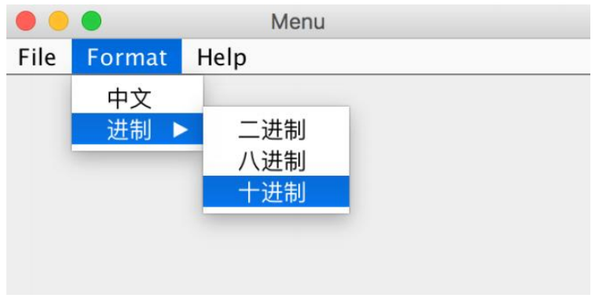
二、实验任务

实验任务 1

编写 Java 应用程序，实现以下登陆界面（需注意密码框输入的内容不显示明文）：

实验任务 2

编写 Java 应用程序，实现以下界面：



实验任务 3

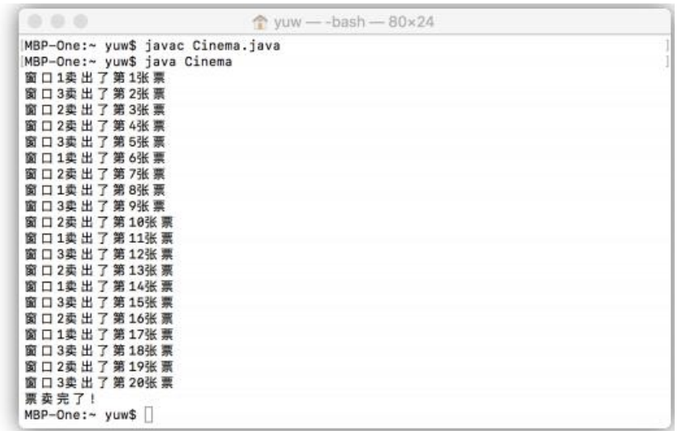
编写一个 Java 多线程应用程序，完成三个售票窗口同时出售 20 张票。具体要求如下：

- 票数要使用同一个静态值；
- 为保证不会出现卖出同一个票数，要 java 多线程同步锁。

设计思路：

- 创建一个站台类 Station，继承 Thread，重写 run 方法，在 run 方法里面执行售票操作。售票要使用同步锁：即有一个站台卖这张票时，其他站台要等这张票卖完。
- 创建主方法调用类。

运行效果参考下图：



三、实验内容

实验任务 1

```java
//ui类
import javax.swing.*;
import java.awt.*;
public class ui extends JFrame {
    public ui() {
        // 设置界面大小
        this.setSize(450, 250);
        // 设置登录界面的Title
        this.setTitle("登录");

        // 设置界面居中
        this.setLocationRelativeTo(null);

        // 设置关闭模式
        this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        // 创建用户名和密码的标签，用于管理文本
        JLabel usernameLabel = new JLabel("用户名:");
```

```
        JLabel passwordLabel = new JLabel("密码:");

        // 创建用户名和密码的文本框
        JTextField usernameField = new JTextField(20);//明文显示，宽度设置为 20 个字符
        JPasswordField passwordField = new JPasswordField(20);//密文显示，宽度设置为
20 个字符

        // 创建登录按钮
        JButton loginButton = new JButton("登录");
        loginButton.setAlignmentX(JButton.CENTER_ALIGNMENT);//将按钮的水平对齐方式设
置为居中对齐

        // 创建布局管理器
        GroupLayout layout = new GroupLayout(getContentPane());
        getContentPane().setLayout(layout);//给隐藏容器设置布局管理器 layout，后续可以通
过 layout 管理容器的布局

        // 设置水平和垂直的间隔
        layout.setAutoCreateGaps(true);
        layout.setAutoCreateContainerGaps(true);

       // 添加组件到布局
       layout.setHorizontalGroup(
              layout.createSequentialGroup()
                     .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)//
创建一个并行组，左对齐
                            .addComponent(usernameLabel)//添加组件到组中
                            .addComponent(passwordLabel)
                     )//添加组件到组中
                     .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                            .addComponent(usernameField)
                            .addComponent(passwordField)
                            .addComponent(loginButton, GroupLayout.Alignment.CENTER)
                     )
       );
        //先是用户名标签和密码标签并排，然后是对应的文本字段也并排，最后是登录按钮。

       layout.setVerticalGroup(
              layout.createSequentialGroup()
                     .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                            .addComponent(usernameLabel)
                            .addComponent(usernameField))
                     .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                            .addComponent(passwordLabel)
                            .addComponent(passwordField)
                     )
                     .addComponent(loginButton)
```

```
       );
        // 显示所有内容
        this.setVisible(true);
        // 设置界面置顶
        this.setAlwaysOnTop(true);
    }
}

//主方法调用类
public class main {
    public static void main(String[] args) {
        ui myLogin=new ui();
    }
}
```

