

---

# Journey Management System

---

## Final Report

---

Group 96

Group Members:

Yiqiong MAI	130804671
Zheng LIU	130804936
Sijia YANG	130804969
Xiaoqian HUANG	130804992
Guangyuan ZHAO	130806848
Junjie LU	130805081

---

# CONTENT

1.	<b><u>INTRODUCTION</u></b> .....	- 3 -
2.	<b><u>SUMMARY OF RESPONSIBILITIES AND ACHIEVEMENTS</u></b> .....	- 3 -
2.1	GROUP MEMBERS' CONTRIBUTION .....	- 3 -
2.2	INDIVIDUAL MEMBER CONTRIBUTION AND SELF-APPRAISAL.....	- 4 -
2.2.1	Yiqiong MAI_ 130806848_ JP2013213431.....	- 4 -
2.2.2	Zheng LIU_ 130804936_ JP2013213501 .....	- 4 -
2.2.3	Sijia YANG_ 130804969_ JP2013213504 .....	- 5 -
2.2.4	Xiaoqian HUANG_ 130804992_ JP2013213507.....	- 5 -
2.2.5	Guangyuan ZHAO_ 130806848_ JP2013213431.....	- 6 -
2.2.6	Junjie LU_ 130805081_ JP2013213515.....	- 6 -
3.	<b><u>PROJECT MANAGEMENT</u></b> .....	- 7 -
3.1	INTRODUCTION .....	- 7 -
3.2	MANAGEMENT METHOD: SCRUM.....	- 7 -
3.2.1	Outline Planning .....	- 7 -
3.2.2	Sprint Cycles.....	- 7 -
3.2.3	Closure.....	- 8 -
3.2.4	Teamwork management and organization .....	- 8 -
3.3	PROJECT PLANNING .....	- 8 -
3.3.1	Resource requirements.....	- 8 -
3.3.2	Activity Organization .....	- 9 -
3.3.2.1	Project schedule .....	- 9 -
3.3.2.2	Chart.....	- 9 -
3.3.2.3	Activity Breakdown.....	- 9 -
3.4	ADAPTING TO CHANGE .....	- 10 -
3.5	GROUP PHOTO.....	- 10 -
	.....	- 10 -
4	<b><u>REQUIREMENTS</u></b> .....	- 11 -
4.1	FACT-FINDING TECHNIQUES .....	- 11 -
4.1.1	Background Reading.....	- 11 -
4.1.2	Observation.....	- 11 -
4.1.3	Interviewing.....	- 11 -
4.2	DETAILED FOUND REQUIREMENTS .....	- 12 -
4.2.1	Functional Requirement.....	- 12 -
4.2.2	Non-functional Requirements .....	- 12 -
4.3	CHANGES OF PRODUCT BACKLOG.....	- 12 -

4.4	SOFTWARE PROTOTYPES .....	- 13 -
4.4.1	<i>Logical user-interface design</i> .....	- 13 -
4.4.2	<i>Physical user-interface design</i> .....	- 13 -
4.5	ITERATION AND ESTIMATION OF THE STORIES.....	- 15 -
5	<b><u>ANALYSIS AND DESIGN</u></b> .....	- 16 -
5.1	ANALYSIS AND ARCHITECTURE DESIGN .....	- 16 -
5.2	CLASS RELATIONSHIP .....	- 17 -
5.3	RE-USABILITY OF SYSTEM COMPONENTS .....	- 17 -
5.4	CLASS DIAGRAM .....	- 18 -
6	<b><u>IMPLEMENTATION AND TESTING</u></b> .....	- 19 -
6.1	IMPLEMENTATION.....	- 19 -
6.1.1	<i>Assumption</i> .....	- 19 -
6.1.2	<i>Implement Strategy</i> .....	- 19 -
6.1.2.1	Integration Build Plan .....	- 19 -
6.1.2.2	Association between classes.....	- 20 -
6.1.2.3	Mapping between design and code .....	- 20 -
6.1.2.4	Realized principles .....	- 20 -
6.1.3	<i>Pair Programming Analysis</i> .....	- 20 -
6.2	TESTING.....	- 20 -
6.2.1	<i>Test Strategies</i> .....	- 21 -
6.2.1.1	Testing cases.....	- 21 -
6.2.1.2	Testing process .....	- 21 -
6.2.1.3	Success criteria.....	- 21 -
6.2.2	<i>Test Techniques</i> .....	- 21 -
6.2.2.1	White-Box testing.....	- 21 -
6.2.2.2	Object-Oriented Testing.....	- 22 -
6.2.3	<i>Using of TDD</i> .....	- 22 -
7	<b><u>CONCLUSION</u></b> .....	- 25 -
8	<b><u>REFERENCE</u></b> .....	- 25 -

## **1. Introduction**

For the purpose of designing a management system to cater to the requirements of operation managers, We, a six-member team, spent two months to work out a robust and reliable journey management system for daily operation and management of a newly tourist area.

Here we conclude parts groups of users which may appear for a real department of tourist area: Operation manager, Driver and Traveler. Operation manager is mainly doing the assignment, scheduling and management for allowing the journey system to run safely. Driver has particular ID and assignment, controlling the train manually on board. Travelers can obtain timetable information from two screens of trains' and stops'.

The most important part is our agile software development team is firmly subject to Agile Software Development Process. In this project, we use Microsoft Office Visio to draw related diagrams and Eclipse for JAVA programming. At the same time, we have 4 core workflows including project management, requirements, analysis and design, and implementation and test.

## **2. Summary of Responsibilities and Achievements**

**Group Number:** 96

**Group Leader:** Yiqiong Mai

**Group Members:**

<b>Name</b>	<b>BUPT Number</b>	<b>QM number</b>
Yiqiong MAI	2013213475	130804671
Zheng LIU	2013213501	130804936
Sijia YANG	2013213504	130804969
Xiaoqian HUANG	2013213507	130804992
Guangyuan ZHAO	2013213431	130806848
Junjie LU	2013213515	130805081

### **2.1 Group members' contribution**

- ✓ Discussing main functions and core principle of the journey management system.
- ✓ Discussing and design interface frame and solution of program together at the beginning of work.
- ✓ Discussing problems with each other during the weekly meeting and planning the milestones and tasks for the next period.
- ✓ Participating in weekly meeting on time with huge enthusiasm.
- ✓ Cooperation in the activities of workflows closely.

- ✓ Refinement for previous stages.

## 2.2 Individual Member Contribution and Self-Appraisal

### 2.2.1 Yiqiong MAI\_ 130806848\_ JP2013213431

Piece of Work	Relevant outcomes
<ul style="list-style-type: none"> <li>✓ Attending all the meetings and joining in discussing and making decisions.</li> <li>✓ The route assignment of the system GUI</li> <li>✓ Project Management of the final report.</li> </ul>	<ul style="list-style-type: none"> <li>✓ The code of route assignment of the system GUI</li> <li>✓ Project Management of the final report.</li> </ul>

#### Self-appraise:

As for this project, I have successfully accomplished all my tasks. I found it really hard to handle when deadlines of all the subjects encounter together. However, I made it with the cooperation with my group mates and the final product runs well. It was a story about persistence. At the beginning, I product my first version of route assignment GUI. It looks ugly and inconvenient for user to use. By learning from the excellent group mates, I ended up producing a nice GUI after rewriting the code for three times. I was so impressed by the sense of achievement.

### 2.2.2 Zheng LIU\_130804936\_ JP2013213501

Piece of Work	Relevant outcomes
<ul style="list-style-type: none"> <li>✓ Keeping track on important information of the regular meetings.</li> <li>✓ Design the structure and details of user interface.</li> <li>✓ Data processing</li> <li>✓ Finishing subject content and frame structure of final report.</li> </ul>	<ul style="list-style-type: none"> <li>✓ Code:</li> <li>✓ Backlog: story details.</li> <li>✓ Report: Introduction and conclusion.;</li> <li>Part of project management, requirement and implement and testing.</li> </ul>

#### Self-appraise:

In this group project, I am mainly in charge of logical design of functions and interfaces, including amount of little things. After first discussion, I summarized requirements and wrote the backlog and design the user interface of our software. Then I took part in some relevant coding program. Later, as the leader of reporting part, I finish the content structure and divide the reporting task into many components, enabling other members in the group to work efficiently. Actually, although facing lots of problems, I overcome majority of them by myself and solve the rest of problems with my teammates. During this period, I faced with lots of deadline and every assignment seems like an impossible task. Finally, when we were successfully running all those functions and presenting our achievements,

I felt so exciting. Through this project, I promote my ability of leadership and cooperation.

