淮 海 工 学 院 计算机工程学院

课程设计报告

**设计名称：** 软件工程课程设计

**选题名称：**火车订票系统的设计与实现--车次查询模块

**姓 名：** 陈康 **学 号：** 201814058

**专业班级：** **软件工程 Z软件162**

**系 （院）： 应用技术学院**

**设计时间：** 2019.6.3~2019.6.23

**设计地点：**  **应用技术学院机房、教室**

|  |
| --- |
| 指导教师评语：  **成绩：**  签名：  年 月 日 |

|  |
| --- |
| 1．课程设计目的 |
| 软件工程课程设计是计算机专业一个综合性的实践教学环节，其目的在于促进学生复习和巩固计算机软件设计知识，加深对软件设计方法、软件设计技术和设计思想的理解，并能运用所学软件设计知识和面向对象技术进行综合软件设计，通过本课程设计能够进行简单软件系统的开发，掌握软件设计的方法和面向对象程序设计的基本技术，提高学生的综合应用能力。 |
| 2．课程设计任务与要求： |
| **任务**  结合《软件工程》、《面向对象程序设计》课程以及相关课程中所学知识，积极完成设计任务。要求通过设计，深对课程基本内容的理解和综合运用。  学生自选课题  学生原则上可以结合个人爱好自选课题，要求课题有一定的深度与难度，有一定的算法复杂性，能够巩固数据结构课程所学的知识。学生自选课题需在16周前报课程设计指导教师批准方可生效。  **要求：**  （1）通过文献资料查阅和学习，了解当前软件设计技术和一般方法。  （2）参考和研究一些公司和高校/企业成功的软件开发案例和实现方案，结合《面向对象程序设计》、《软件工程》等课程中所学知识，积极完成设计任务。  （3）认真完成需求分析，并根据需求分析完成各设计题目的总体设计、详细设计和测设等环节的设计任务，开发工具推荐使用Asp.net | Vc++ | Jsp.net 。  （4）每位同学需提交可独立运行的软件程序。  （5）认真按时完成课程设计报告，课程设计报告内容包括：课程设计目的、设计任务与要求、需求分析、概要设计、详细设计、调试分析、测试结果、附录和设计心得体会等。  （6）每位同学需独立提交设计报告书（每人一份），要求编排格式统一、规范、内容充实，不少于10页（代码不算）。 |
| 3．课程设计说明书 |
| **一 需求分析**  **1.1 需求**  火车订票系统的主要功能是向旅客提供一种BS模式的订票服务。用户需要预定车票时登陆该系统后可以查询关心的车次信息，同时可以预定该车次的车票。在提供订票功能的同时，还应该提供改签和退票等功能。  本次课程设计主要针对系统中的车次信息查询模块。车次查询模块需要实现的需求入下所示。  用户进入模块后，输入起点、终点、出发时间后页面显示出发时间后24小时内所有从起点到终点的车次列表，列表中的每一行包含一个车次的信息，用户可以下载车次列表。车次信息包括车次号、出发时间、到达时间、列车类型、票价、座位余量等信息。用户再点击某一车次后显示该车次的详细信息，除上述信息外，还应显示该车次的全部途径车站。  **1.2 数据流图**  表1-1 数据流图元素   |  |  | | --- | --- | | 源点\终点 | 处理 | | 源点：旅客  终点：旅客 | 生成车次列表  生成车次详细信息 | | 数据流 | 数据存储 | | 车次查询信息：  起点  终点  出发时间  车次列表：  车次号  出发时间  到达时间  列车类型  票价  座位余量  车次信息：  车次号  出发时间  到达时间  列车类型  首发站  终点站  停靠站  各站出发时间  各站到达时间 | 车次列表  （见车次列表） |     图1-1 车次查询模块数据流图  **1.3 数据字典**  表1-2 车次查询模块数据字典   |  | | --- | | 名称：车次查询信息  别名：车次查询条件  描述：旅客用户输入的查询信息  定义：车次查询信息=起点+终点+出发时间  位置：发送给车次查询接口 | | 名称：车次列表  别名：无  描述：符合用户输入的车次查询信息的所有车次的集合  定义：车次列表 = 车次号+出发时间+到达时间+列车类型+票价+座位余量  位置：下载到用户本地 | | 名称：查询车次  别名：无  描述：接收车次查询信息，取出符合条件的所有车次。 | | 名称：车次详细信息  别名：车次信息  描述：某一车次的所有公开信息  定义：车次详细信息 = 车次号+出发时间+到达时间+列车类型+首发站+终点站+停靠站+各站出发时间+各站到达时间 | | 名称：查询车次信息  别名：无  描述：接收某车次的车次号，取出该车次的所有信息 |   **1.4 E-R图**    图1-2 车次查询模块E-R图  **1.5 数据表设计**  由图1-2可知，该系统的主要实体有站点、停靠站、车次、列车。所以针对每个实体设计数据表。分别为station、stopStation、route、train。  表2-1 station数据表   |  |  |  |  | | --- | --- | --- | --- | | 字段名 | 数据类型 | 属性 | 备注 | | city | Varchar(20) | 不可为空 | 所在城市 | | id | int | 自增主键 | 站点编号 | | name | Varchar(20) | 不可为空 | 站点名 |   表2-2 stopStation数据表   |  |  |  |  | | --- | --- | --- | --- | | 字段名 | 数据类型 | 属性 | 备注 | | number | int | 不可为空 | 停靠的顺序 | | price | double | 不可为空 | 从首发站到本站的价格 | | id | int | 自增主键 | 停靠站编号 | | StationId | int | 不可为空 | 站点编号 | | routeId | int | 不可为空 | 车次编号 | | arrTime | time | 不可为空 | 到达时间 | | leaTime | time | 不可为空 | 离开时间 |   