北京邮电大学计算机学院2023-2024学年第一学期

软件工程专业《Java 程序设计》-期末课程作业

|  |  |
| --- | --- |
| 班级： 姓名： 禄禄鱼 学号： | |
| 注意事项 | 一、学生必须独立完成本作业。  二、**任何雷同或者有抄袭嫌疑的作业，会导致提交者的总评成绩为0分**。  三、学生可以通过阅读、查找纸质资料或者网上电子资料解答题目。但是，不允许**【抄袭】**  四、请保留作业题目，删除【作业解答】中不必要的文字或者标题，保持作业回答干净整洁  五、作业必须以PDF格式提交。作业文件命名：**学号-姓名.pdf** |

评分标准

1. 输出结果和工程项目截图，截图应清晰，大小合适：10%；
2. 实现程序的基本要求：70%；
3. 代码有必要的注释，符合javadoc规范：10%；
4. 代码结构清晰、缩进合理；文档干净整洁；作业文档中的代码字体大小合理规整，代码有语法加亮：10%；

【**特别强调**】任何抄袭或者疑似抄袭的代码，都将导致你的作业被二次复审。请保存好自己的作业，不要给任何人参考。

Quiz 1（30 points）

Design an object-oriented class hierarchy consisting of the following classes:

An ***Individual*** super-class, which has the instance variables **name** (String type), **address** (String), **phone number** (String), and **email address** (String).

A ***Learner*** sub-class which inherits from Individual and has an additional instance variable called **year** (integer type). The year variable should have integer constants defined indicating first-year through fourth-year status. The constants should be defined in an interface and be implemented by the corresponding class.

A ***Worker*** sub-class which also inherits from Individual and has two additional instance variables: **office location** (String type) and **pay rate** (double type).

A ***Staff*** sub-class which inherits from Worker and has one more additional instance variable called **job title** (String type).

In each of these classes, override the toString() method to return a string consisting of the class name and the name instance variable.

Next, write a test program that:

Instantiates objects of each of the four classes: Individual, Learner, Worker, and Staff.

Stores references to these four objects in an array of the Individual super-class type.

Iterates through the array invoking the toString() method on each object, which will output identifying text due to polymorphism resolving the proper overridden method at runtime.

Quiz 2 (30 points)

1. Write a client program that use UDP socket send a message “Are you Okay” to a server on port 8001.Add appropriate exception handling (10 points).

2) Write a multi-threaded UDP server that listens on port 8001. When a client sends messages, create a new thread to handle that client. In the thread, receive the message, and then append this message to a "messages.txt" file along with a timestamp. Since multiple threads will access this file, synchronize file access to avoid errors (20 points).

*Hints:*

*Use PrintWriter/FileOutputStream to append string to file*

*Synchronize block around file append operation*

Quiz 3 (30 points)

Write a program to simulate a small-scale warehouse operations system with the following requirements:

a) Packages are represented by strings containing a unique ID code. The Warehouse can hold a maximum of MAX\_PACKAGES packages at once.

b) The ArrivalTruck thread repeatedly generates packages, inserts into Warehouse storage, and notifies DepartureTruck thread if storage was full.

c) The DepartureTruck thread repeatedly removes packages, prints shipping details, sleeps for a bit, and notifies ArrivalTruck if storage became available.

d) The Warehouse provides storePackage() and retrievePackage() methods that truck threads can synchronize on.

e) Implement ArrivalTruck, DepartureTruck and Warehouse classes, using threads.

f) Truck threads should use wait/notify to handle warehouse at max capacity or becoming empty.

Quiz 4 (10 points)

You may choose to finish both or either of the following requests and you will get the corresponding scores.

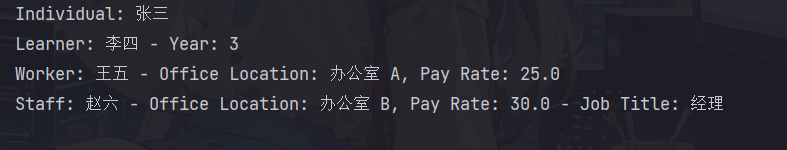
1. Based on Quiz 3, add a UI layer using JavaFX that shows current warehouse storage capacity and number of packages. Update this dashboard dynamically as trucks add/remove packages. (5 points).
2. Based on Quiz 3, truck threads should now record each package arrival/departure using JDBC to a MySQL database table containing columns like: .

CAUTION: Post all your codes for this question here, maybe including some code snippets from quiz 3.