### 2.2.3 Sijia YANG\_130804969\_JP2013213504

Piece of Work	Relevant outcomes
<ul style="list-style-type: none"> <li>✓ write user GUI</li> <li>✓ Design demonstration GUI</li> <li>✓ test</li> </ul>	<ul style="list-style-type: none"> <li>✓ Report:6.2.3</li> <li>✓ Code: Main GUI designer, function developement</li> </ul>

#### Self-appraise:

I'm very appreciating that I have an opportunity to cooperate with all my lovely and responsible classmates. After the whole coursework achievement, I have learned a great deal of experience. My main responses were finishing the GUI and testing the system, as well as accomplishing the corresponding part of the report. Every time I show my work to my teammates, they can always find some drawbacks about it. Thus, I improved my work step by step through their advices and materials in the Internet. At the end, all of us are satisfy with the program's interface and its functions.

In my view, this coursework played an important part in learning software engineering. At first, I know nothing practical about agile software development. After reading the materials and all the requirements and discussing the blueprint of the system, the whole team members all have a clear imagine about this project. By dividing jobs to proper members, all of us worked effectively and efficiently.

### 2.2.4 Xiaoqian HUANG\_130804992\_JP2013213507

Piece of Work	Relevant outcomes
<ul style="list-style-type: none"> <li>✓ Discussed the functions and the construction of the management system.</li> <li>✓ Analyzed the framework of the system.</li> <li>✓ Be responsible for designing the algorithm and programming the control classes and entity classes.</li> <li>✓ Tested part of the classes using Junit.</li> <li>✓ Wrote part of the report in "Desgin and Analysis .</li> </ul>	<ul style="list-style-type: none"> <li>✓ Codes of control class and entity class - AssignDriver, AssignTrain, ScheduleRoute , ScheduleJourney, ControlOnBoardScreen, ControlStopScreen, ControlSynchronize, Train, Driver, Route and Journey.</li> <li>✓ Junit codes – AssignTrain , ScheduleJourney</li> <li>✓ Descriptions and Diagrams of the classes – 5.1, 5.2, 5.3 in the report.</li> </ul>

#### Self-appraise:

At the first time, when I read the requirement of the project I am very confused about what the system is look like and how to realize this powerful system. But in the first meeting, we analyzed the requirements and warmly discussed the corresponding functions and structures, and I gradually have the thoughts about how to program it. In the following meetings, we consummated our thoughts of

framework and classes and finally I have a clear recognition about the system. Through the cooperation, I learned the importance of communication in group working and I also acquired many experience from others.

### 2.2.5 Guangyuan ZHAO\_130806848\_JP2013213431

Piece of Work	Relevant outcomes
<ul style="list-style-type: none"> <li>✓ Design the GUI frame, including welcome menu, synchronize, on-board screen and stop screen frame. Besides, write code to achieve these functions.</li> <li>✓ Write some reports, including 4.1 and 4.2.</li> </ul>	<ul style="list-style-type: none"> <li>✓ Report : 4.1.2 , 4.1.3 and 4.2</li> <li>✓ Code: welcome class, menu class, synchronizes class, on-board class and stop class.</li> </ul>

#### Self-appraise:

From the beginning, I consider this coursework as a very difficult project. But after our team-discussion and trying our best to finish everyone's own work, we finish this coursework finally. I think I have learned how to divide a huge difficult work into many small easy works and finish it in teamwork. Besides, when I code the GUI, I find many frames are easy to draw but difficult to realize in code. So I search information on the Internet and gain a lot of new knowledge, such like the usage of cardlayout and JTable. I feel very happy and satisfied in the searching process.

### 2.2.6 Junjie LU\_130805081\_JP2013213515

Piece of Work	Relevant outcomes
<ul style="list-style-type: none"> <li>✓ Implement the function of read and update the information from file, write the class of OnBoard and OnScreen.</li> <li>✓ Write report with 6.1.</li> </ul>	<ul style="list-style-type: none"> <li>✓ Report : 6.1</li> <li>✓ Code: Class of OnBoard and OnScreen. Implement of I/O of the system.</li> </ul>

#### Self-appraise:

I am appreciating that I can a member of this group, which has lovely and excellent members. I had a lot of harvest in the project.

This project is a seemingly impossible task for me for the reason that I have poor experience. But after we have some meetings, we divided the project into more subtasks and we found that our ideas for the project became clear. From my task, I consolidated the method of I/O operation in JAVA and learned the definition and use of control class. After finishing the whole program, I get an entirely view of the software development process, especially the comprehension of some rules and theories.

Besides the knowledge in the software development, we gained something we cannot get in lecture, which is the collaboration. We knew the importance of teamwork in a complicated system. Thanks for this experience.

### **3. Project Management**

#### **3.1 Introduction**

Our team is consisting with six juniors, one leader and five members from Internet of Things. To promote the efficiency and allow everyone to devote in their specific field, we organize once again and divide our group in three subgroups.

To make effective management, it is important to make sure the project is going on in an intense but orderly manner and revise regularly. As a result, our development containing three-time periods. First, we spent two weeks on analyzing the requirement and prepare backlog. Then, we focused on implementing the software system with details and designing roughly one month. Finally, the last period is to review, evaluate and perfect our report.

#### **3.2 Management method: Scrum**

##### **3.2.1 Outline Planning**

The aim of this project is to develop a tourist train journey management system using Agile methods. The main objects of the system are operation manager, train, route, journey and the screen boards. The software architecture mainly contains three parts.

- ✓ Assignment

In this part, the operation managers can read the assigned information by using the system. They can also assign new routes, journeys, drivers and trains or make changes to the assigned data.

- ✓ Train control

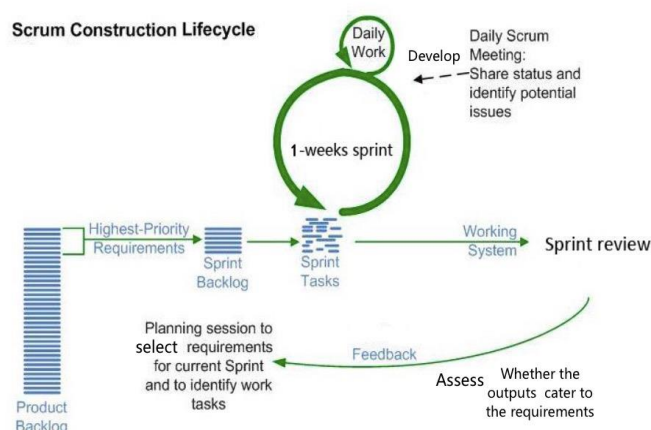
In this part, the operation managers are able to start or stop the train remotely with the help of the system.

- ✓ Information showing

In this part, the system enables the operation managers to synchronize the data of all the screen boards.

The software is developed on the basis of the architecture. Besides accomplishing the functions above, there still be details need to be defined, developed and test using the Agile method. Moreover, the design of the system is supposed to enable the following changing requirements and software alternation.

##### **3.2.2 Sprint Cycles**





### 3.2.3 Closure

In order to wrap up the project, documentation is needed to access the lessons that learned from the project. Moreover, building system helps frames and user manual.

### 3.2.4 Teamwork management and organization

#### ✓ Subgroup:

Since there were so many tasks have to be finished, many tasks can be done at the same time.

By subgrouping both of the group members and tasks, several parts of the project can be run in the meantime. In this way, the efficiency was largely improve. Also, this is the meaning of scrum.

Here is the chat of subgrouping.

ID	Tasks	Leader	Members
1	GUI development	Sijia YANG	Guangyuan ZHAO, Yiqiong MAI
2	Core code development	Xiaoqian HUANG	Junjie LU
3	Data and function processing	Zheng LIU	—

#### ✓ Regular meeting:

Apart from the subtasks, group members got together frequently for group meetings. During meeting, members discussed and made decisions together. Many tasks, like designing GUIs, building software architecture and writing the final report, were finished with the efforts of all the group members.