实验任务 2

```
//myMenu 类
import javax.swing.*;
public class myMenu extends JFrame{
    public  myMenu(){
        //设置界面大小
        this.setSize(400,250);
        //设置界面的 Title
        this.setTitle("Menu");
        //设置界面居中
        this.setLocationRelativeTo(null);
        //设置关闭模式
        this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        //初始化菜单，创建整个菜单对象
        JMenuBar jMenuBar=new JMenuBar();

        //创建菜单上三个选项的对象
        JMenu FileJMenu=new JMenu("File");
        JMenu FormatJMenu=new JMenu("Format");
        JMenu HelpJMenu=new JMenu("Help");

        //创建选项 Format 的选项
        JMenuItem LanguageItem=new JMenuItem("中文");
        JMenuItem SystemItem=new JMenuItem("进制");

        //菜单上面的选项添加到菜单对象 jMenuBar 中
        jMenuBar.add(FileJMenu);
        jMenuBar.add(FormatJMenu);
        jMenuBar.add(HelpJMenu);
```

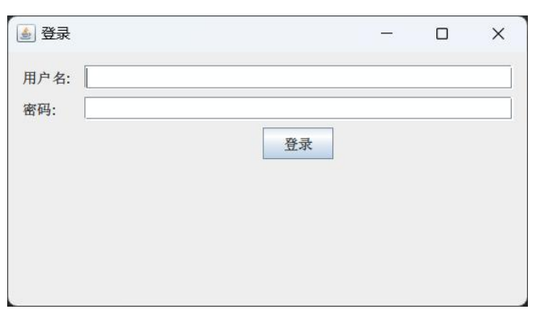
```
        //菜单的选项 Format 添加其对应的选项
        FormatJMenu.add(LanguageItem);
        FormatJMenu.add(SystemItem);

        //为界面设置菜单
        this.setJMenuBar(jMenuBar);
        //显示所有内容，包括界面、菜单、菜单上的选项、选项的选项
        this.setVisible(true);
        //设置界面置顶，沉底
        this.setAlwaysOnTop(true);
    }
}

//主方法调用类
public class mainMenu {
    public static void main(String[] args) {
        myMenu mymenu=new myMenu();
    }
}
```

