|  |
| --- |
| 1、系统功能： 自动贩卖机系统对自动贩卖机的商品补充、套餐的购买、订单创建、贩卖机的维修、开启和关闭进行了简单的模拟，运用工厂方法模式、建造者模式、单例模式、职责链模式、外观模式对系统进行了优化，极大地拓展了系统的可维护性，可读性。   2、系统详细设计2.1补充商品 功能：在贩卖机缺货的时候，输入工厂名称以补充商品  模式：工厂方法模式  模式优点： 管理者只需知道工厂的类名，无需知道内部运行的逻辑即可对商品进行补充。如果有新的商品要加入，无需修改源代码，只需增加工厂类和产品类，即可增加新的商品。     1. **package** add\_item; 2. **import** java.awt.FlowLayout; 3. **import** javax.swing.\*; 4. **import** java.awt.\*; 5. **import** java.awt.event.ActionEvent; 6. **import** java.awt.event.ActionListener; 8. **public** **class** Client\_add\_item **extends** JFrame { 9. JPanel panel = **new** JPanel(); 10. JLabel pwp = **new** JLabel("工厂名"); 11. JTextField qwq = **new** JTextField(10); 12. JButton repair = **new** JButton("补充商品"); 14. **public** Client\_add\_item(String name) { 15. **this**.setLayout(**new** FlowLayout(FlowLayout.LEFT)); 16. **this**.setSize(600, 100); 17. **this**.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); 18. **this**.setLocationRelativeTo(**null**); 19. **this**.setVisible(**true**); 20. **this**.setTitle("自动贩卖机系统"); 21. **this**.setLayout(**new** FlowLayout(1, 10, 10)); 23. repair.addActionListener(**new** repairListener()); 25. **this**.add(pwp); 26. **this**.add(qwq); 27. **this**.add(repair); 28. } 30. **class** repairListener **implements** ActionListener { 31. @Override 32. **public** **void** actionPerformed(ActionEvent e) { 33. **try** { 34. Item ab\_drink; 35. Item\_Factory ab\_fact; 36. String cName = "add\_item."; 37. String str\_factory = qwq.getText(); 38. cName = cName + str\_factory; 39. Class c = Class.forName(cName); 40. Object obj = c.newInstance(); 41. ab\_fact = (Item\_Factory) obj; 42. ab\_drink = ab\_fact.produceDrink(); 43. ab\_drink.add(); 44. } **catch** (Exception b) { 45. System.out.println(b.getMessage() + "qwq"); 46. } 47. } 49. } 50. }  2.2购买套餐 功能：以套餐的形式，一次购买多个商品，促进消费  模式：建造者模式  模式优点： 用户只需知道套餐的类名，无需知道内部运行的逻辑即可对商品进行购买。如果管理员想要增加新的套餐，无需修改源代码，只需增加套餐子类，即可增加新的套餐。     1. **package** add\_meat; 3. **import** java.awt.FlowLayout; 5. **import** javax.swing.\*; 7. **import** java.awt.\*; 8. **import** java.awt.event.ActionEvent; 9. **import** java.awt.event.ActionListener; 11. **public** **class** Client\_add\_meat **extends** JFrame { 12. JPanel panel = **new** JPanel(); 13. JLabel pwp = **new** JLabel("套餐名"); 14. JTextField qwq = **new** JTextField(10); 15. JButton repair = **new** JButton("购买套餐"); 17. **public** Client\_add\_meat(String name) { 18. **this**.setLayout(**new** FlowLayout(FlowLayout.LEFT)); 19. **this**.setSize(600, 100); 20. **this**.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); 21. **this**.setLocationRelativeTo(**null**); 22. **this**.setVisible(**true**); 23. **this**.setTitle("自动贩卖机系统"); 24. **this**.setLayout(**new** FlowLayout(1, 10, 10)); 26. repair.addActionListener(**new** repairListener()); 28. **this**.add(pwp); 29. **this**.add(qwq); 30. **this**.add(repair); 31. } 33. **class** repairListener **implements** ActionListener { 34. @Override 35. **public** **void** actionPerformed(ActionEvent e) { 36. **try** { 37. String cName = "add\_meat."; 38. String str\_factory = qwq.getText(); 39. cName = cName + str\_factory; 40. Class c = Class.forName(cName); 41. Object obj = c.newInstance(); 42. MealBuilder mb = (MealBuilder)obj; 43. Admin user\_ = **new** Admin(); 44. user\_.setMealBuilder(mb); 45. Meal meal = user\_.construct(); 46. System.out.println("套餐："); 47. System.out.println(meal.getNoodles()); 48. System.out.println(meal.getDrink()); 49. System.out.println(meal.getMeat()); 50. } **catch** (Exception b) { 51. System.out.println(b.getMessage() + "qwq"); 52. } 53. } 55. } 56. }  2.3创建订单 功能：一次交易对应生成一个订单。  模式：单例模式  模式优点：单一交易生成一个订单以确保交易的成功进行，避免因生成多个订单造成其他对象对订单访问的混乱，如果不使用单一模式，可能会造成支付后没有获得商品的情况。     