All the group members joined in a Wechat group. In the Wechat Group, members communicated with each other every day. Problems can be solved efficiently, which enables members cooperated better. Besides, group members got together and had a group meeting every week, sitting around a table and discussing actively. During the meeting, members demonstrated the accomplishments and made elaborate plan for the next week. In this way, members were well monitor and motivate.

## 3.3 Project Planning

### 3.3.1 Resource requirements

#### ✓ Hardware requirements:

PCs are needed by all the members, which are used for writing proposal and coding. Smartphones are also needed because they play an important role for the connection between teammates.

#### ✓ Software requirements:

We need some software for coding, drawing diagrams, writing documents and processing data, such as Eclipse, Photoshop, Word and Excel.

✓ Others:

A suitable room for routine meetings, where enables us sit around and talk with each other closely.

### 3.3.2 Activity Organization

Activity organization helps group members to clear the tasks and schedule, which makes the accomplishing and monitoring of tasks much more convenient. Moreover, there would be changes happen during this long period. Actions have to be taken to adapt the development to the new changes.

#### 3.3.2.1 Project schedule

Deadlines of the submissions serve as important time points. The schedule should be make on the basis of the deadlines. However, not all the tasks can always be finished on time. In order to deal with the risks of failing accomplishing the tasks, a flexible time need to be reserved, which is used for rescuing when the 30% anticipated problems or 20% unanticipated problems happen. Then we group the tasks together and assignment their durations. The setting of duration is supposed to be appropriate and as short as possible. After planning the elaborate schedule, tasks were able to deliver to the group members. With the sound assignment of time and people resources, the projects were able to run well.

#### 3.3.2.2 Chart

ID	Activity	Duration (days)	Dependencies	Milestones	Due Date
T1	Capture requirements	7	T1		04-01-2016
T2	Plan the project	7	T1		04-08-2016
T3	Product backlog	17	T3	M1 (Deliverable)	04-25-2016
T4	Analyze and design	3	T4		04-28-2016
T5	Subgroup and start coding	13	T5		05-11-2016
T6	Pair codes	2	T5,T6	M2	05-13-2016
T7	Test and refine	7	T2,T3,T4,T7		05-20-2016
T8	Write report	6	T7	M3 (Deliverable)	05-21-2016
T9	Demonstration	--		M4 (Deliverable)	06-03-2016

#### 3.3.2.3 Activity Breakdown

ID	Activities	Breakdowns
T1	Capture requirements	Capture user of the system
		Analyze the requirements for each user

		Building software prototypes
T2	Plan the project	List the main tasks
		Schedule the tasks
T3	Product backlog	List epics and user stories
		Analyze the priority and iteration
		Fill the product backlog
		Refine the product backlog and submission.
T4	Analyze and design	Design the GUI of the software
		Design the class diagram
		Design for extension of the system
T5	Subgroup and start coding	Code individually
T6	Pair codes	Pair the codes together and debug
T7	Test and refine	Test the quality of codes and refine for robustness
T8	Write report	Build the Architecture of the final report
		Divide report into few parts and assign to members
		Write the report individually
		Combine the report and refine
		Submit the final report and codes
T9	Demonstration	Demonstrate the system

### 3.4 Adapting to change

Since the duration of the project covers months, there will be many changes happen in this period. The developing group has to manage the changes monitoring and reporting.

#### ✓ Weekly reports

Group members wrote a brief weekly report in turns. The report contains the accomplishments, assessment of the weekly work, which serves as a good record of group work. As a result, all the changes happened during the development of software were also recorded clearly.

#### ✓ Weekly Meeting

Group members got together and have a group meeting every week. New changes were analyzed and discussed efficiently. The adaption to the plans and design were also decided during the weekly meetings.

### 3.5 Group Photo



## 4 Requirements

### 4.1 Fact-finding techniques

#### 4.1.1 Background Reading

The aim of the project is to develop a better management of journey management system to users. To better understand and design the management system, every group member has done a large amount of reading by searching relevant information (Training company reports, documentation of existing system, general sightseeing description and policy manuals) through internet. Using agile method, we spared no efforts to design the system as flexible and extensive as possible. Moreover, it was developed to be adaptable to the continuously changing of user requirements and general market.

#### 4.1.2 Observation

Observation is one of the fact-finding techniques, which can easily shorten the distance between the developers and the system. By observing the user using the current system, we can understand of the system better.

By researching some train management system (like underground railway systems), we summarized some requirements and build relevant functions.

- ✓ For operation manager, we design control function so that the manager can stop and start the train remotely. If there is train late, it cannot arrive or depart from the station on time, which may cause collision. In this case, we have to stop the train remotely in time to avoid the collision.
- ✓ For tourists, we add the time table and the late time of the next train on the stop screen so that tourists can manage their time properly according to the time table instead of waiting for a long time.
- ✓ For drivers, they can control trains manually on board. The operation manager can check status of trains controlled by drivers.

#### 4.1.3 Interviewing

In order to fully understand the requirements of users, we interviewed some of the main stakeholders: **operation manager, driver and tourist**. For the reason that the system was specifically produced for the tourist train, the users are some group of persons.

From the interview, we got that:

- ✓ Operation manager: They need get the train's real-time information so that they can synchronize it to all the drivers and users. Besides, they can stop and assign the trains properly.
- ✓ Driver: Every driver needs average allocation of work. So they hope that the manager can assign the work reasonably. They also need the real-time information of all the trains.
- ✓ Tourist: They hope that they can get the time table and the information whether the train is behind schedule in order to save time.

## 4.2 Detailed Found Requirements

### 4.2.1 Functional Requirement

- ✓ The operation manager should be able to access the information of trains and drivers, including their ID and status (available/ occupied). Besides, they modify the information of drivers and trains.
- ✓ The operation assigns the particular driver to the particular train if both of them are available. Meanwhile, he/she can assigned the particular train to the particular journey if both of them are available.
- ✓ The operation manager should be able to select the stations of the particular route and set duration.
- ✓ The operation manager should be able to manage the number of journeys for the particular route and schedule departure time of each journey.
- ✓ The operation manager should know the train's real-time statues and release the information.
- ✓ The operation manager should be able to control the train remotely.
- ✓ The driver should be able to start or stop the train manually on board.
- ✓ The tourist should be able to gain the information including the next stops and the latest time of the train on the on-board screens. They can also gain the information of upcoming train and all train time table on the train stops screen.

### 4.2.2 Non-functional Requirements

- ✓ Product requirements:  
The system should be started within 1 second with average processing time of no more than 1 second. Instruments are in simple user language. Error probability should be less than 10% with success rate of recovery from error up to 90%. The system should predict more than 90% invalid manipulation by users.
- ✓ Organizational requirements:  
The system should be developed as a standalone Java application with interfaces coded by Java GUI. The GUI interfaces should be in the similar style. Interfaces of the same function should be in consistent color and size.
- ✓ External requirements:  
The information of driver, train, journey and should be saved in the system. Only for the working reason could the operation manager check and modify the information.