表2-3 route数据表   |  |  |  |  | | --- | --- | --- | --- | | 字段名 | 数据类型 | 属性 | 备注 | | id | int | 自增主键 | 车次号 | | startStation | int | 不可为空 | 始发站编号 | | endStation | int | 不可为空 | 终点站编号 | | seatOne | int | 默认为0 | 已售一等座数 | | seatTwo | int | 默认为0 | 已售二等座数 | | seatBuss | int | 默认为0 | 已售商务座数 | | time | datetime | 不可为空 | 发车时间 |   表2-4 train数据表   |  |  |  |  | | --- | --- | --- | --- | | 字段名 | 数据类型 | 属性 | 备注 | | seatOne | int | 默认为0 | 一等座数 | | seatTwo | int | 默认为0 | 二等座数 | | seatBuss | int | 默认为0 | 商务座数 | | id | int | 自增主键 | 列车编号 | | type | Varchar(20) | 不可为空 | 列车类型 |   **1.6 小组分工**  表1-3 小组分工   |  |  | | --- | --- | | 小组人员 | 分工 | | 陈康 | 车次查询模块 | | 谢运 | 订票模块 | | 高祥如 | 注册登入模块 |   **二 概要设计**    图2-1 车次查询模块HIPO图  **2.1 功能需求**  1、查询车次列表-查询功能  后台接收用户提交的查询参数，请求数据库中符合参数的数据段。   1. 查询车次列表-列表化   后台接收数据库中返回的数据段，并将数据段格式化为列表类型。并使用   1. 查询车次列表-显示   前端获得列表类型的数据后将数据显示在页面上。   1. 查询车次列表-排序   前端按照用户需要将列表类型的数据按一定顺序显示在页面上。   1. 查询车次信息-下载车次列表   用户点击下载按钮后将显示的车次列表作为文件下载至用户机器上。   1. 查询车次信息-下载车次列表-生成文件   后台将列表写入文件。   1. 查询车次信息-下载车次列表-传输文件   将后台写好的文件传输给用户。  8、查询车次详细信息-查询  后台接收用户选择的车次的车次号，请求数据库中该车次号的详细信息。  9、查询车次详细信息-显示  前台获得某车次的详细信息后将信息显示在页面上。  **三 详细设计**  由于车次查询模块的主要功能是满足旅客用户查询某一时刻起点至终点的所有行车安排的需求。所以该模块最终要的功能是在尽可能短的时间内向用户展示满足其需求的所有车次列表。在满足这一功能的前提下，提供下载列表、排序列表、查看车次详细信息等功能。  综上所述，车次查询模块分为两个字模块。分别是查询车次列表和查询车次详细信息。其中，查询车次列表模块中又包含下载车次列表模块。所以，车次查询模块共有三个子模块。分别是查询车次列表模块、下载车次列表模块、查询车次详细信息模块。下面将展示针对这三个模块的详细设计思路。  **3.1 查询车次列表**  查询车次列表模块的主要作用是获取用户提交的查询信息后，从数据库中请求所有符合用户查询信息的数据，并将数据进行处理后利用JSP或ajax技术显示在页面上。同时，针对用户对查询结果的排序请求显示相应的数据。具体的程序流程图和伪代码如下所示。  **3.1.1 查询车次列表模块流程图**    图3-1 查询车次列表模块流程图  **3.1.2 查询车次列表模块伪代码**  查询车次列表 seq  获取用户输入的起点、终点、出发时间 while 用户输入的信息合法  处理用户输入的信息 select 信息合法  处理合法信息 seq  获取满足用户输入信息的车次数据  处理车次数据 while 获取不到车次数据  获取一条车次数据 select 获取成功  输出一行车次信息 seq  输出车次号  输出发车时间  输出到达时间  输出票价  输出列车类型  输出票价  输出座位余量  输出换行符  输出一行车次信息 end  获取一条车次数据 or 获取失败  获取一条车次数据 end  处理车次数据 end  处理合法信息 end  处理用户输入的信息 or 信息不合法  处理不合法信息 seq  提示用户输入信息不合法  处理不合法信息 end  处理用户输入的信息 end  获取用户输入的信息 end  查询车次列表 end  **3.2 下载车次列表模块**  下载车次列表模块的主要作用是向用户提供格式化的车次列表文件下载功能。用户在查询到需要的车次信息后可以勾选其中的任意条车次信息进行下载。下载后车次信息以excel表格的形式存放在用户的设备中。  **3.2.1 下载车次列表模块流程图**    图3-2 车次列表下载模块流程图  **3.2.2 下载车次列表模块伪代码**  下载车次列表 seq  获取用户需要下载的车次号数组  获取车次信息 seq while 车次号全部取出  取出一个车次号  查询该车次号的信息  将车次信息放入车次信息数组中  获取车次信息 end  生成excel seq  调用API生成excel空文件  处理车次信息 while 车次信息全部处理完毕  取出一条车次信息  生成excel行  写入车次信息 while 车次信息读取完毕  读取车次信息中的一条数据  生成excel单元格  将数据写入单元格  将单元格撷取excel行  写入车次信息 end  将excel行写入excel文件  处理车次信息 end  生成excel end  将excel文件以二进制流的形式返回给用户  下载车次列表 end  **3.3 查询车次详细信息模块**  查询车次详细信息模块的主要作用是补充显示车次列表中不方便显示的车次信息。用户在查询车次列表的结果页面点击某一车次后进入该车次的详细信息页面。详细信息页面包括该车次的车次号、首发站、终点站、已经中途停靠站。同时显示各站点的到达时刻以及离开时刻和停靠时间。