108 《多媒体技术与应用》 实验报告 2008～2009学年第一学期 学 院 化学与分子工程学院 班 级 应材071 姓 名 张三 学 号 10079999 任课教师 李飞（周五3,4节） 计算机教学实验中心 2008年 10 月 《多媒体技术与应用》实 验 报 告 学号： 姓名： 班级： 实验名称：Photoshop综合实验：制作足球 实验地点： 所使用的工具软件及环境： 一、实验目的： 1．掌握Photoshop的形状工具； 2东然侠伙斜劫橇添梅渝仗酬翘答厂及脯辈采柠藏竣旁碰壶窥石毋憋丝甘湾野咎家违枉嗣厂攻猿前

**浙江理工大学**

**《Java程序设计》**

**实验报告**

**20244 ～2025 学年第2 学期**

|  |  |
| --- | --- |
| **学 院** | **计算机科学与技术学院** |
| **班 级** | **23计科（）班** |
| **姓 名** | **盲灯** |
| **学 号** |  |
| **任课教师** | **孙麒**  （上课时间：周四6-8 节） |

**《Java程序设计》实 验 报 告（二）**

|  |  |
| --- | --- |
| **实验名称**: 异常处理 | **实验地点**： 10-413 |
| **所使用的工具软件及环境：**  环境：Windows 8.1;jdk 1.8;  工具：eclipse | |
| **实验目的：**  了解异常处理机制，掌握异常处理的设计方法。 | |
| **实验内容：**  **用户自定义一个异常，编程创建并抛出某个异常类的实例。运行该程序并观察执行结果。**  **例如：用户密码的合法化验证。**  **要求密码有4-6个数字组成。若长度不落在这个范围或不是有数字组成。抛出自己的异常。**  **要求：**  **1．图形界面，面向对象编程。**  **2．异常定义，抛出，捕捉三个流程。** | |
| **代码实现：**  **我的代码分为四个部分**  **第一个部分：PasswordException.java，用于异常定义**  package shiyan2;  public class PasswordException extends Exception {      public PasswordException(String message) {          super(message);      }  }  **第二个部分：PasswordValidator.java，用于抛出异常**  package shiyan2;  public class PasswordValidator {      public static void validatePassword(String password) throws PasswordException {          if (password.length() < 4 || password.length() > 6) {              throw new PasswordException("密码长度必须在4到6个数字" +                      "之间");          }          if (!password.matches("\\d+")) {              throw new PasswordException("密码必须由数字组成");          }      }  }  **第三部分：PasswordApp.java,用于实现密码验证功能，包含捕获异常和可视化页面的实现**  package shiyan2;  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  public class PasswordApp extends JFrame {      private JTextField passwordField;      private JLabel label, resultLabel;      private JButton validateButton;      public PasswordApp() {          setTitle("密码验证");          setSize(400, 300);          setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);          setLocationRelativeTo(null);  *// 设置主布局为 BorderLayout*          setLayout(new BorderLayout());  *// 创建顶部面板*          JPanel topPanel = new JPanel();          topPanel.setLayout(new GridLayout(1, 2));          topPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));  *// 添加标签*          label = new JLabel("请输入密码（4-6位数字）:");          topPanel.add(label);  *// 添加输入框*          passwordField = new JTextField(20);          topPanel.add(passwordField);  *// 创建中间面板*          JPanel centerPanel = new JPanel();          centerPanel.setLayout(new GridBagLayout());          centerPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));  *// 添加验证按钮*          validateButton = new JButton("验证");          validateButton.addActionListener(new ActionListener() {              @Override              public void actionPerformed(ActionEvent e) {                  String password = passwordField.getText();                  try {                      PasswordValidator.validatePassword(password);                      resultLabel.setText("密码合法！");                      resultLabel.setForeground(Color.GREEN);                  } catch (PasswordException ex) {                      resultLabel.setText("密码无效: " + ex.getMessage());                      resultLabel.setForeground(Color.RED);                  }              }          });  *// 使用 GridBagConstraints 来布局组件*          GridBagConstraints gbc = new GridBagConstraints();          gbc.gridx = 0;          gbc.gridy = 0;          gbc.insets = new Insets(10, 10, 10, 10);          centerPanel.add(validateButton, gbc);  *// 创建底部面板*          JPanel bottomPanel = new JPanel();          bottomPanel.setLayout(new BorderLayout());          bottomPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));  *// 添加结果标签*          resultLabel = new JLabel("");          resultLabel.setHorizontalAlignment(SwingConstants.CENTER);          bottomPanel.add(resultLabel, BorderLayout.CENTER);  *// 将所有面板添加到主框架*          add(topPanel, BorderLayout.NORTH);          add(centerPanel, BorderLayout.CENTER);          add(bottomPanel, BorderLayout.SOUTH);      }      public static void main(String[] args) {          SwingUtilities.invokeLater(new Runnable() {              @Override              public void run() {                  new PasswordApp().setVisible(true);              }          });      }  }  **第四个部分：PasswordTest.java,为测试类用于运行主程序**  package shiyan2;  public class PasswordTest {      public static void main(String[] args) {          PasswordApp.main(args);      }  } | |

|  |
| --- |
| **实验结果：**   1. **运行之后运行结果如图所示**   16d1faa3-305a-4346-b766-fd7c4247134a   1. **输入正确的密码结果如图所示**        1. **输入密码长度过长或过短结果如图所示**        1. **输入密码为非数字如图所示** |
| **收获与体会**   1. **在实际开发中，异常处理是非常重要的一部分。通过自定义异常类PasswordException，并使用throw关键字抛出异常，我深刻理解了如何在代码中捕获和处理异常。这不仅提高了代码的健壮性，还增强了用户体验，避免了程序因未处理的异常而崩溃。** 2. **通过使用Swing库创建了一个更大的、更美观的GUI界面，我对图形界面的设计有了更深的认识。特别是使用BorderLayout和GridBagLayout布局管理器，使得界面更加整洁和易于操作。这次实验让我明白，一个好的用户界面不仅仅是功能性的，还需要考虑用户的视觉体验。** 3. **在密码验证结果的显示中，我采用了绿色表示成功，红色表示失败。这种鲜明的颜色设置不仅使用户能够迅速理解验证结果，还提升了界面的美观度。这种细节上的处理让程序更加人性化，用户体验也更好。** 4. **为了让界面更加美观和舒适，我调整了窗口的大小，增加了组件之间的间距，并添加了边框和内边距。这些小改动让整个界面看起来更加大气和专业。** 5. **这次实验采用了面向对象编程的思想，将代码分为几个独立的类，如PasswordException、PasswordValidator和PasswordApp。这种模块化的编程方式不仅使代码结构更加清晰，还便于维护和扩展。通过这种方式，我在实际编程中更好地体会到了面向对象编程的优势。** |