## 4.3 Changes of product backlog

Story Name	Description	Prio rity	Iteration Number	Acceptance Criteria	Data started	Data Finished
Progress bar	As an operation manager, I want to have a look at the progress car of the trains. So that I can get to know the running status of trains well.	8	2	The running status must be the same as reality.	8th, May	15th, May

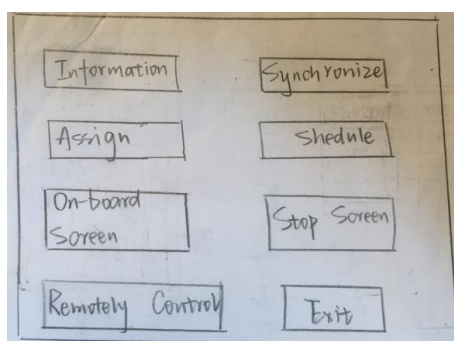
## 4.4 Software Prototypes

### 4.4.1 Logical user-interface design

- ✓ **Welcome Menu:** Welcome Menu is the first page of the system interface. The operation manager can go to any functions of the system by clicking the specific button in this page.
- ✓ **Information:** The information page offers the operation manager a way to maintain the details and status information of trains', drivers' and journeys'. In this page, the operation manager can search information of a particular driver, train or journey. Meanwhile, he/she can also add or delete information with the change of dispatching. Moreover, the operation manager can rank the informant to obtain more details.
- ✓ **Synchronize:** On synchronizing page, the operation manager can obtain the whole current location of those running trains from the client system. Meanwhile, it can also calculate the late time if it doesn't run on time and update the living status on several screens.
- ✓ **Assign:** On the assignment page, the operation managers can assign drivers to trains and assign trains to journeys. Also, they can also cancel the predefined assignments. In addition, the operation managers can only assign available trains and drivers.
- ✓ **Schedule:** On schedule page, the operation manager can choose the stops for a particular route and schedule those journeys of the route. In addition, this page can also add routes or journeys and modify those details of the new journey.
- ✓ **On-board screen:** This screen is designed to help operation manager to check the details of on-board screens, which are viewed by tourists. In this page, the operation manager can check the train timetable and late status by clicking particular button.
- ✓ **Stop screen:** This screen is designed to help operation manager to check the details of stop screens, which are used by tourists. In this page, the operation manager can check the stop screen of any stops by clicking the particular button. Also, the screen will show the details of next train and its time table.
- ✓ **Remotely control:** In this page, the operation manager can remotely control any trains by clicking a particular button. Then, it will get a feedback of whether the remotely control is successful or not.

### 4.4.2 Physical user-interface design

- ✓ Welcome Menu:



- ✓ Information:

Driver		Train	Journey
		Search (ID)	
ID	Status	Timetable	
1	available	Check	
...	occupied	check	
n	n	...	
[+] [-]		input (ID)	input (status) [reset] [back]

✓ Synchronize:

Train	Schedule 1-time	Live location	last stop time	late time

Check Synchronize

Back

✓ Assignment:

TRAIN DRIVER

Assign Driver To Train

Assign Train To Journey

Back

✓ Assign driver to train

TRAIN	DRIVER
train1	driver A1 driver B2
train2	driver C3
train3	
train n	

Back

✓ Assign train to journey

JOURNEY	TRAIN
Journey1	Train1 I Train2 II
Journey2	Train3 III
Journey m	

Back

✓ Schedule: Select stops for route

Route1  
Route2  
   
   
   
Route n

→ Outward: Center Station  
duration: /min  
1th station: 12  
   
nth station: 15  
← Return: 16  
duration: /min  
1th station: 15  
   
nth station: Center Station

Back

✓ Schedule: Select journey for route

	Journey1	Journey2	input outward time	input return time
Central Station	9:00	10:00		
nth station	12:00	13:00		
nth station	13:00	14:00		
Central Station	16:00	17:00		

Save Back

✓ On-board screen: for tourist

TRAIN	late time:
train1	Stop Time
train2	stop 12:00
train n	stop m 17:00

Back

✓ Stop screen: for tourist

STOP

Next Train: Next Train: 12:00

Arriving Time: 12:00

Late Time: 12:00

Stops	Time

Back

✓ Remotely Control

TRAIN	State:	Current Speed:
train1	start	Stop
train2	Acceleration	Deceleration
train n		

Back

#### 4.5 Iteration and estimation of the stories

Iteration	Realized Work	Estimation workload
Iteration 1 【3.14-3.28】 Rough structure and function of the whole system	<ul style="list-style-type: none"> <li>✓ Deciding all necessary user and fundamental functions of the system.</li> <li>✓ Making group schedule and establishing milestones.</li> <li>✓ Iteration of stories with aspects of user, developer and companies.</li> <li>✓ Conclude the details of requirements.</li> <li>✓ Dividing our group members into three subgroups and separating specific tasks of each group.</li> </ul>	All group members
Iteration 2 【3.29-4.15】 GUI design in java	<ul style="list-style-type: none"> <li>✓ Design the software prototype.</li> <li>✓ Modify the prototype by concerning about the detail of requirements and convenience of users.</li> <li>✓ Realize those GUI interfaces in JAVA.</li> </ul>	3 group members
Iteration 3 【4.16-4.29】 Function realization	<ul style="list-style-type: none"> <li>✓ Realize the several functions of the system such as assignment, schedule, management and synchronize.</li> <li>✓ Realize other additional functions such like displaying timetable of stop screen and train screen.</li> </ul>	2 group members
Iteration 4 【4.30-5.7】 System completion and data Implement	<ul style="list-style-type: none"> <li>✓ Combine all classes together and make the whole system run without error.</li> <li>✓ Design implication strategy and built plan.</li> <li>✓ Finish the implementation by separate the whole system of several parts.</li> </ul>	2 group members
Iteration 5 【5.7-5.14】 Test of the whole system	<ul style="list-style-type: none"> <li>✓ Test the system part and part.</li> <li>✓ Maximize the fault tolerant.</li> <li>✓ Detecting the errors of system and revise them.</li> <li>✓ Finish the last iteration of the stories to improve the functions thoroughly.</li> </ul>	2 group members

Story Id	Story Name	Story Point
1	Maintain information	1
2	Assign train to journey	2
3	Assign driver to train	2
4	Schedule route and journey	2
5	Track train status(外挂)	1
6	Update train status	4
7	Control train remotely	3
8	synchronize on-board screen information	5
9	Control train manually on board	3
10	Synchronize and display train stop information	5



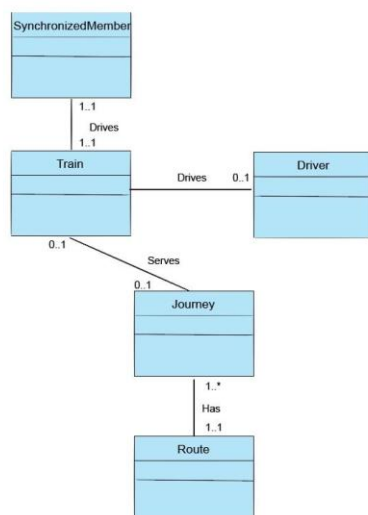
## 5 Analysis and Design

### 5.1 Analysis and Architecture Design

Analysis classes	Class	Detail
<b><u>Boundary class:</u></b> <i>(We use Train class to define all the attributes of train.)</i>	AssignDriverGUI	Design the boundary of assigning driver to the train.
	AssignTrainGUI	Design the boundary of assigning train to the journey.
	ScheduleRouteGUI	Design the boundary of scheduling stops to the route.
	ScheduleJourneyGUI	Design the boundary of scheduling departing time and returning time of the journey.
	InformationGUI	Design the boundary of maintaining information of driver, train and journey. The manager can add, delete, search and modify the information.
	OnBoardGUI	Design the boundary of displaying the next arriving stop, late time, the scheduled arriving time and the timetable of the train.
	StopScreenGUI	Design the boundary of displaying the information of the next arriving train, the late time and the timetable of the journey.
	SynchronizeGUI	Design the boundary of displaying the information from the external system and the manager can synchronize the information
	RemoteControlGUI	Design the boundary of displaying the states of the train and the manager can remotely start and stop the train and control the speed of the train.
	AssignGUI	Design the boundary of providing entrance of AssignDriver GUI and AssignTrain GUI.
	MenuGUI	Design the boundary of providing entrance of different GUI.
<b><u>Entity class:</u></b> Model information that is long-lived; logical data structure.	Train	Collect the attributes of the train, like train ID, current state of the train, which driver is assigned to the train etc.
	Driver	Collect the attributes of the driver, like driver ID, assigned train id etc.
	Route	Collect the attributes of the routes, like route ID, the duration time of each stop, stop names etc.
	Journey	Collect the attributes of the journey, like journey ID, arriving time, assign to which route etc.
<b><u>Control class:</u></b> Used to encapsulate and coordination of the main	AssignDriver	Control the data flow between train and driver and accomplish the assignment.
	AssignTrain	Control the data flow between train and journey and accomplish the assignment.
	ScheduleRoute	Control the information of the route and assign the stops and the duration to the route.
	ScheduleJourney	Control the data flow between journey and route, calculate

actions and control flows.		the arriving time of each stop and accomplish the assignment.
	ControlOnBoardScreen	Control the data flow between train, journey and the on board screen and calculate the next station according to the time and the timetable.
	ControlStopScreen	Control the data flow between route, journey and stop screen and select the next arriving journey of the stop.
	ControlSynchronize	Control the data flow between train and the on board screen, collecting the data from the train and synchronize the information to the stop screen and on board screen.
	ExternalSystem	Create interface that provide the simulation of external system's functions (collect the current location of a train on a particular journey) and record it to the management system.