作业解答

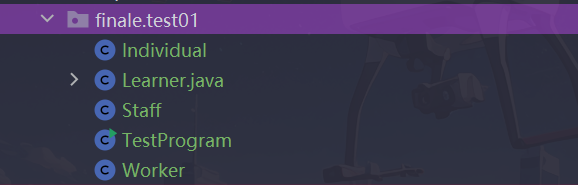
# 题目1

## 输出结果截图



## 工程项目截图

IDE中工程项目截图，显示项目文件



## 源代码

### Individual.java

package school.finale**;**public class Individual {  
 protected String name**;** protected String address**;** protected String phoneNumber**;** protected String emailAddress**;** public Individual(String name**,** String address**,** String phoneNumber**,** String emailAddress) {  
 this.name = name**;** this.address = address**;** this.phoneNumber = phoneNumber**;** this.emailAddress = emailAddress**;** }  
  
 @Override  
 public String toString() {  
 return getClass().getSimpleName() + ": " + name**;** }  
}

### Learner.java

package school.finale**;**import school.finale.Individual**;**interface LearnerYear {  
 int *FIRST\_YEAR* = 1**;** int *SECOND\_YEAR* = 2**;** int *THIRD\_YEAR* = 3**;** int *FOURTH\_YEAR* = 4**;**}  
  
public class Learner extends Individual implements LearnerYear {  
 private int year**;** public Learner(String name**,** String address**,** String phoneNumber**,** String emailAddress**,** int year) {  
 super(name**,** address**,** phoneNumber**,** emailAddress)**;** this.year = year**;** }  
  
 @Override  
 public String toString() {  
 return super.toString() + " - Year: " + year**;** }  
}

### Worker.java

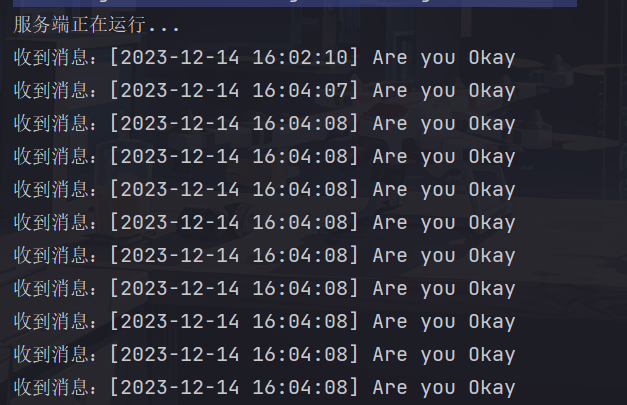
package school.finale**;**public class Worker extends Individual {  
 private String officeLocation**;** private double payRate**;** public Worker(String name**,** String address**,** String phoneNumber**,** String emailAddress**,** String officeLocation**,** double payRate) {  
 super(name**,** address**,** phoneNumber**,** emailAddress)**;** this.officeLocation = officeLocation**;** this.payRate = payRate**;** }  
  
 @Override  
 public String toString() {  
 return super.toString() + " - Office Location: " + officeLocation + ", Pay Rate: " + payRate**;** }  
}

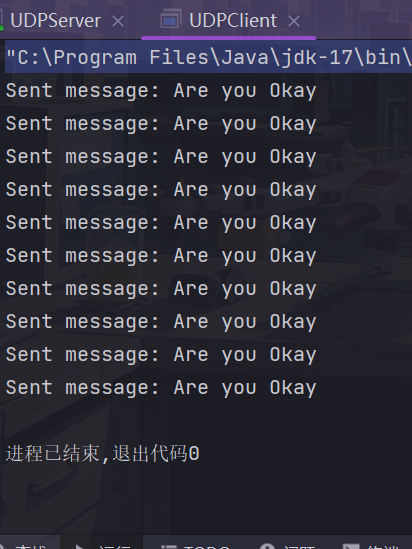
### Staff.java

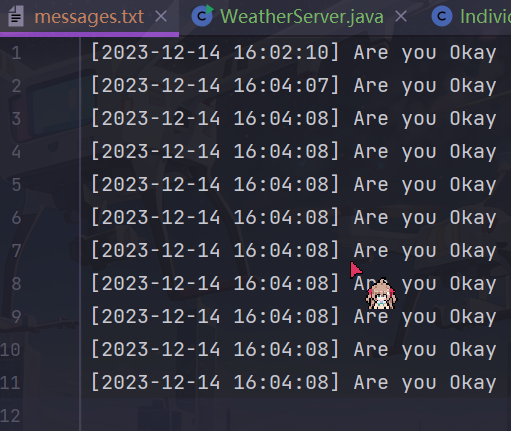
package school.finale**;**public class Staff extends Worker {  
 private String jobTitle**;** public Staff(String name**,** String address**,** String phoneNumber**,** String emailAddress**,** String officeLocation**,** double payRate**,** String jobTitle) {  
 super(name**,** address**,** phoneNumber**,** emailAddress**,** officeLocation**,** payRate)**;** this.jobTitle = jobTitle**;** }  
  
 @Override  
 public String toString() {  
 return super.toString() + " - Job Title: " + jobTitle**;** }  
}