1. **package** buy\_order; 3. **public** **class** Order\_No { 4. **private** **static** Order\_No instance = **null**; 5. **private** String no; 6. **private** Order\_No() 7. { 9. } 11. **public** **static** Order\_No getInstance() { 12. **if** (instance == **null**) { 13. System.out.println("获得新订单号"); 14. instance = **new** Order\_No(); 15. instance.setOrder\_No("1"); 16. } 17. **else** { 18. System.out.println("重复下单且与上一个未交易的订单相同，获取旧订单号"); 19. } 20. **return** instance; 21. } 23. **private** **void** setOrder\_No(String no) { 24. **this**.no = no; 25. } 27. **public** String getOrder\_No() { 28. **return** **this**.no; 29. } 31. }  2.4外观模式 功能：贩卖机的开启和关闭  模式：外观模式  模式优点：贩卖机的启动与关闭只需按下开关机按钮，管理者无需知道硬件内部实现的操作，通过外观模式来实现管理者对贩卖机的管理。     1. **package** open\_machine; 3. **import** java.awt.FlowLayout; 5. **import** javax.swing.\*; 7. **import** java.awt.\*; 8. **import** java.awt.event.ActionEvent; 9. **import** java.awt.event.ActionListener; 10. **public** **class** Client\_Control **extends** JFrame 11. { 12. JButton open = **new** JButton("开启贩卖机"); 13. JButton close = **new** JButton("关闭贩卖机"); 14. **public** Client\_Control(String name){ 15. **this**.setLayout(**new** FlowLayout(FlowLayout.LEFT)); 16. **this**.setSize(600,100); 17. **this**.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); 18. **this**.setLocationRelativeTo(**null**); 19. **this**.setVisible(**true**); 20. **this**.setTitle("自动贩卖机系统"); 21. **this**.setLayout(**new** FlowLayout(1,10,10)); 23. open.addActionListener(**new** openListener()); 24. close.addActionListener(**new** closeListener()); 26. **this**.add(open); 27. **this**.add(close); 28. } 29. **class** openListener **implements** ActionListener{ 30. @Override 31. **public** **void** actionPerformed(ActionEvent e) { 32. Main\_Swicth switch\_ = **new** Main\_Swicth(); 33. switch\_.Activate(); 34. } 36. } 37. **class** closeListener **implements** ActionListener{ 38. @Override 39. **public** **void** actionPerformed(ActionEvent e) { 40. Main\_Swicth switch\_ = **new** Main\_Swicth(); 41. switch\_.Deactivate(); 42. } 44. } 45. }  2.5贩卖机维修 功能：准确地通知维修人员对贩卖机进行维修  模式：职责链模式  模式优点： 用户无需知道需要通知哪些技术人员来维修，它将请求的发送者和接收者解耦 ，简化了对象，使得对象不需要知道链的结构 ，允许动态地新增或者删除责任链 。     1. **package** repair; 3. **import** java.awt.FlowLayout; 5. **import** javax.swing.\*; 7. **import** java.awt.\*; 8. **import** java.awt.event.ActionEvent; 9. **import** java.awt.event.ActionListener; 10. **public** **class** Client\_repair **extends** JFrame 11. { 12. JPanel panel=**new** JPanel(); 13. JLabel pwp = **new** JLabel("故障程度"); 14. JTextField qwq = **new**   JTextField(10); 15. JButton repair = **new** JButton("贩卖机维修"); 16. **public** Client\_repair(String name){ 17. **this**.setLayout(**new** FlowLayout(FlowLayout.LEFT)); 18. **this**.setSize(600,100); 19. **this**.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); 20. **this**.setLocationRelativeTo(**null**); 21. **this**.setVisible(**true**); 22. **this**.setTitle("自动贩卖机系统"); 23. **this**.setLayout(**new** FlowLayout(1,10,10)); 25. repair.addActionListener(**new** repairListener()); 27. **this**.add(pwp); 28. **this**.add(qwq); 29. **this**.add(repair); 30. } 31. **class** repairListener **implements** ActionListener{ 32. @Override 33. **public** **void** actionPerformed(ActionEvent e) { 34. Advance lv1,lv2,lv3,lv4; 36. lv1=**new** Lv1\_Machinic(); 37. lv2=**new** Lv2\_Machinic(); 38. lv3=**new** Lv3\_Machinic(); 39. lv4=**new** Lv4\_Machinic(); 41. lv1.setSuccessor(lv2); 42. lv2.setSuccessor(lv3); 43. lv3.setSuccessor(lv4); 45. String str = qwq.getText(); 46. **int** Falut = Integer.parseInt(str); 47. Repair\_Request lr1=**new** Repair\_Request(Falut); 48. lv1.handleRequest(lr1); 49. } 51. } 52. }  3课程设计总结 通过本次课程设计，我对所使用过的几个设计模式：单例模式，工厂方法模式，外观模式，职责链模式，外观模式，都有了更加深入的了解。在系统的设计和模式的选择之间应该根据系统的需求来决定。了解了应在什么样的场景下使用什么样的模式，才能降低系统的耦合度，增强可拓展性，提高系统对资源的运用率与系统的运行效率。 |