## 5.2 Class relationship

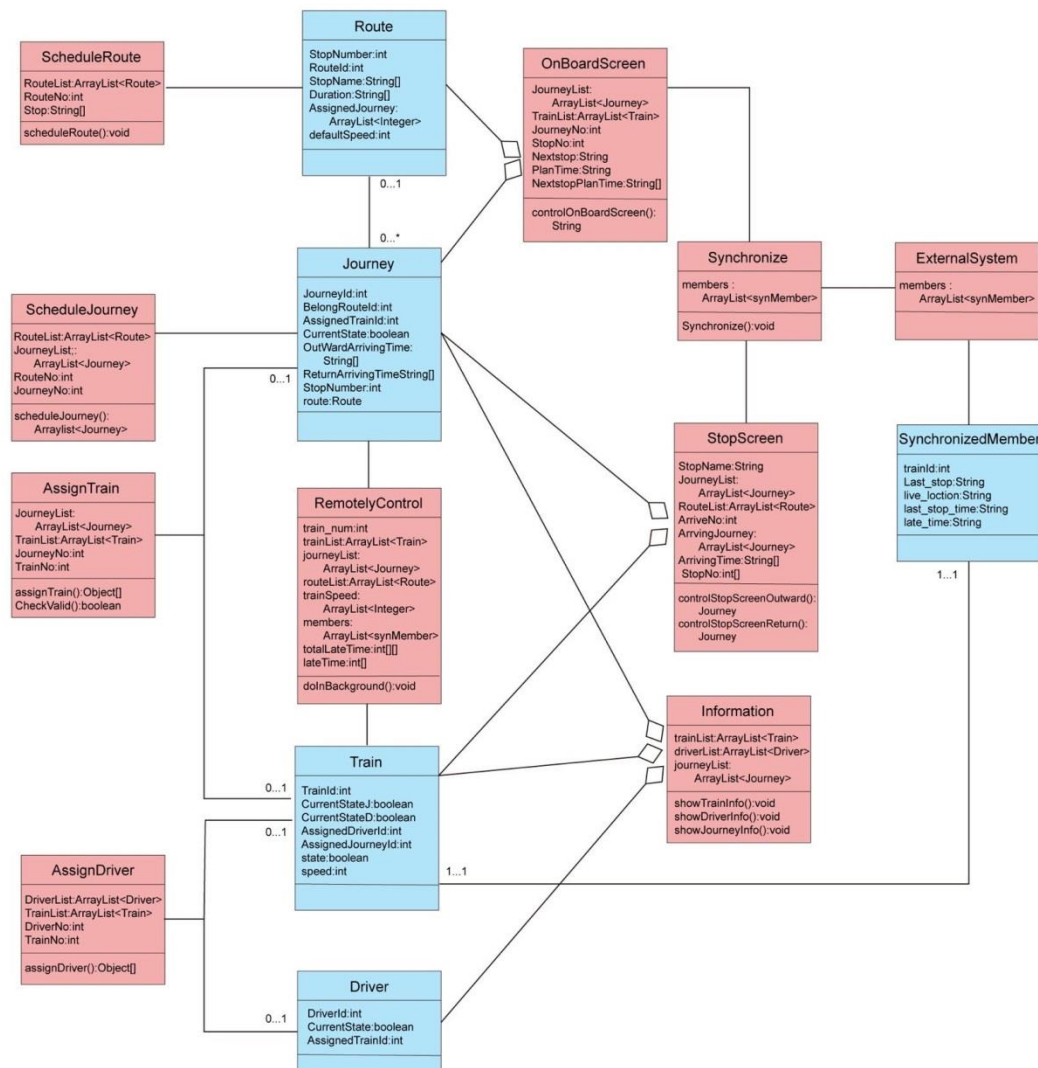


## 5.3 Re-usability of system components

Coupling, the number of dependencies between subsystems, indicates the strengths of interconnections. To decrease coupling, we avoid writing a lot of duplicate codes that have the same function. For example, there are large amounts of operation of file. It will be full of repeated code if we add the code of I/O to everywhere the file operations occurs. In order to avoid it, we wrote the code in the control class so that we can just call the method of I/O in a simple statement.

Cohesion, the number of dependencies within a subsystem, is a measure of level of functional integration within a module. In our system, we have some operations about assignment, such as assign driver to train, assign train to journey and so on. We wrote the code in the different classes for the reason that we can changed some attributes or method in the future individually.

## 5.4 Class Diagram



## 5.5 Design Principles Applied

- ✓ **Single Responsibility Principle (SRP)**

Every object in a system should have a single responsibility, and all the object's services should be focused on carrying out that single responsibility. In our train management, For example, we create the class of AssignDriver to assign the driver instead of write the method in the driver class. There are many similar examples, which aim to let one class only have one function. We can just say that a class should have only one reason to change.

- ✓ **Open-Closed Principle (OCP)**

The Open-Closed Principle means software modules should be opened for extension and closed for modification. We designed our code in a general way to meet specialized requirements. For instance, the stations in a city are fixed, but the system will behaves finely when new stations come into use without the necessity to modify the code inside.

✓ **Don't Repeat Yourself (DRY)**

In our process of programming, we avoid using “cut and paste” to copying the code and then modifying it for using in other place. In our GUI design, the information of driver, train, and journey occur in many interface. We write an informationGUI class so that we can use these data simply and avoid paste the same code many times.

✓ **Dependency Inversion Principle (DIP)**

High-level modules should not depend on low-level modules; instead, both should depend on abstractions. And abstractions should not depend on details; instead, details should depend on abstractions.

## **6 Implementation and Testing**

### **6.1 Implementation**

#### **6.1.1 Assumption**

- ✓ The registering information of each train and driver cannot be duplicated.
- ✓ Driver can only be assigned to one train; train can only be assigned to one journey.
- ✓ User can add and delete route with the given stations. A route consists of many journeys, which are difference in terms of time.
- ✓ When the operation manager get the real time data from outsource, he (or she) are authorized to decide whether update the data or not.
- ✓ The passengers in train can see the schedule and the late time of this train.
- ✓ The passengers in stop can see the information of the next train.
- ✓ When emergency happens, operation manager has the jurisdiction to control the train without the operation of driver.

#### **6.1.2 Implement Strategy**

Language: JAVA /Tools: Eclipse

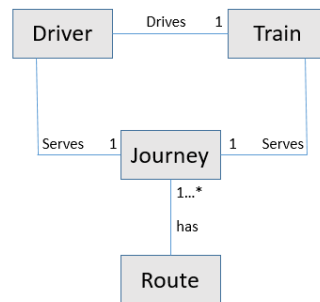
##### **6.1.2.1 Integration Build Plan**

In our system, the software is built incrementally in manageable steps and every step can produce small integration or testing problem. We divided the source code into three parts: GUI, entity class and control class. There are two groups to program them respectively.

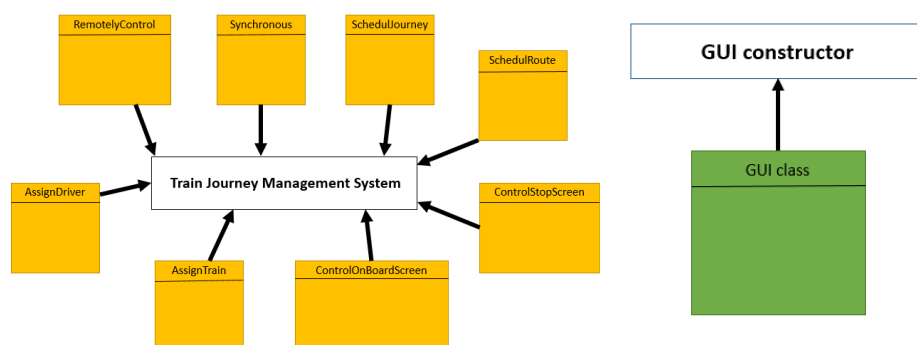
The GUI is used for better interaction between users and system. It also meets our expectation as first. The entity classes define the attributes of important things and some methods. The kernel of the system is the control classes. The operation of users will produce some data that to be record and stored.