# 题目2

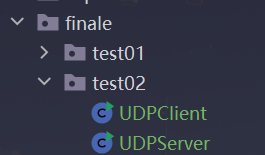
## 输出结果截图







## 工程项目截图



## 源代码

### UDPClient.java

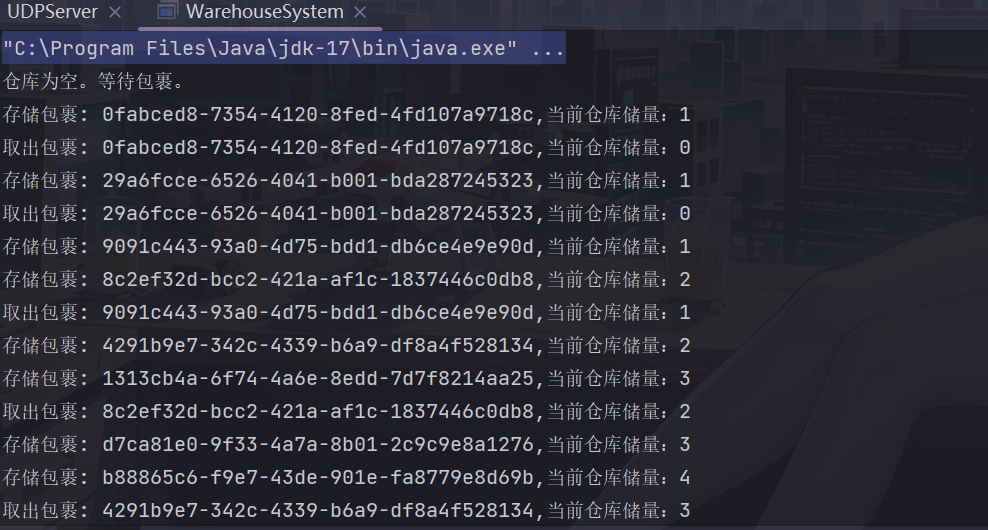
package school.finale.test02**;**import java.net.DatagramPacket**;**import java.net.DatagramSocket**;**import java.net.InetAddress**;**public class UDPClient {  
 public static void main(String[] args) {  
 try {  
 // 创建UDP套接字  
 DatagramSocket socket = new DatagramSocket()**;** // 服务器地址和端口  
 InetAddress serverAddress = InetAddress.*getLocalHost*()**;** int serverPort = 8001**;** // 发送消息  
 String message = "Are you Okay"**;** byte[] data = message.getBytes()**;** DatagramPacket packet = new DatagramPacket(data**,** data.length**,** serverAddress**,** serverPort)**;** // 每一秒发送一个包，发送10次  
 for (int i = 0**;** i < 10**;** i++) {  
 socket.send(packet)**;** System.*out*.println("Sent message: " + message)**;** Thread.*sleep*(100)**;** // 暂停0.1秒  
 }  
  
 // 关闭套接字  
 socket.close()**;** } catch (Exception e) {  
 e.printStackTrace()**;** }  
 }  
}

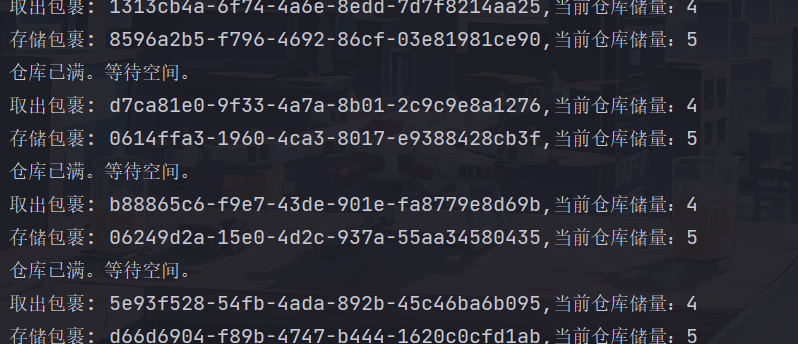
### UDPServer.java

package school.finale.test02**;**import java.io.FileWriter**;**import java.io.PrintWriter**;**import java.net.DatagramPacket**;**import java.net.DatagramSocket**;**import java.text.SimpleDateFormat**;**import java.util.Date**;**public class UDPServer {  
 public static void main(String[] args) {  
 try {  
 // 创建UDP套接字  
 DatagramSocket socket = new DatagramSocket(8001)**;** System.*out*.println("服务端正在运行...")**;** while (true) {  
 // 接收消息  
 byte[] buffer = new byte[1024]**;** DatagramPacket packet = new DatagramPacket(buffer**,** buffer.length)**;** socket.receive(packet)**;** // 创建新线程处理客户端  
 Thread clientHandler = new Thread(() -> *handleClient*(packet))**;** clientHandler.start()**;** }  
 } catch (Exception e) {  
 e.printStackTrace()**;** }  
 }  
  
 private static void handleClient(DatagramPacket packet) {  
 try {  
 // 解析消息  
 String message = new String(packet.getData()**,** 0**,** packet.getLength())**;** // 添加时间戳  
 String timestamp = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date())**;** String messageWithTimestamp = "[" + timestamp + "] " + message**;** // 同步写入文件  
 synchronized (UDPServer.class) {  
 try (PrintWriter writer = new PrintWriter(new FileWriter("messages.txt"**,** true))) {  
 writer.println(messageWithTimestamp)**;** }  
 }  
  
 System.*out*.println("收到消息：" + messageWithTimestamp)**;** } catch (Exception e) {  
 e.printStackTrace()**;** }  
 }  
}

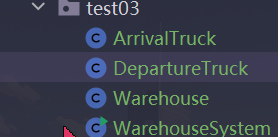
# 题目3

## 结果截图





## 工程项目截图



## 源代码

### Warehouse.java

package school.finale.test03**;**import java.util.LinkedList**;**import java.util.Queue**;**public class Warehouse {  
 private static final int *MAX\_PACKAGES* = 5**;** private Queue<String> storage = new LinkedList<>()**;** // 存储包裹的方法  
 public synchronized void storePackage(String packageId) {  
 try {  
 // 等待仓库不满  
 while (storage.size() == *MAX\_PACKAGES*) {  
 System.*out*.println("仓库已满。等待空间。")**;** wait()**;** }  
  