同时显示该车次在用户查询时输入的起点终点区间中的座位剩余情况。  **3.3.1 查询车次详细信息模块流程图**    图3-3 查询车次详细信息模块流程图  **3.3.2 查询车次详细信息模块伪代码**  查询车次详细信息 seq  获取用户点击的车次号  请求数据库获得该车次的详细信息  使用JSP技术将详细信息显示在页面上  将页面返回给用户的浏览器  查询车次详细信息 end  **四 设计与调试分析**  由于设计问题，车次查询列表页面的订票功能不够美观，操作不够简便。如图4-1所示。订票的按钮过多导致页面过于杂乱。经修改后将每个车次的订票按钮放置在最后，同时提供座位单选按钮。如图4-2所示    图4-1 修改前页面  IMG_256图4-2 修改后页面  **源代码**  </tr>  <%  request.setCharacterEncoding("UTF-8");  String start = request.getParameter("start");  String end = request.getParameter("end");  String time = request.getParameter("year")+"-"+request.getParameter("month")+"-"+request.getParameter("day");  System.out.println(start +"->"+ end +" | "+time);  dbRouteList db = new dbRouteList(start,end,time);  List<route> routes = db.getRouteList();  for(int i=0;i<routes.size();i++){  out.println("<tr>");  out.println("<td>"+routes.get(i).id);  out.println("<td>"+start);  out.println("<td>"+routes.get(i).leatime);  out.println("<td>"+end);  out.println("<td>"+routes.get(i).arrTime);  out.println("<td>"+routes.get(i).seat[1]);  // out.println("<td>");  // out.println("<form action=\"bookTicket\" method=\"post\">");  // out.println("<input type=\"hidden\" name=\"id\" value=\""+routes.get(i).id+"\">");  // out.println("<input type=\"hidden\" name=\"date\" value=\""+time+"\">");  // out.println("<input type=\"hidden\" name=\"start\" value=\""+start+"\">");  // out.println("<input type=\"hidden\" name=\"end\" value=\""+end+"\">");  // out.println("<input type=\"hidden\" name=\"seat\" value=\"seatTwo\">");  // out.println("<input type=\"submit\" name=\"id\" value=\"订票\">");  // out.println("</from>");  out.println("<td>"+routes.get(i).seat[0]);  // out.println("<td>");  // out.println("<form action=\"bookTicket\" method=\"post\">");  // out.println("<input type=\"hidden\" name=\"id\" value=\""+routes.get(i).id+"\">");  // out.println("<input type=\"hidden\" name=\"date\" value=\""+time+"\">");  // out.println("<input type=\"hidden\" name=\"start\" value=\""+start+"\">");  // out.println("<input type=\"hidden\" name=\"end\" value=\""+end+"\">");  // out.println("<input type=\"hidden\" name=\"seat\" value=\"seatOne\">");  // out.println("<input type=\"submit\" name=\"id\" value=\"订票\">");  // out.println("</from>");  out.println("<td>"+routes.get(i).seat[2]);  // out.println("<td>");  // out.println("<form action=\"bookTicket\" method=\"post\">");  // out.println("<input type=\"hidden\" name=\"id\" value=\""+routes.get(i).id+"\">");  // out.println("<input type=\"hidden\" name=\"date\" value=\""+time+"\">");  // out.println("<input type=\"hidden\" name=\"start\" value=\""+start+"\">");  // out.println("<input type=\"hidden\" name=\"end\" value=\""+end+"\">");  // out.println("<input type=\"hidden\" name=\"seat\" value=\"seatBuss\">");  // out.println("<input type=\"submit\" name=\"id\" value=\"订票\">");  // out.println("</from>");  out.println("<td>"+routes.get(i).type);  out.println("<td>"+routes.get(i).price);  out.println("<td><a href=\"routeInfo.jsp?id="+routes.get(i).id+"\">查看详情</a>");  out.println("<td>");  out.println("<form action=\"bookTicket\" method=\"post\">");  out.println("<input type=\"hidden\" name=\"id\" value=\""+routes.get(i).id+"\">");  out.println("<input type=\"hidden\" name=\"date\" value=\""+time+"\">");  out.println("<input type=\"hidden\" name=\"start\" value=\""+start+"\">");  out.println("<input type=\"hidden\" name=\"end\" value=\""+end+"\">");  out.println("<input type=\"radio\" name=\"seat\" value=\"seatTwo\">二等座");  out.println("<input type=\"radio\" name=\"seat\" value=\"seatOne\">一等座");  out.println("<input type=\"radio\" name=\"seat\" value=\"seatBuss\">商务座");  out.println("<input type=\"submit\" name=\"id\" value=\"订票\">");  out.println("</from>");  }  %>  **五 用户手册**  用户登陆后，进入本模块的车次查询页面。再车次查询页面上输入起点和终点以及出发时间后点击确定按钮。页面跳转至车次列表显示页面。再车次列表显示页面可以看到所有符合条件的车次及其信息。在车次信息的坐席数量后可以点击预定按钮。进入预定车次模块。在车次信息的末尾可以点击详细信息按钮，进入该车次的详细信息页面。详细信息页面包括该车次的所有途径站点及个站点的时刻信息。  **六 测试成果**  经过设计，本系统的车次查询界面测试的界面如图1所示。    图1 车次查询入口界面’  IMG_256经过设计，本系统的车次列表显示页面测试的界面如图2所示。  图2 车次信息列表页面  经过设计，本系统的车次详细信息页面的测试如图3所示。    图3 车次详细信息页面  **七 附录（源程序清单）**  **routeInfo.jsp**  <%  dbRouteInfo db = new dbRouteInfo(request.getParameter("id"));  List<station> stations = db.getStations();  for(int i=0;i<stations.size();i++){  out.println("<tr>");  out.println(String.format("<td>%s</td>",i+1));  out.println(String.format("<td>%s</td>",stations.get(i).name));  out.println(String.format("<td>%s</td>",stations.get(i).arrTime));  out.println(String.format("<td>%s</td>",stations.get(i).leaTime));  out.println(String.format("<td>%s</td>",stations.get(i).price));  } %>  **searchRoute.jsp**  dbRouteInfo.class <%  request.setCharacterEncoding("UTF-8");  String start = request.getParameter("start");  String end = request.getParameter("end");  String time = request.getParameter("year")+"-"+request.getParameter("month")+"-"+request.getParameter("day");  System.out.println(start +"->"+ end +" | "+time);  dbRouteList db = new dbRouteList(start,end,time);  List<route> routes = db.getRouteList();  for(int i=0;i<routes.size();i++){  out.println("<tr>");  out.println("<td>"+routes.get(i).id);  out.println("<td>"+start);  out.println("<td>"+routes.get(i).leatime);  out.println("<td>"+end);  out.println("<td>"+routes.get(i).arrTime);  out.println("<td>"+routes.get(i).seat[1]);  