Under the build of this part, we found that everybody can change the data of the system directly. So we used the method of serialization to save data so that we can ensure the security of the data. This is because the file in our system cannot be operated without our software. It can also improve the efficiency of file operation at the same time.

### 6.1.2.2 Association between classes



### 6.1.2.3 Mapping between design and code



### 6.1.2.4 Realized principles

All the designed principles which are mentioned in the Design and Analysis part are realized in our system.

### 6.1.3 Pair Programming Analysis

It is a difficult task for us in a busy semester, but fortunately, we have six group members to tackle all the obstacles together.

For a big team programming projects like this, the one of the difficulties is the arrangement of time and responsibility. Besides, every member in our team has an English test in the different date. During the first meeting, we determined the leader assign tasks according to members' schedule and ability. During the process, leader kept motivating the members. All the members kept attending the meetings and finish the subtasks before due day. Finally, we combine all the individual work together. Good atmosphere in a team is the best way to improve efficiency.

## 6.2 Testing

Our testing procedure demonstrates that the journey management software meets its functional requirements and system features. Meanwhile, we detected some defects of the software and tried our best to handle them.

The period of testing took up 30% timeline of the whole developing procedure. Going with perfection process of our software, the testing process went through three different levels, including unit testing, system testing and acceptance testing.

### 6.2.1 Test Strategies

There are four aspects we thinking about in making testing strategies: What tests to run, how to run them, when to run them, how to determine whether the testing effort is successful.

#### 6.2.1.1 Testing cases

- ✓ This part is to answer the question what tests to run,
- ✓ Roughly 70% of the tests will be automated and the reminders will be manual.
- ✓ These cases are designed to check the correction of user interface and to ensure we can handle the incorrect of inputs.
- ✓ We designed test case for almost all classes to confirm the condition of implementation of the fundamental functions.

#### 6.2.1.2 Testing process

- ✓ This part is to answer how to run them and when to run them
- ✓ During the part of unit testing, we mainly use object-oriented testing method to test individual component. Meanwhile, Junit was in the process of test driven development agile process.
- ✓ During system testing, we mainly used white box testing techniques to ensure that all statements and conditions have been executed at least once.

#### 6.2.1.3 Success criteria

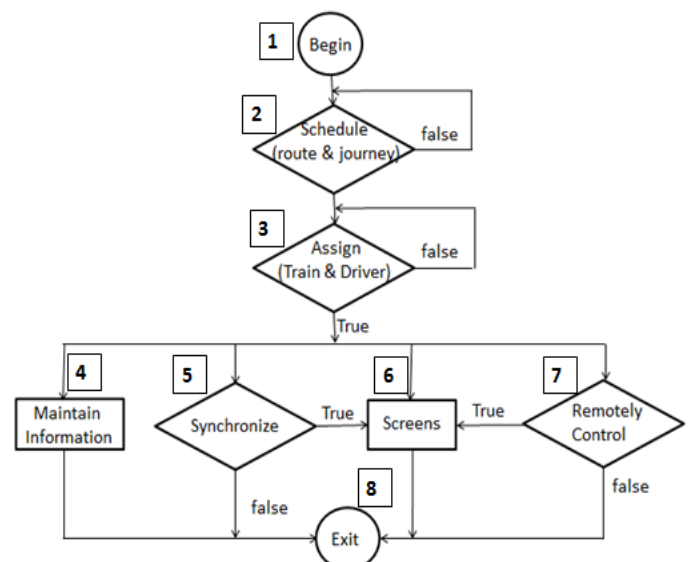
- ✓ This part is to answer the question how to determine whether the testing effort is successful
- ✓ Success criteria – 85% of test cases passed and no high priority defects unsolved.

### 6.2.2 Test Techniques

During the testing process, we use white-box testing to test the internal program logic and use black-box testing to test software requirements. The details are as follow.

#### 6.2.2.1 White-Box testing

- ✓ Basic path passing:
- ✓ Path 1-2-3-4-8 tests the validation of basic functions of management. During this path, the operation manager can maintain all relevant information of entities including routes, journeys, trains and drivers. During the process of testing, operation manager will add a new route into system,



select passing stations. Also, she/he sets the outward departure time and return departure time of every journey of the specific route. Then, in assignment page, operation manager will arrange a train to the specific journey and arrange a driver to the specific train. Above all, the creation of a route has finished and the operation manager can check the status in information page. Meanwhile, he/she can add or delete the information of trains and drivers. After every input of system, user needs to confirm and save the information. Each operation needs correct type of inputs so that the operation will be done; otherwise, a warning will be received.

- ✓ Path 5-6-8 tests the validation of the synchronize function of management. In synchronize page, the operation manager will get the current location of all running trains and system will calculate the late status and late time of all trains. Then the operation manager can synchronize all late information to the relevant screens, including the stop screen and on-board screen. If the information of the screens and the details of synchronization are not matched, the system will return a warning.
- ✓ Path 7-6-8 tests the validation of the remotely control function of management. In remotely control page, the operation manager can control the running status of any trains, including stop, start and adjust speed. If the operation is wrong, the running status will not be displayed in the screens.

### 6.2.2.2 Object-Oriented Testing

The object-oriented testing began prior to the existence of source code. We tested the individual classes and subsystem by using JUnit component, helping us to modify our code and system.

### 6.2.3 Using of TDD

#### ✓ Schedule route

At first, we should test whether we can create a route correctly, including setting the stations we plan to stop and scheduling the duration between them. In the test we plan to create two routes. Route1 has 3 stop. The duration from “Guangzhou” to “Beijing” is 10 minutes and the duration from “Beijing” to “Hebei” is 30 minutes. Route 2 has 2 stops and the duration between them is one hour. By invoking schedule() method and sending the stop name and duration arrays, we can successfully create two routes, whose parameters are both correctly set without interfere.

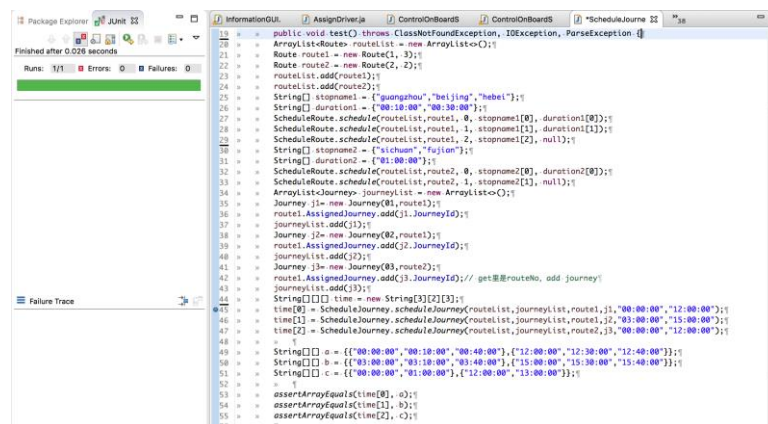
```

18 public void testSchedule() throws ClassNotFoundException, IOException, ParseException {
19     ArrayList<Route> routeList = new ArrayList<>();
20     Route route1 = new Route(1, 3);
21     Route route2 = new Route(2, 2);
22     routeList.add(route1);
23     routeList.add(route2);
24     String[] stopname1 = {"Guangzhou", "Beijing", "Hebei"};
25     String[] duration1 = {"10", "30"};
26     String[] result1 = new String[2][2];
27     result1[0] = ScheduleRoute.schedule(routeList, route1, 0, stopname1[0], duration1[0]);
28     result1[1] = ScheduleRoute.schedule(routeList, route1, 1, stopname1[1], duration1[1]);
29     String[] result2 = {"Guangzhou", "Hebei"};
30     String[] duration2 = {"60"};
31     result2[0] = ScheduleRoute.schedule(routeList, route2, 0, stopname1[0], duration2[0]);
32     result2[1] = ScheduleRoute.schedule(routeList, route2, 1, stopname1[1], duration2[1]);
33     String[] stopname2 = {"Guangzhou", "Hebei"};
34     String[] duration2 = {"60"};
35     String[] result2 = new String[2][2];
36     result2[0] = ScheduleRoute.schedule(routeList, route2, 0, stopname1[0], duration2[0]);
37     result2[1] = ScheduleRoute.schedule(routeList, route2, 1, stopname1[1], duration2[1]);
38     String[] stopname2 = {"Guangzhou", "Hebei"};
39     String[] duration2 = {"60"};
40     String[] result2 = new String[2][2];
41     result2[0] = ScheduleRoute.schedule(routeList, route2, 0, stopname1[0], duration2[0]);
42     result2[1] = ScheduleRoute.schedule(routeList, route2, 1, stopname1[1], duration2[1]);
43     assertEquals("routeList.size()", routeList.size(), 2);
44     assertEquals("route1.stopname()", route1.stopname(), stopname1);
45     assertEquals("route2.stopname()", route2.stopname(), stopname2);
46     assertEquals("route1.duration()", route1.duration(), duration1);
47     assertEquals("route2.duration()", route2.duration(), duration2);
48     assertEquals("route1.stopname()", route1.stopname(), stopname1);
49     assertEquals("route2.stopname()", route2.stopname(), stopname2);
50     assertEquals("route1.duration()", route1.duration(), duration1);
51     assertEquals("route2.duration()", route2.duration(), duration2);
52     assertEquals("route1.stopname()", route1.stopname(), stopname1);
53     assertEquals("route2.stopname()", route2.stopname(), stopname2);
54     assertEquals("route1.duration()", route1.duration(), duration1);
55     assertEquals("route2.duration()", route2.duration(), duration2);
56     assertEquals("route1.stopname()", route1.stopname(), stopname1);
57     assertEquals("route2.stopname()", route2.stopname(), stopname2);
58     assertEquals("route1.duration()", route1.duration(), duration1);
59     assertEquals("route2.duration()", route2.duration(), duration2);
60 }

