实验课程名称：Java语言程序设计A

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验项目名称** | **实验8：GUI设计** | | | **实验成绩** |  |
| **实 验 者** |  | **专业班级** | **信管1502** | **组 别** | **1人1组** |
| **同 组 者** | **无** | | | **开始日期** | **2018年6月5日** |
| 第一部分：实验预习报告（包括实验目的及意义，实验基本原理与方法，主要仪器设备及耗材，实验内容及要求，实验方案与技术路线）  一．实验目的及意义  1．理解并应用事件处理机制。  2．掌握Swing简单的窗体及按钮设计。  3．理解并应用Java FX委托事件处理模型。  4．掌握Java FX的简单的窗体及按钮设计。  二．实验基本原理与方法  1．事件处理机制及ActionListener接口的实现。  2．使用Swing类库设计GUI的大体思路。  3．Java FX委托事件处理模型  4．使用Java FX设计GUI的大体思路。  三．主要仪器设备及耗材  1．PC及其兼容机。  2．计算机操作系统。  3．程序编辑器EditPlus。  4．Java开发工具JDK。  四．实验内容及要求  实验8-1：编写Swing应用程序，显示一窗体，窗体标题栏显示"Subject"，窗体正上方有一按钮，按钮上显示"Which Lesson?"，单击按钮，窗体中央显示信息"Java Programming!"。  实验8-2：编写Swing应用程序，显示一窗体，窗体标题栏显示"My Swing Program"，窗体中间有两个按钮，按钮上分别显示"OK"、"Cancel"，单击OK按钮，显示"Welcome to Swing Program! "，单击"Cancel"，退出应用程序。  实验8-3：使用Java FX，编写一个简单的GUI应用程序，显示一个窗体，窗体标题栏显示"My Java FX Program"，窗体中间有两个按钮，按钮上分别显示"OK"、"Cancel"，单击OK按钮，显示"Welcome to Java FX Program! "，单击"Cancel"，退出应用程序。  五．实验方案及技术路线（设计思路、算法、注意事项）  实验8-1设计思路  所有实验的设计思路都和题目要求有关，界面要求按钮在上方，在设置按钮控件时就要指明改按钮在上方，窗体中央显示信息的要求则需要Label控件置于中央。  实验8-2设计思路  本实验要求两个按钮都在Swing窗体的中间，可以利用container类的网格布局解决——GridLayout三行三列，中间的格子放置一个JPanel（面板），面板可以放置两个按钮。  实验8-3设计思路  本实验使用JavaFX元素，具体实现方式如下图：    （舞台内部，UI元素们成树形结构，根节点RootNode为FlowPane“流式面板”类）  第二部分：实验过程记录（可加页）（代码、运行结果、实验中出现的问题及解决过程）  ————————————————实验8-1代码———————————————————  import java.awt.BorderLayout;  import java.awt.Container;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.SwingConstants;  import javafx.scene.layout.Border;  public class SwingSubject  {  public static void main (String[] args)  {  Base base = new Base();  base.CreatJFrame("Subject");  }  }  class Base extends JFrame  {  private static final long serialVersionUID = -7945004261284306797L;      //某个神秘码，不声明会触发本JAVA编译器的警报机制  public void CreatJFrame(String title)  {  JFrame jf = new JFrame(title);  Container container= jf.getContentPane();  JButton jb = new JButton("Which Lesson?");  JLabel jl = new JLabel("Java Programming!",SwingConstants.CENTER);  container.add(BorderLayout.NORTH,jb);  jf.add(BorderLayout.CENTER,jl);  jf.setVisible(true);  jl.setVisible(false);  jf.setSize(800,450);  jf.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  jb.addActionListener(new ActionListener()  {  public void actionPerformed(ActionEvent e)  {  jl.setVisible(true);  }  });  }  }  ————————————————实验8-1运行———————————————————    点击按钮    ————————————————实验8-2代码———————————————————  import java.awt.Container;  import java.awt.GridLayout;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.\*;  public class MySwingProgram  {  public static void main(String[] args)  {  new Base2().CreateJFrame("My first Swing Program");  }  }  class Base2 extends JFrame  {  private static final long serialVersionUID = 4761583961790092691L;      //某个神秘码，不声明会触发本JAVA编译器的警报机制  public void CreateJFrame(String title)  {  JFrame jf = new JFrame(title);  Container container = jf.getContentPane();  container.setLayout(new GridLayout(3,3,10,10));  JButton jbOk=new JButton("OK");  JButton jbCancel= new JButton("Cancel");  JLabel jl = new JLabel("");  JPanel jp= new JPanel(new GridLayout(1,3,10,10));  jp.add(jbOk);  jp.add(jbCancel);  container.add(new JLabel(""));  container.add(new JLabel(""));  container.add(new JLabel(""));  container.add(new JLabel(""));  container.add(jp);  container.add(new JLabel(""));  container.add(new JLabel(""));  container.add(jl);  container.add(new JLabel(""));  //在容器的九个格子布局里填充，使得最后两个按钮能产生放在中间的效果  jf.setVisible(true);  //jl.setVisible(false);  jf.setSize(800,450);  jf.