 // 存储包裹  
 storage.add(packageId)**;** System.*out*.println("存储包裹: " + packageId+",当前仓库储量："+storage.size())**;** // 通知可能等待的卸货卡车  
 notify()**;** } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt()**;** }  
 }  
  
 // 取出包裹的方法  
 public synchronized String retrievePackage() {  
 try {  
 // 等待仓库不空  
 while (storage.isEmpty()) {  
 System.*out*.println("仓库为空。等待包裹。")**;** wait()**;** }  
  
 // 取出包裹  
 String packageId = storage.poll()**;** System.*out*.println("取出包裹: " + packageId+",当前仓库储量："+storage.size())**;** // 通知可能等待的装货卡车  
 notify()**;** return packageId**;** } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt()**;** return null**;** }  
 }  
}

### ArrivalTruck.java

package school.finale.test03**;**import java.util.UUID**;**public class ArrivalTruck extends Thread {  
 private Warehouse warehouse**;** public ArrivalTruck(Warehouse warehouse) {  
 this.warehouse = warehouse**;** }  
  
 @Override  
 public void run() {  
 while (true) {  
 // 生成唯一UUID，模拟包裹  
 String packageId = UUID.*randomUUID*().toString()**;** // 存储包裹到仓库  
 warehouse.storePackage(packageId)**;** try {  
 // 模拟卡车到达的时间间隔  
 Thread.*sleep*(1000)**;** } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt()**;** }  
 }  
 }  
}

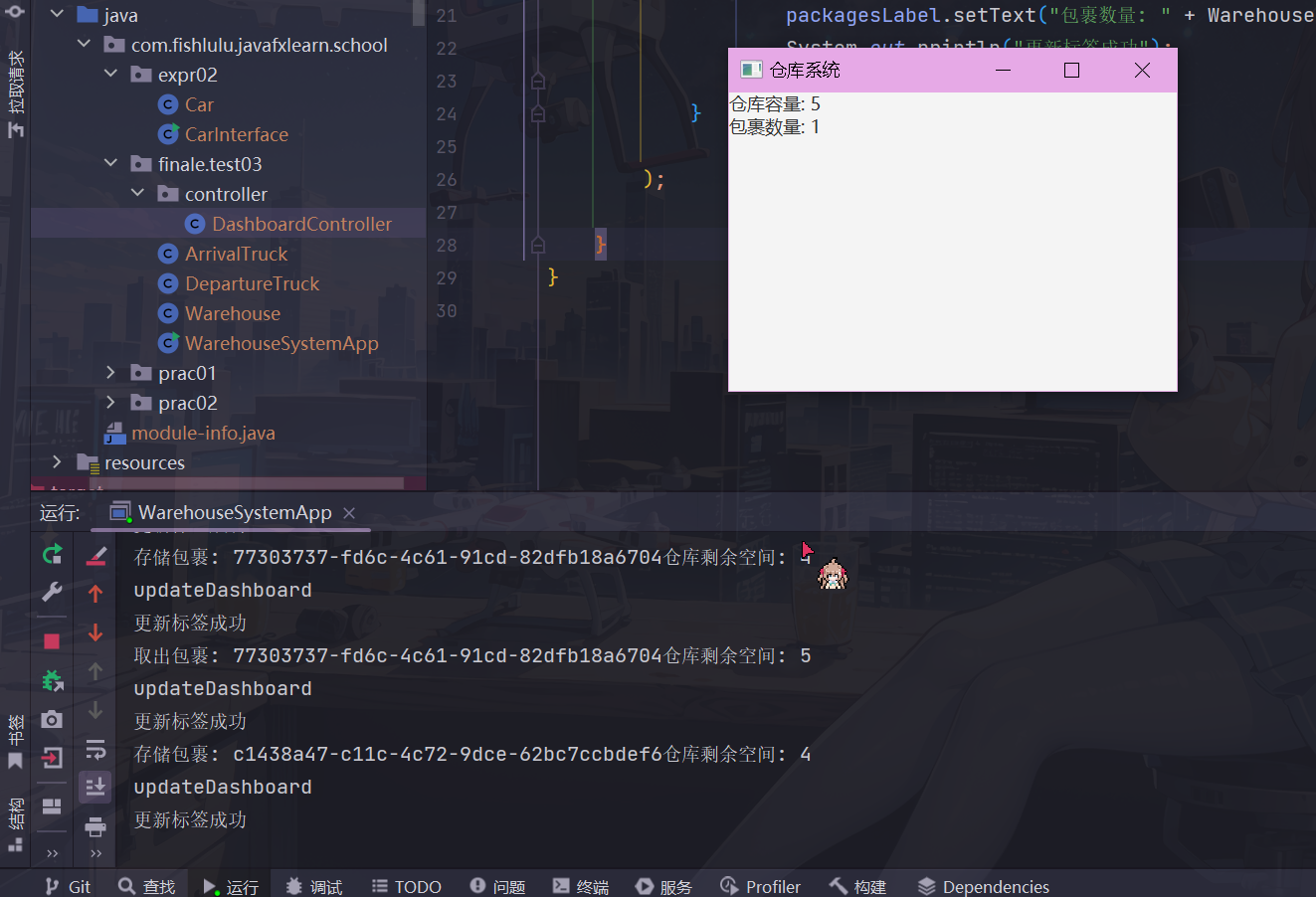
### DepatureTruck.java

package school.finale.test03**;**public class DepartureTruck extends Thread {  
 private Warehouse warehouse**;** public DepartureTruck(Warehouse warehouse) {  
 this.warehouse = warehouse**;** }  
  