实验任务 3

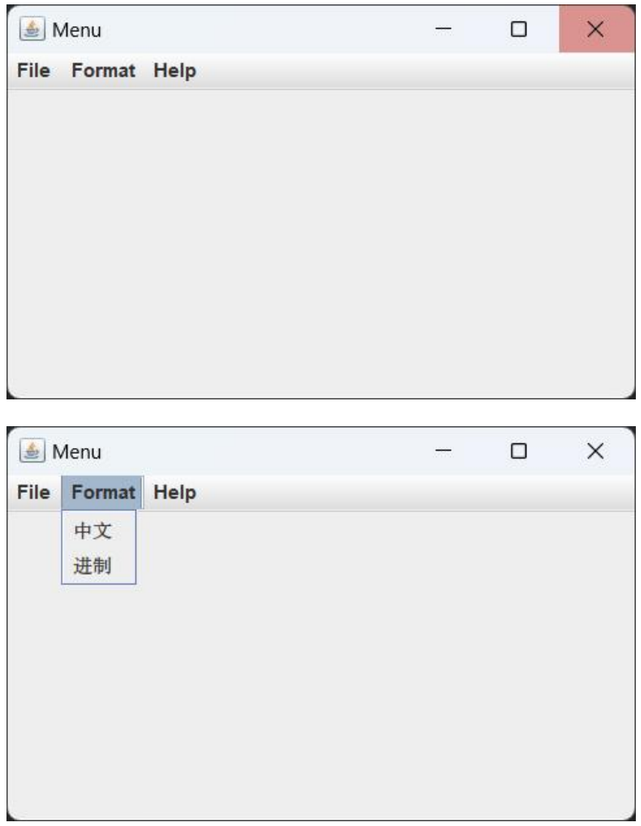
```
// 定义一个站点类，继承自 Thread 类
public class Station extends Thread {

    // 所有对象共享的卖票数据
    static int ticket = 0;  // 卖出的票数，初始化为 0
    static boolean ticketsSoldOut = false;  // 是否卖光了的标志，初始化为 false
    static Object obj = new Object();  // 对象锁，用于同步

    //定义一个带参的构造函数，用于新建线程的同时可以顺便把名字给赋值
    public Station(String name) {
        super(name);  // 调用 Thread 类的构造方法，设置线程的名称
    }

    // 线程执行的主体方法
    @Override
    public void run() {
        //循环执行卖票的操作
        while (true) {
            // 使用 synchronized 关键字保证线程安全，锁定 obj 对象
            synchronized (obj) {
                // 如果卖的票数小于 20，继续卖票
                if (ticket < 20) {
                    ticket++;  // 增加卖出的票数
                    System.out.println(Thread.currentThread().getName() + "卖出了第" +
ticket + "张票");
                } else {
                    // 如果票卖完了且还没有标记为卖光
                    if (!ticketsSoldOut) {
                        System.out.println("票卖完了！ ");
                        ticketsSoldOut = true;  // 标记为卖光
                    }
                    // 退出循环，结束线程执行
                    break;
                }
            }
            Thread.yield();  // 重新分配 CPU，线程重新抢 CPU
        }
    }
}

//主方法调用类
public class SaleTicketMain {
    public static void main(String[] args) {
        //设置三个售票窗口
        Station Counter1=new Station("Counter1");
        Station Counter2=new Station("Counter2");
        Station Counter3=new Station("Counter3");
        //三个窗口同时卖票
        Counter1.start();
        Counter2.start();
        Counter3.start();
    }
}
```

四、实验过程原始数据记录（程序运行结果截图）

实验任务 1



实验任务 2



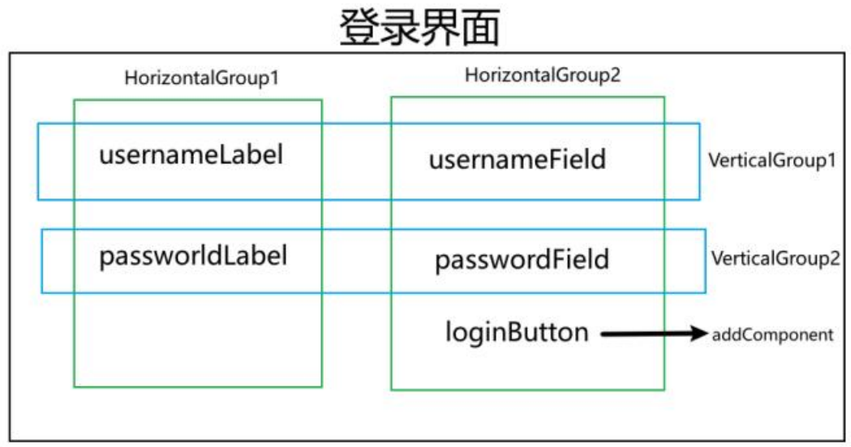
实验任务 3

```
"D:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:D:\JavaDevelop\IntelliJ IDEA 2021.2.1\li
Counter2卖出了第1张票
Counter2卖出了第2张票
Counter2卖出了第3张票
Counter2卖出了第4张票
Counter2卖出了第5张票
Counter2卖出了第6张票
Counter2卖出了第7张票
Counter2卖出了第8张票
Counter3卖出了第9张票
Counter3卖出了第10张票
Counter3卖出了第11张票
Counter1卖出了第12张票
Counter1卖出了第13张票
Counter1卖出了第14张票
Counter1卖出了第15张票
Counter1卖出了第16张票
Counter1卖出了第17张票
Counter1卖出了第18张票
Counter1卖出了第19张票
Counter1卖出了第20张票
票卖完了!
```