```

## ✓ Schedule journey

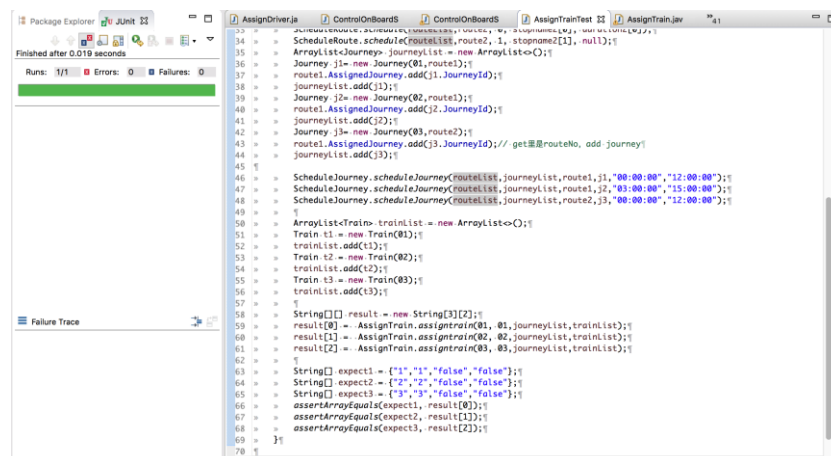
After creating routes, we wanted to schedule journeys for routes with different departing time and returning time. In this test, we create 3 journeys. Journey 1 and Journey 2 belong to route1 while Journey 3 belongs to route 2. Journey1 departs at 00:00:00 and returns at 12:00:00; Journey 2 departs at 03:00:00 and returns at 15:00:00; Journey 3 departs at 00:00:00 and returns at 12:00:00. According to the route's duration, we expect Journey 1 arrives at "Beijing" at 00:10:00 since the duration is 10 minutes. And other expectations are calculated in the same way. By invoking `scheduleJourney()` method with these parameter, results returned by the method is exactly the same as we expect.



```
1 public void test() throws ClassNotFoundException, IOException, ParseException {
2     ArrayList<Route> routelist = new ArrayList<>();
3     Route route1 = new Route(1, 3);
4     Route route2 = new Route(2, 2);
5     routelist.add(route1);
6     routelist.add(route2);
7     String[] stopname1 = {"Guangzhou", "Beijing", "Hebei"};
8     String[] duration1 = {"00:10:00", "00:30:00"};
9     ScheduleRoute.schedule(routelist, route1, 0, stopname1[0], duration1[0]);
10    ScheduleRoute.schedule(routelist, route1, 1, stopname1[1], duration1[1]);
11    ScheduleRoute.schedule(routelist, route2, 0, stopname1[0], null);
12    String[] stopname2 = {"Taichuan", "Fujian"};
13    String[] duration2 = {"01:00:00"};
14    ScheduleRoute.schedule(routelist, route2, 0, stopname2[0], duration2[0]);
15    ScheduleRoute.schedule(routelist, route2, 1, stopname2[1], null);
16    ArrayList<Journey> journeylist = new ArrayList<>();
17    Journey j1 = new Journey(01, route1);
18    routelist.add(j1);
19    Journey j2 = new Journey(02, route1);
20    routelist.add(j2);
21    Journey j3 = new Journey(03, route2);
22    routelist.add(j3);
23    JourneyList.add(j1);
24    JourneyList.add(j2);
25    JourneyList.add(j3);
26    String[] time = new String[3][2];
27    time[0] = ScheduleJourney.scheduleJourney(routelist, journeylist, route1, 0, "00:00:00", "12:00:00");
28    time[1] = ScheduleJourney.scheduleJourney(routelist, journeylist, route1, 1, "03:00:00", "15:00:00");
29    time[2] = ScheduleJourney.scheduleJourney(routelist, journeylist, route2, 0, "00:00:00", "12:00:00");
30    String[] a = {"00:00:00", "00:10:00", "00:40:00", "12:00:00", "12:40:00"};
31    String[] b = {"03:00:00", "03:10:00", "03:40:00", "15:00:00", "15:40:00"};
32    String[] c = {"00:00:00", "01:00:00", "12:00:00", "13:00:00"};
33    assertEquals(time[0], a);
34    assertEquals(time[1], b);
35    assertEquals(time[2], c);
36 }
```

## ✓ Assign train:

Now we want to assign the train to the journeys. Note that a train can only be assigned to one specified journey at one time while one journey can only have one train, too. Thus, we need to verify that once we successfully assign one train to one journey, both train's and journey's state must change to "false" which means it cannot be assigned again. At the same time, train must record the journeyId it assigned to and journey will record the trainId, too. The array result record all the information that will change after assignment (i.e. state and assigned\_id), which is the same as we expect.

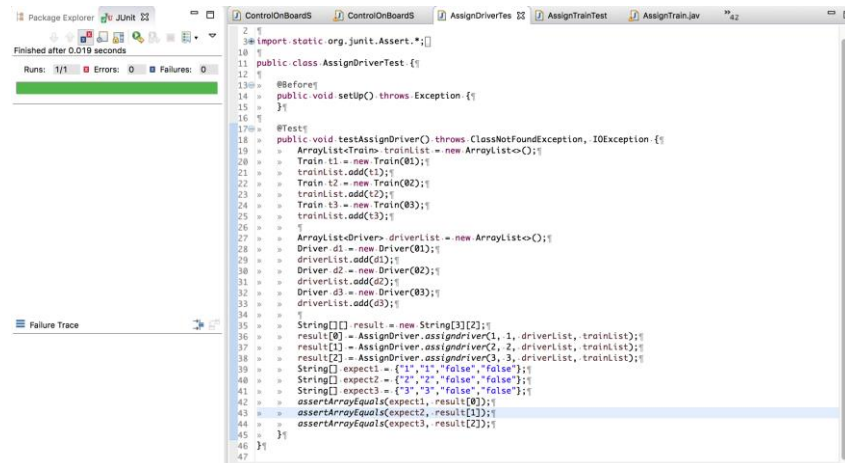


```
1 public void test() throws ClassNotFoundException, IOException, ParseException {
2     ArrayList<Journey> journeylist = new ArrayList<>();
3     Journey j1 = new Journey(01, route1);
4     JourneyList.add(j1);
5     Journey j2 = new Journey(02, route1);
6     JourneyList.add(j2);
7     Journey j3 = new Journey(03, route2);
8     JourneyList.add(j3);
9     ArrayList<Train> trainlist = new ArrayList<>();
10    Train t1 = new Train(01);
11    Train t2 = new Train(02);
12    Train t3 = new Train(03);
13    trainlist.add(t1);
14    trainlist.add(t2);
15    trainlist.add(t3);
16    String[] result = new String[3][2];
17    result[0] = AssignTrain.assignTrain(01, 01, journeylist, trainlist);
18    result[1] = AssignTrain.assignTrain(02, 02, journeylist, trainlist);
19    result[2] = AssignTrain.assignTrain(03, 03, journeylist, trainlist);
20    String[] expect1 = {"1", "1", "false", "false"};
21    String[] expect2 = {"2", "2", "false", "false"};
22    String[] expect3 = {"3", "3", "false", "false"};
23    assertEquals(expect1, result[0]);
24    assertEquals(expect2, result[1]);
25    assertEquals(expect3, result[2]);
26 }
```