 @Override  
 public void run() {  
 while (true) {  
 // 取出包裹并打印出货细节  
 String packageId = warehouse.retrievePackage()**;** try {  
 // 模拟卡车离开的时间间隔  
 Thread.*sleep*(2000)**;** } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt()**;** }  
 }  
 }  
}

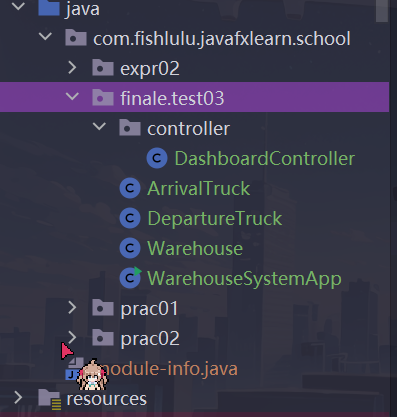
### WarehouseSystem.java

package school.finale.test03**;**public class WarehouseSystem {  
 public static void main(String[] args) {  
 Warehouse warehouse = new Warehouse()**;** ArrivalTruck arrivalTruck = new ArrivalTruck(warehouse)**;** DepartureTruck departureTruck = new DepartureTruck(warehouse)**;** arrivalTruck.start()**;** departureTruck.start()**;** }  
}

# 题目4（第四题第1问）



## 工程项目截图



## .源代码

### Warehouse.java

package com.fishlulu.javafxlearn.school.finale.test03**;**import java.util.LinkedList**;**import java.util.Queue**;**public class Warehouse {  
 private static final int *MAX\_PACKAGES* = 5**;** private Queue<String> storage = new LinkedList<>()**;** private static int *numPackages***;** public synchronized void storePackage(String packageId) {  
 try {  
 while (storage.size() == *MAX\_PACKAGES*) {  
 System.*out*.println("仓库已满。等待空间。")**;** wait()**;** }  
  
 storage.add(packageId)**;** *numPackages*++**;** System.*out*.println("存储包裹: " + packageId+"仓库剩余空间: " + (*MAX\_PACKAGES* - *numPackages*))**;** notify()**;** } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt()**;** }  
 }  
  
 public synchronized String retrievePackage() {  
 try {  
 while (storage.isEmpty()) {  
 System.*out*.println("仓库为空。等待包裹。")**;** wait()**;** }  
  
 String packageId = storage.poll()**;** *numPackages*--**;** System.*out*.println("取出包裹: " + packageId+"仓库剩余空间: " + (*MAX\_PACKAGES* - *numPackages*))**;** notify()**;** return packageId**;** } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt()**;** return null**;** }  
 }  
  
 public static int getMaxCapacity() {  
 return *MAX\_PACKAGES***;** }  
  
 public static int getNumPackages() {  
 return *numPackages***;** }  
}

### ArrivalTruck.java

package com.fishlulu.javafxlearn.school.finale.test03**;**import java.util.UUID**;**public class ArrivalTruck extends Thread {  
 private Warehouse warehouse**;** private Runnable onUpdateDashboard**;** public ArrivalTruck(Warehouse warehouse) {  
 this.warehouse = warehouse**;** super.setName("ArrivalTruck")**;** }  
  
 // 设置更新仓库情况的回调方法  
 public void setOnUpdateDashboard(Runnable onUpdateDashboard) {  
 this.onUpdateDashboard = onUpdateDashboard**;** }  
  
 @Override  
 public void run() {  
 Thread.*currentThread*().setName("ArrivalTruckDashBoard")**;** while (true) {  
 String packageId = UUID.*randomUUID*().toString()**;** warehouse.storePackage(packageId)**;** // 通知 UI 更新  
 if (onUpdateDashboard != null) {  
 onUpdateDashboard.run()**;** }  
  
 try {  
 Thread.*sleep*(1000)**;** } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt()**;** }  
 }  
 }  
}

### DepartureTruck.java

package com.fishlulu.javafxlearn.school.finale.test03**;**public class DepartureTruck extends Thread {  
 private Warehouse warehouse**;** private Runnable onUpdateDashboard**;** public DepartureTruck(Warehouse warehouse) {  
 this.warehouse = warehouse**;** super.setName("DepartureTruck")**;** }  
  
 // 设置更新仓库情况的回调方法  
 public void setOnUpdateDashboard(Runnable onUpdateDashboard) {  
 this.onUpdateDashboard = onUpdateDashboard**;** }  
  