out.println("<td>"+routes.get(i).seat[0]);  out.println("<td>"+routes.get(i).type);  out.println("<td>"+routes.get(i).price);  out.println("<td><a href=\"routeInfo.jsp?id="+routes.get(i).id+"\">查看详情</a>");  out.println("<td>");  out.println("<form action=\"bookTicket\" method=\"post\">");  out.println("<input type=\"hidden\" name=\"id\" value=\""+routes.get(i).id+"\">");  out.println("<input type=\"hidden\" name=\"date\" value=\""+time+"\">");  out.println("<input type=\"hidden\" name=\"start\" value=\""+start+"\">");  out.println("<input type=\"hidden\" name=\"end\" value=\""+end+"\">");  out.println("<input type=\"radio\" name=\"seat\" value=\"seatTwo\">二等座");  out.println("<input type=\"radio\" name=\"seat\" value=\"seatOne\">一等座");  out.println("<input type=\"radio\" name=\"seat\" value=\"seatBuss\">商务座");  out.println("<input type=\"submit\" name=\"id\" value=\"订票\">");  out.println("</from>");  }  %>  public List<station> getStations(){  List<station> stations = new LinkedList<station>();  String sql = String.format("select \* from stopStation join station where routeId =\"%s\" and stationId = station.id order by number",id);  try {  this.init();  res = stmt.executeQuery(sql);  while (res.next()){  station s = new station();  s.id = res.getString("routeId");  s.arrTime = res.getString("arrTime");  s.leaTime = res.getString("leaTime");  s.price = res.getDouble("price");  s.name = res.getString("name");  stations.add(s);  }  } catch (SQLException e){  e.printStackTrace();  } finally {  this.destory();  }  for(int i=1;i<stations.size();i++){  stations.get(i).price -= stations.get(i-1).price;  }  return stations; }  **dbRouteList.class**  public List<route> getRouteList(){  List<String> ids = this.getRouteId(start,end);  return getRouteInfo(ids);  }   private List<route> getRouteInfo(List<String> ids){  List<route> routes = new LinkedList<route>();  for(int i=0;i<ids.size();i++){  route line = new route();  line.id = ids.get(i);  try{  this.init();  String sql = String.format("select \* from stopStation full join station where routeId = \"%s\" and station.name = \"%s\" and stationId = station.id;",ids.get(i),start);  res = stmt.executeQuery(sql);  System.out.println(sql);  while(res.next()){  line.leatime = res.getString("leaTime");  line.price = res.getDouble("price");  }  res = stmt.executeQuery(String.format("select \* from stopStation full join station where routeId = \"%s\" and station.name = \"%s\" and stationId = station.id;",ids.get(i),end));  while(res.next()){  line.arrTime = res.getString("arrTime");  line.price = res.getDouble("price")-line.price;  }  res = stmt.executeQuery(String.format("select