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  jbOk.addActionListener(new ActionListener(){    @Override  public void actionPerformed(ActionEvent e) {  jl.setText("Welcome to my first Swing Program!");  }  });  jbCancel.addActionListener(new ActionListener(){    @Override  public void actionPerformed(ActionEvent e) {  jf.dispose();  }  });  }  }  ————————————————实验8-2运行———————————————————    点击OK按钮后    ————————————————实验8-3代码———————————————————  import javafx.application.\*;  import javafx.stage.\*;  import javafx.scene.\*;  import javafx.scene.control.\*;  import javafx.scene.layout.\*;  import javafx.event.\*;  import javafx.geometry.\*;  @SuppressWarnings("restriction")  //如果不做此声明，会触发JAVA的警报机制  //要想解决这个问题，先在配置界面修改相关的设置  public class MyJavaFX extends Application  {  Label response;  public void start(Stage myStage)  {  myStage.setTitle("MyJavaFXProgram");  FlowPane rootNode = new FlowPane(10,10);  rootNode.setAlignment(Pos.CENTER);  Scene myScene = new Scene(rootNode,800,450);    //16:9  myStage.setScene(myScene);  response = new Label("");  Button OK = new Button("OK");  Button Cancel = new Button("Cancel");  OK.setOnAction(new EventHandler<ActionEvent>(){    @Override  public void handle(ActionEvent event) {  response.setText("Welcome to my Java FX Program!");  }  });  Cancel.setOnAction(new EventHandler<ActionEvent>(){    @Override  public void handle(ActionEvent event) {  myStage.close();  }  });  rootNode.getChildren().add(OK);  rootNode.getChildren().add(Cancel);  rootNode.getChildren().add(response);  myStage.show();  }  }    ————————————————实验8-3运行———————————————————    按下“OK”按钮后    第三部分 结果与讨论（可加页）  一、实验结果分析  第一，在实验8-1中，Java语法允许在类中声明引用变量（即引用自己的变量，同时也允许构造自己）。但是这种方式逻辑性不好，不能体现Java语言编程之美，强烈不建议这样使用。  第二，实验8-2和实验8-3都是要求设置一个窗体，显示两个控件。但是后台的实现逻辑不同。Swing窗体会覆盖“UI”，即如果同时在某一个位置多次放置控件，这些控件会不会报错，但是会相互覆盖，只显示最后一个控件。但是JavaFX处理的机制不同，所有的元素都会被解释成一个节点。  二、小结、建议及体会  关于“Serializable”接口  java.io，javax.swing.JFrame实现了这个接口。  这个接口的主要的功能是序列化，当继承了这个接口，Java会启动序列化功能—进行序列化和反序列化。  序列化的主要目的是：保持版本的兼容性，当有版本升级时，序列化让对象保持唯一性。  当把对象写进流里就会根据此机制生成一个相应的序号，从流里读取对象时就根据这个序号来检查对象的完整性，于是io包封装了这个接口。  生成序号的过程有点繁琐，我们也不仅仅只在JFrame中接触到此IO，如果我们想要在服务器端编程某个类来处理和返回客户端的IO流，Servlet是个不错的选择。在Servlet类之中，JAVA编译器IDE MyEclipse会自动地帮我们生成一个serialVersionUID：  private static final longserialVersionUID = 1L;    这是一个默认的解决方案。  再一个是是根据类名、接口名、成员方法及属性等来生成一个64位的哈希字段：    当我们程序里面的某一个类实现了Serializable接口，如果没有定义serialVersionUID，MyEclipse会提供这个提示功能告诉我们去定义——尽管可以编译通过，但是会有潜在的风险：    这个serialVersionUID为了让该类别Serializable向后兼容。若此类Serialized保存在硬盘之后，遭遇了发生在此类别的field(增加或减少或改名)事件，当你Deserialize时，就会出现Exception“不兼容”。  当serialVersionUID不再改变，更改过的field就会以type的预设值Deserialize，以达到避开不兼容问题的目的。  第四部分 实验评分标准（教师自行设计）及成绩   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | 观测点 | 考核目标 | 权重 | 得分 | | 实验预习 | 程序设计方案、技术路线（设计思路、算法、注意事项）科学、合理、可行 | 实验方案设计能力 | 20% |  | | 实验过程 | 1. 按时参加实验，不迟到，不早退 2. 源代码正确、完整、可读性强、编码规范，运行结果正确，实验中出现的问题得到解决 | 学生的实验态度、UML面向对象建模与设计能力、分析并解决问题的专业素养及编程技能、良好的编程风格与习惯 | 50% |  | | 结果分析 | 实验结果分析到位、总结完善、体会深刻 | 学生对实验数据处理与分析的能力；对专业知识的综合应用能力；事实求实的精神 | 30% |  | | 该项实验报告最终得分 | | |  | |   教师签名： | | | | | |