 @Override  
 public void run() {  
 Thread.*currentThread*().setName("DepartureTruckDashBoard")**;** while (true) {  
 warehouse.retrievePackage()**;** // 通知 UI 更新  
 if (onUpdateDashboard != null) {  
 onUpdateDashboard.run()**;** }  
 try {  
 Thread.*sleep*(1500)**;** } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt()**;** }  
 }  
 }  
}

### DashBoardController

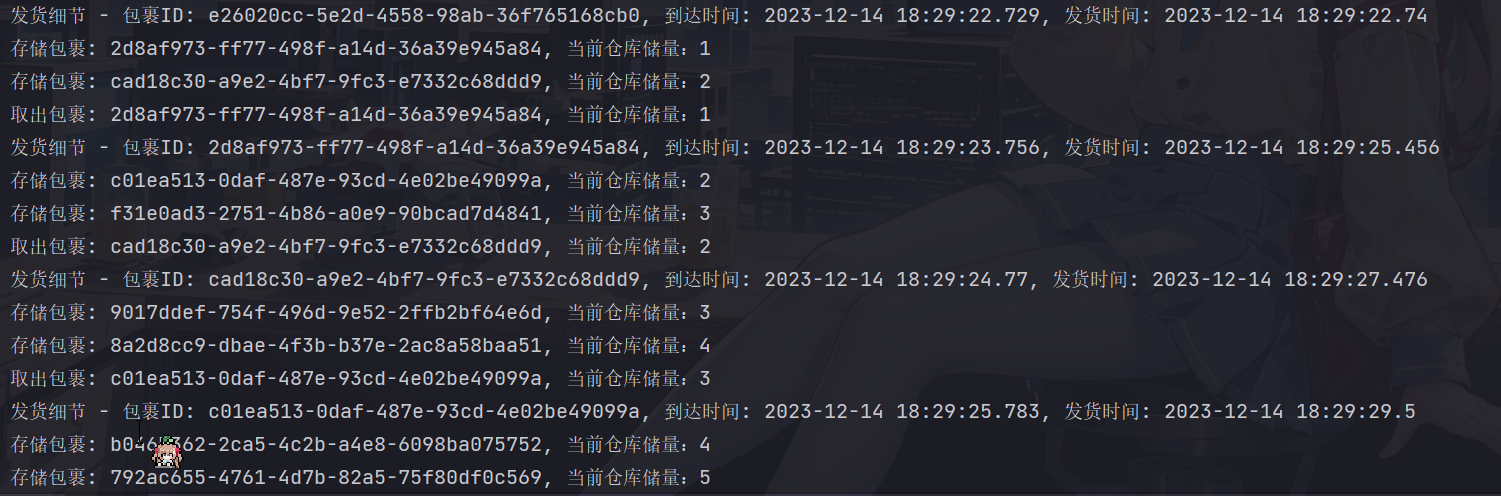
package com.fishlulu.javafxlearn.school.finale.test03.controller**;**import com.fishlulu.javafxlearn.school.finale.test03.Warehouse**;**import javafx.application.Platform**;**import javafx.scene.control.Label**;**public class DashboardController {  
 private Label capacityLabel**;** private Label packagesLabel**;** public void setLabels(Label capacityLabel**,** Label packagesLabel) {  
 this.capacityLabel = capacityLabel**;** this.packagesLabel = packagesLabel**;** }  
  
 public Runnable updateDashboard() {  
 return () -> Platform.*runLater*(()->{  
 System.*out*.println("updateDashboard")**;** if (capacityLabel != null && packagesLabel != null) {  
 capacityLabel.setText("仓库容量: " + Warehouse.*getMaxCapacity*())**;** packagesLabel.setText("包裹数量: " + Warehouse.*getNumPackages*())**;** System.*out*.println("更新标签成功")**;** }  
 }  
  
 )**;** }  
}

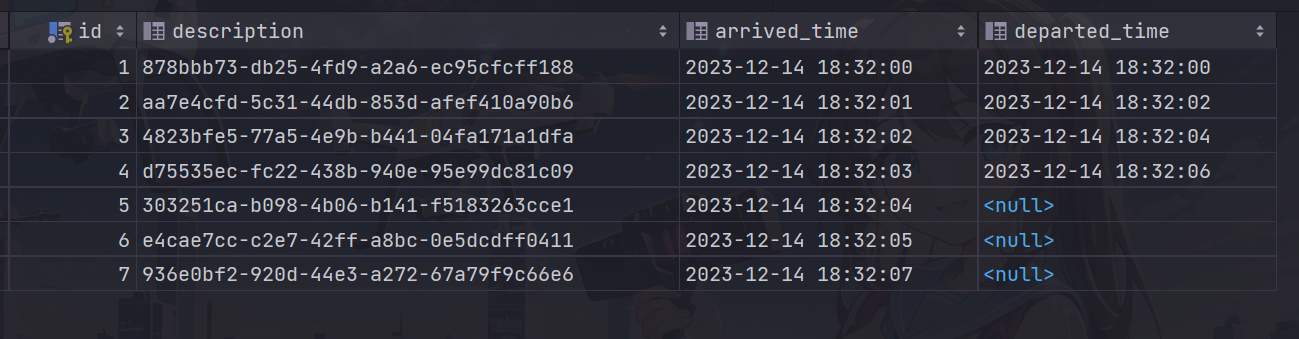
### WarehouseSystemApp

package com.fishlulu.javafxlearn.school.finale.test03**;**import com.fishlulu.javafxlearn.school.finale.test03.controller.DashboardController**;**import javafx.application.Application**;**import javafx.scene.Scene**;**import javafx.scene.canvas.Canvas**;**import javafx.scene.control.Label**;**import javafx.scene.layout.VBox**;**import javafx.stage.Stage**;**