五、实验结果及分析

实验一：

实验一新定义一个 ui 类继承 JFrame 类，通过 setSize()，setTitle()，setLocationRelativeTo() 等方法设置界面，然后，创建 JLabel，JTextField 等组件，最后使用布局管理器 GroupLayout，实现水平和垂直方向的组件布局，基本框架如图所示。



问题一：

各个组件粘在一起，不美观。

改进——添加两行代码，增加组件之间的间距：

```
layout.setAutoCreateGaps(true);
layout.setAutoCreateContainerGaps(true);
```

问题二：

登录按钮错位。

原因——登录按钮独自在一个 HorizontalGroup 中，所以自己一列

```
layout.setHorizontalGroup(
        layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                        .addComponent(usernameLabel)//添加组件到组中
                        .addComponent(passwordLabel)
                )//添加组件到组中
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                        .addComponent(usernameField)
                        .addComponent(passwordField)
                )
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                        .addComponent(loginButton)
                )

);
```

改进：

```
layout.setHorizontalGroup(
        layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                 //创建一个并行组，左对齐
                        .addComponent(usernameLabel)//添加组件到组中
                        .addComponent(passwordLabel)//添加组件到组中
                )
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                        .addComponent(usernameField)
                        .addComponent(passwordField)
                        .addComponent(loginButton, GroupLayout.Alignment.CENTER)
                )
);
```

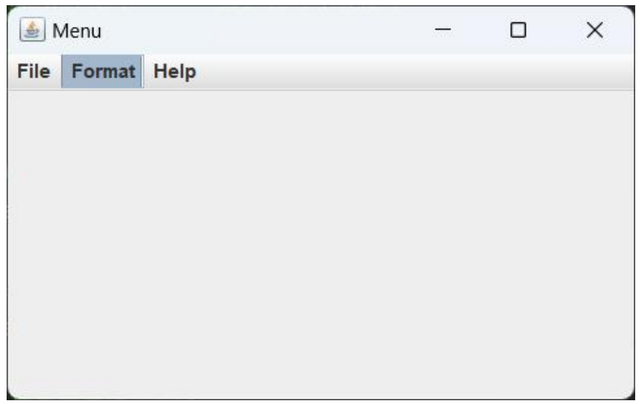
登录按钮添加到第二个 HorizontalGroup 中，以其他方式实现居中对齐。

实验二：

实验二中定义了一个名为 myMenu 的 Java 类，继承自 JFrame。首先，通过 setSize()，setTitle()，setLocationRelativeTo()等方法设置界面。接着，通过 JMenuBar 创建了一个菜单栏对象，并初始化了三个主菜单选项："File"、"Format"和"Help"。在"Format"菜单下，创建了两个子选项："中文"和"进制"。

然后，通过 jMenuBar.add()方法将菜单上的选项添加到菜单对象 jMenuBar 中，通过 setJMenuBar(jMenuBar)方法将菜单栏对象设置到窗口中，确保菜单能够与窗口关联。最后，通过 setVisible(true)显示了整个窗口，包括窗口本身、菜单栏以及各个菜单选项。另外，通过 setAlwaysOnTop(true)设置窗口置顶，使其始终显示在其他窗口之上。

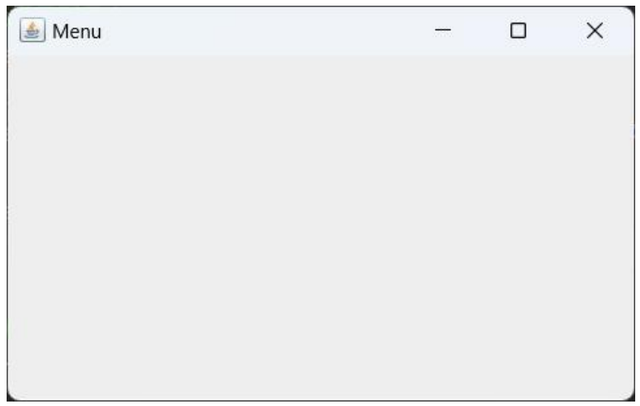
问题一：



Format 下没有选项。

原因：没有使用 JMenu 的对象的 add 方法将 JMenuItem 添加到 JMenu 的对象中。

问题二：



没有显示菜单。

原因：this.setVisible()方法放的位置错了，应该要放在选项添加到菜单、菜单添加到界面之后。

实验三：

实验三中，创建了一个名为 Station 的类，该类继承自 Thread 类，用于模拟一个火车站的售票窗口。每个窗口对应一个线程，通过继承 Thread 实现多线程。