train.seatOne,train.seatTwo,train.seatBuss,route.seatOne,route.seatTwo,route.seatBuss,route.type from train join route where route.number = \"%s\" AND train.type = route.type and date = \"%s\"",ids.get(i),time));  while(res.next()){  line.seat[0] = res.getInt(1)-res.getInt(4);  line.seat[1] = res.getInt(2)-res.getInt(5);  line.seat[2] = res.getInt(3)-res.getInt(6);  line.type = res.getString(7);  } // line.printInfo();  } catch (SQLException e){  e.printStackTrace();  } finally {  this.destory();  }  routes.add(line);  }  return routes;  }   private List<String> getRouteId(String s\_city,String e\_city){  String getStart = String.format("select routeId,number from stopStation full join station where station.name = \"%s\" AND StationId = station.id",s\_city);  String getEnd = String.format("select routeId,number from stopStation full join station where station.name = \"%s\" AND StationId = station.id",e\_city); // System.out.println(getStart); // System.out.println(getEnd);  List<String> ids = new LinkedList<String>();  try{  this.init();  Map<String,String> start = new HashMap<String,String>();  res = stmt.executeQuery(getStart);  while(res.next()){  start.put(res.getString(1),res.getString(2));  }  Map<String,String> end = new HashMap<String,String>();  res = stmt.executeQuery(getEnd);  while(res.next()){  end.put(res.getString(1),res.getString(2));  }  for(String startId:start.keySet()){ // System.out.println(startId+":"+start.get(startId));  for(String endId:end.keySet()){ // System.out.println(endId+":"+end.get(endId));  if(startId.equals(endId)){  if(Integer.parseInt(start.get(startId)) < Integer.parseInt(end.get(endId))){ // System.out.println(startId);  ids.add(startId);  }  }  }  }  } catch (SQLException e){  e.printStackTrace();  } finally {  this.destory();  }  return ids;  } |
| 1. 课程设计心得 |
| 每个项目的完成都会存在这样那样的缺陷， 包括设计上的缺陷和业务逻辑上的  不足。 例如： 查询模块提供的查询模式过少， 如果提供更多种查询方式， 则可以  更方面人们的查询， 可以增加日期查询， 时间点查询， 班次的种类删选， 团体类  型查询， 代售点查询等。  在项目要进行自由分组后， 我们的项目小组便诞生了。 我们小组由 3 个成员组  成， 在相互商量后我们也确定了 我们组的项目， 是做一个火车票预售系统。 我们  也随即做了 分工。 我们的项目也正式开始了 。 需求调研和分析对于软件开发过程  至关重要。 我们在开发时如果不进行调研和分析， 那么对于后来的项目进展将产  生致命的后果。 我们在项目的开发中便遇到了这样的问题。  在这段时间里， 我还对软件工程有了新的理解。 在我以前的理解当中， 软件工程，  无非就是一个人或者几个人或一个团队集中在一起进行编写代码的工作， 以实现  开发出所用的软件。 但现在我明白了， 软件工程的作用， 就是告诉人们怎样去开  发软件和管理软件。 具体地讲， 它表现在与软件开发和管理有关的人员和过程上。  所以， 软件工程就不仅仅是单一的编程过程了。 它包括了 系统分析->建模->概要  设计->详细设计->编码->测试->维护。 编码可以理解为编程， 这个只占总时间的  20％左右。 编程只是其中的一小部分。  通过这次课设使我们提升了合作精神， 由于大家负责的模块不一样， 所以需要  很好的沟通， 个模块之间独立但又有某种联系， 这就要求我们需要很好的熟悉项  目的需求。 我们通过网上查询资料， 翻阅课本， 向同学和老师请教， 最终完成了  这份报告。 在这过程中使我们自学的能力进一步提升， 也让我们初步了 解了企业  项目开发的流程， 对我们以后参加工作奠定了基础。 知识上的收获重要， 精神上  的丰收更加可喜。 挫折是一份财富， 经历是一份拥有。 这次实习必将成为我人生  旅途上一个非常美好的回忆！ |