**//一定要先在module-info文件设置导出,我因为这个吃了大亏T^T**public class WarehouseSystemApp extends Application {  
  
 public static void main(String[] args) {  
 *launch*(args)**;** }  
 @Override  
 public void start(Stage primaryStage) throws Exception {  
  
 Canvas canvas = new Canvas(400**,** 200)**;** // 创建 UI 元素  
 Label capacityLabel = new Label("仓库容量: ")**;** Label packagesLabel = new Label("包裹数量: ")**;** VBox root = new VBox(capacityLabel**,** packagesLabel)**;** // 创建 Warehouse 对象  
 Warehouse warehouse = new Warehouse()**;** // 创建 DashboardController 对象  
 DashboardController controller = new DashboardController()**;** controller.setLabels(capacityLabel**,** packagesLabel)**;** // 创建 ArrivalTruck 和 DepartureTruck 对象  
 ArrivalTruck arrivalTruck = new ArrivalTruck(warehouse)**;** DepartureTruck departureTruck = new DepartureTruck(warehouse)**;** // 设置更新 UI 的回调方法  
 arrivalTruck.setOnUpdateDashboard(controller.updateDashboard())**;** departureTruck.setOnUpdateDashboard(controller.updateDashboard())**;** // 启动卡车线程  
 arrivalTruck.start()**;** departureTruck.start()**;** // 创建场景并设置主Stage  
 Scene scene = new Scene(root**,** 300**,** 200)**;** primaryStage.setTitle("仓库系统")**;** primaryStage.setScene(scene)**;** primaryStage.show()**;** }  
  
  
}

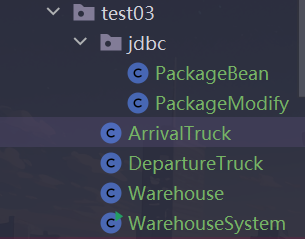
# 题目4.2（第四题第2问）





离开时间为null则是还没来得及出货

## 工程项目截图



## 源代码

### PackageBean.java

package school.finale.test03.jdbc**;**import java.sql.Timestamp**;**public class PackageBean {  
 // private static final String CREATE\_TABLE\_SQL = "CREATE TABLE IF NOT EXISTS package\_records ("  
 // + "id INT AUTO\_INCREMENT PRIMARY KEY,"  
 // + "description VARCHAR(255),"  
 // + "arrived\_time TIMESTAMP,"  
 // + "departed\_time TIMESTAMP"  
 // + ")";  
  
 private String description**;** private Timestamp arrivedTime**;** private Timestamp departedTime**;** public PackageBean() {  
 }  
 public PackageBean(String description**,** Timestamp arrivedTime**,** Timestamp departedTime) {  
 this.description = description**;** this.arrivedTime = arrivedTime**;** this.departedTime = departedTime**;** }  
  
 public String getDescription() {  
 return description**;** }  
  
 public void setDescription(String description) {  
 this.description = description**;** }  
  
 public Timestamp getArrivedTime() {  
 return arrivedTime**;** }  
  
 public void setArrivedTime(Timestamp arrivedTime) {  
 this.arrivedTime = arrivedTime**;** }  
  
 public Timestamp getDepartedTime() {  
 return departedTime**;** }  
  
 public void setDepartedTime(Timestamp departedTime) {  
 this.departedTime = departedTime**;** }  
}

### PackageModify.java -> jdbc执行类

package school.finale.test03.jdbc**;**import java.sql.Connection**;**import java.sql.DriverManager**;**import java.sql.PreparedStatement**;**import java.sql.SQLException**;**import java.sql.Timestamp**;**public class PackageModify {  
 // JDBC连接信息  
 private static final String *JDBC\_URL* = "jdbc:mysql://localhost:3306/dbtest2"**;** private static final String *USER* = "root"**;** private static final String *PASSWORD* = "123456"**;** // SQL语句插入数据  
 private static final String *INSERT\_PACKAGE\_SQL* = "INSERT INTO package\_records (description, arrived\_time, departed\_time) VALUES (?, ?, ?)"**;** // SQL语句更新离开时间  
 private static final String *UPDATE\_DEPARTED\_TIME\_SQL* = "UPDATE package\_records SET departed\_time = ? WHERE description = ?"**;** static {  
 // 加载JDBC驱动  
 try {  
 Class.*forName*("com.mysql.cj.jdbc.Driver")**;** } catch (ClassNotFoundException e) {  
 e.printStackTrace()**;** }  
  