Station 类的每个对象包括三个静态变量：ticket 表示已售票数（初始为 0），ticketsSoldOut 是票是否售罄的标志（初始为 false，表示没有卖光），以及 obj 作为对象锁进行同步。

在 run()方法中，用 while 无限循环模拟一直卖票的过程。在卖票的过程中，使用 synchronized 关键字锁定对象 obj，确保在同一时刻只有一个线程能够进入临界区，避免出现“票卖多了”，“同一张票卖了多次”，“某些票没有卖”的情况，从而保证线程安全。

在每次卖票后，通过 Thread.yield()实现线程让步，重新分配 CPU 资源，让其他线程有机会抢占 CPU。

问题一：

```
"D:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:D:\JavaDevelop
Counter1卖出了第1张票
Counter1卖出了第2张票
Counter1卖出了第3张票
Counter1卖出了第4张票
Counter1卖出了第5张票
Counter1卖出了第6张票
Counter1卖出了第7张票
Counter1卖出了第8张票
Counter1卖出了第9张票
Counter1卖出了第10张票
Counter1卖出了第11张票
Counter1卖出了第12张票
Counter1卖出了第13张票
Counter1卖出了第14张票
Counter1卖出了第15张票
Counter1卖出了第16张票
Counter1卖出了第17张票
Counter1卖出了第18张票
Counter1卖出了第19张票
Counter1卖出了第20张票
```

只有一个窗口在卖票。

原因——synchronized 关键字锁错地方了：

```
@Override
public void run() {
    synchronized(obj) {
        while (ticket <= 100) {
            ticket++;
            System.out.println(Thread.currentThread().getName() + "卖出了第" + ticket
+ "张票");
        }
    }
}
```

应该要锁在 while 里面而不是外面，不然只有等第一个进去的线程卖光了才轮到其他线程。

问题二：

```
Counter1卖出了第11张票
Counter1卖出了第12张票
Counter1卖出了第13张票
Counter1卖出了第14张票
Counter1卖出了第15张票
Counter1卖出了第16张票
Counter1卖出了第17张票
Counter1卖出了第18张票
Counter1卖出了第19张票
Counter1卖出了第20张票
票卖完了！
票卖完了！
票卖完了！
```

票会被卖光 3 次。

原因：

```
@Override
 public void run() {
     // 循环执行卖票的操作
     while (true) {
         // 使用 synchronized 关键字保证线程安全，锁定 obj 对象
         synchronized (obj) {
             // 如果卖的票数小于 20，继续卖票
             if (ticket < 20) {
                 ticket++;  // 增加卖出的票数
                 System.out.println(Thread.currentThread().getName() + "卖出了第" +
ticket + "张票");
             } else {
                 // 如果票卖完了且还没有标记为卖光
                     System.out.println("票卖完了！ ");
                 // 退出循环，结束线程执行
                 break;
             }
         }
         Thread.yield();  // 重新分配 CPU，线程重新抢 CPU
     }
 }
```

synchronized 关键字在是在 while 循环中锁定对象 obj，这意味着可能有多个线程在 while 循环中等待进入临界区，即等待其他线程释放锁。当一个线程售罄所有票后，其他线程仍然可以获取锁并继续执行，然后打印“票卖完了！”。

改进——添加一个旗帜用于判断票是否售罄即可：

```
    @Override
```

```
    public void run() {
        // 循环执行卖票的操作
        while (true) {
            // 使用 synchronized 关键字保证线程安全，锁定 obj 对象
            synchronized (obj) {
                // 如果卖的票数小于 20，继续卖票
                if (ticket < 20) {
                    ticket++;  // 增加卖出的票数
                    System.out.println(Thread.currentThread().getName() + "卖出了第" +
ticket + "张票");
                } else {
                    // 如果票卖完了且还没有标记为卖光
                    if (!ticketsSoldOut) {
                        System.out.println("票卖完了！ ");
                        ticketsSoldOut = true;  // 标记为卖光
                    }
                    // 退出循环，结束线程执行
                    break;
                }
            }
            Thread.yield();  // 重新分配 CPU，线程重新抢 CPU
        }
    }
```