 }  
  
  
 // 插入包裹记录  
 public static void insertPackage(PackageBean packageBean) {  
 try (Connection connection = DriverManager.*getConnection*(*JDBC\_URL***,** *USER***,** *PASSWORD*)**;** PreparedStatement preparedStatement = connection.prepareStatement(*INSERT\_PACKAGE\_SQL*)) {  
 preparedStatement.setString(1**,** packageBean.getDescription())**;** preparedStatement.setTimestamp(2**,** packageBean.getArrivedTime())**;** preparedStatement.setTimestamp(3**,** packageBean.getDepartedTime())**;** preparedStatement.executeUpdate()**;** } catch (SQLException e) {  
 e.printStackTrace()**;** }  
 }  
  
 // 更新包裹离开时间  
 public static void updateDepartedTime(String description**,** Timestamp departedTime) {  
 try (Connection connection = DriverManager.*getConnection*(*JDBC\_URL***,** *USER***,** *PASSWORD*)**;** PreparedStatement preparedStatement = connection.prepareStatement(*UPDATE\_DEPARTED\_TIME\_SQL*)) {  
 preparedStatement.setTimestamp(1**,** departedTime)**;** preparedStatement.setString(2**,** description)**;** preparedStatement.executeUpdate()**;** } catch (SQLException e) {  
 e.printStackTrace()**;** }  
 }  
}

### Warehouse.java

package school.finale.test03**;**import java.util.LinkedList**;**import java.util.Queue**;**import school.finale.test03.jdbc.PackageBean**;**public class Warehouse {  
 private static final int *MAX\_PACKAGES* = 5**;** private Queue<PackageBean> storage = new LinkedList<>()**;** // 存储包裹的方法  
 public synchronized void storePackage(PackageBean packageBean) {  
 try {  
 // 等待仓库不满  
 while (storage.size() == *MAX\_PACKAGES*) {  
 System.*out*.println("仓库已满。等待空间。")**;** wait()**;** }  
  
 // 存储包裹  
 storage.add(packageBean)**;** System.*out*.println("存储包裹: " + packageBean.getDescription() + ", 当前仓库储量：" + storage.size())**;** // 通知可能等待的卸货卡车  
 notify()**;** } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt()**;** }  
 }  
  
 // 取出包裹的方法  
 public synchronized PackageBean retrievePackage() {  
 try {  
 // 等待仓库不空  
 while (storage.isEmpty()) {  
 System.*out*.println("仓库为空。等待包裹。")**;** wait()**;** }  
  
 // 取出包裹  
 PackageBean packageBean = storage.poll()**;** System.*out*.println("取出包裹: " + packageBean.getDescription() + ", 当前仓库储量：" + storage.size())**;** // 通知可能等待的装货卡车  
 notify()**;** return packageBean**;** } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt()**;** return null**;** }  
 }  
}

### ArrivalTruck.java

package school.finale.test03**;**import java.sql.Timestamp**;**import java.util.UUID**;**import school.finale.test03.jdbc.PackageBean**;**import school.finale.test03.jdbc.PackageModify**;**public class ArrivalTruck extends Thread {  
 private Warehouse warehouse**;** public ArrivalTruck(Warehouse warehouse) {  
 this.warehouse = warehouse**;** }  
  
 @Override  
 public void run() {  
 while (true) {  
 // 生成唯一UUID，模拟包裹  
 String packageId = UUID.*randomUUID*().toString()**;** // 获取当前时间戳，表示包裹到达时间  
 Timestamp arrivedTime = new Timestamp(System.*currentTimeMillis*())**;** PackageBean packageBean = new PackageBean(packageId**,** arrivedTime**,** null)**;** // 存储包裹到仓库  
 warehouse.storePackage(packageBean)**;** //添加包裹到数据库  
 PackageModify.*insertPackage*(packageBean)**;** try {  
 // 模拟卡车到达的时间间隔  
 Thread.*sleep*(1000)**;** } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt()**;** }  
 }  
 }  
}

### DepatureTruck.java

package school.finale.test03**;**import java.sql.Timestamp**;**import school.finale.test03.jdbc.PackageBean**;**import school.finale.test03.jdbc.PackageModify**;**public class DepartureTruck extends Thread {  
 private Warehouse warehouse**;** public DepartureTruck(Warehouse warehouse) {  
 this.warehouse = warehouse**;** }  
  
 @Override  
 public void run() {  
 while (true) {  
 // 取出包裹并打印出货细节  
 PackageBean packageBean = warehouse.retrievePackage()**;** //设置发货时间  
 packageBean.setDepartedTime(new Timestamp(System.*currentTimeMillis*()))**;** if (packageBean != null) {  
 System.*out*.println("发货细节 - 包裹ID: " + packageBean.getDescription() +  
 ", 到达时间: " + packageBean.getArrivedTime() +  
 ", 发货时间: " + packageBean.getDepartedTime())**;** PackageModify.*updateDepartedTime*(packageBean.getDescription()**,** packageBean.getDepartedTime())**;** }  
  
 try {  
 // 模拟卡车离开的时间间隔  
 Thread.*sleep*(2000)**;** } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt()**;** }  
 }  
 }  
}

### WarehouseSystem.java

package school.finale.test03**;**public class WarehouseSystem {  
 public static void main(String[] args) {  
 Warehouse warehouse = new Warehouse()**;** ArrivalTruck arrivalTruck = new ArrivalTruck(warehouse)**;** DepartureTruck departureTruck = new DepartureTruck(warehouse)**;** arrivalTruck.start()**;** departureTruck.start()**;** }  
}