### ✓ Assign driver:

After we assign the train to the journey, we should assign a driver to the train so that it can start its journey. The test is almost the same as the “assign train test”, in which we saw driver and train’s information is correctly modified after assignment. It should be pointed out that one train has two states, which are referring to journey and driver respectively.



```
1 2
3 import static org.junit.Assert.*;
4
5 10
6 11 public class AssignDriverTest {
7
8 12
9 13 @Before
10 14 public void setUp() throws Exception {
11 15
12 16
13 17 @Test
14 18 public void testAssignDriver() throws ClassNotFoundException, IOException {
15 19     ArrayList<Train> trainList = new ArrayList<>();
16 20     Train t1 = new Train(01);
17 21     trainList.add(t1);
18 22     Train t2 = new Train(02);
19 23     trainList.add(t2);
20 24     Train t3 = new Train(03);
21 25     trainList.add(t3);
22 26
23 27     ArrayList<Driver> driverList = new ArrayList<>();
24 28     Driver d1 = new Driver(01);
25 29     driverList.add(d1);
26 30     Driver d2 = new Driver(02);
27 31     driverList.add(d2);
28 32     Driver d3 = new Driver(03);
29 33     driverList.add(d3);
30 34
31 35     String[] result = new String[3][2];
32 36     result[0] = AssignDriver.assignDriver(1, 1, driverList, trainList);
33 37     result[1] = AssignDriver.assignDriver(2, 2, driverList, trainList);
34 38     result[2] = AssignDriver.assignDriver(3, 3, driverList, trainList);
35 39     String[] expect1 = {"1", "1", "false", "false"};
36 40     String[] expect2 = {"2", "2", "false", "false"};
37 41     String[] expect3 = {"3", "3", "false", "false"};
38 42     assertEquals(expect1, result[0]);
39 43     assertEquals(expect2, result[1]);
40 44     assertEquals(expect3, result[2]);
41 45
42 46
43 47
44 48
45 49
46 50
47 51
48 52
49 53
50 54
51 55
52 56
53 57
54 58
55 59
56 60
57 61
58 62
59 63
60 64
61 65
62 66
63 67
64 68
65 69
66 70
67 71
68 72
69 73
70 74
71 75
72 76
73 77
74 78
75 79
76 80
77 81
78 82
79 83
80 84
81 85
82 86
83 87
84 88
85 89
86 90
87 91
88 92
89 93
90 94
91 95
92 96
93 97
94 98
95 99
96 100
97 101
98 102
99 103
100 104
101 105
102 106
103 107
104 108
105 109
106 110
107 111
108 112
109 113
110 114
111 115
112 116
113 117
114 118
115 119
116 120
117 121
118 122
119 123
120 124
121 125
122 126
123 127
124 128
125 129
126 130
127 131
128 132
129 133
130 134
131 135
132 136
133 137
134 138
135 139
136 140
137 141
138 142
139 143
140 144
141 145
142 146
143 147
144 148
145 149
146 150
147 151
148 152
149 153
150 154
151 155
152 156
153 157
154 158
155 159
156 160
157 161
158 162
159 163
160 164
161 165
162 166
163 167
164 168
165 169
166 170
167 171
168 172
169 173
170 174
171 175
172 176
173 177
174 178
175 179
176 180
177 181
178 182
179 183
180 184
181 185
182 186
183 187
184 188
185 189
186 190
187 191
188 192
189 193
190 194
191 195
192 196
193 197
194 198
195 199
196 200
197 201
198 202
199 203
200 204
201 205
202 206
203 207
204 208
205 209
206 210
207 211
208 212
209 213
210 214
211 215
212 216
213 217
214 218
215 219
216 220
217 221
218 222
219 223
220 224
221 225
222 226
223 227
224 228
225 229
226 230
227 231
228 232
229 233
230 234
231 235
232 236
233 237
234 238
235 239
236 240
237 241
238 242
239 243
240 244
241 245
242 246
243 247
244 248
245 249
246 250
247 251
248 252
249 253
250 254
251 255
252 256
253 257
254 258
255 259
256 260
257 261
258 262
259 263
260 264
261 265
262 266
263 267
264 268
265 269
266 270
267 271
268 272
269 273
270 274
271 275
272 276
273 277
274 278
275 279
276 280
277 281
278 282
279 283
280 284
281 285
282 286
283 287
284 288
285 289
286 290
287 291
288 292
289 293
290 294
291 295
292 296
293 297
294 298
295 299
296 300
297 301
298 302
299 303
300 304
301 305
302 306
303 307
304 308
305 309
306 310
307 311
308 312
309 313
310 314
311 315
312 316
313 317
314 318
315 319
316 320
317 321
318 322
319 323
320 324
321 325
322 326
323 327
324 328
325 329
326 330
327 331
328 332
329 333
330 334
331 335
332 336
333 337
334 338
335 339
336 340
337 341
338 342
339 343
340 344
341 345
342 346
343 347
344 348
345 349
346 350
347 351
348 352
349 353
350 354
351 355
352 356
353 357
354 358
355 359
356 360
357 361
358 362
359 363
360 364
361 365
362 366
363 367
364 368
365 369
366 370
367 371
368 372
369 373
370 374
371 375
372 376
373 377
374 378
375 379
376 380
377 381
378 382
379 383
380 384
381 385
382 386
383 387
384 388
385 389
386 390
387 391
388 392
389 393
390 394
391 395
392 396
393 397
394 398
395 399
396 400
397 401
398 402
399 403
400 404
401 405
402 406
403 407
404 408
405 409
406 410
407 411
408 412
409 413
410 414
411 415
412 416
413 417
414 418
415 419
416 420
417 421
418 422
419 423
420 424
421 425
422 426
423 427
424 428
425 429
426 430
427 431
428 432
429 433
430 434
431 435
432 436
433 437
434 438
435 439
436 440
437 441
438 442
439 443
440 444
441 445
442 446
443 447
444 448
445 449
446 450
447 451
448 452
449 453
450 454
451 455
452 456
453 457
454 458
455 459
456 460
457 461
458 462
459 463
460 464
461 465
462 466
463 467
464 468
465 469
466 470
467 471
468 472
469 473
470 474
471 475
472 476
473 477
474 478
475 479
476 480
477 481
478 482
479 483
480 484
481 485
482 486
483 487
484 488
485 489
486 490
487 491
488 492
489 493
490 494
491 495
492 496
493 497
494 498
495 499
496 500
497 501
498 502
499 503
500 504
501 505
502 506
503 507
504 508
505 509
506 510
507 511
508 512
509 513
510 514
511 515
512 516
513 517
514 518
515 519
516 520
517 521
518 522
519 523
520 524
521 525
522 526
523 527
524 528
525 529
526 530
527 531
528 532
529 533
530 534
531 535
532 536
533 537
534 538
535 539
536 540
537 541
538 542
539 543
540 544
541 545
542 546
543 547
544 548
545 549
546 550
547 551
548 552
549 553
550 554
551 555
552 556
553 557
554 558
555 559
556 560
557 561
558 562
559 563
560 564
561 565
562 566
563 567
564 568
565 569
566 570
567 571
568 572
569 573
570 574
571 575
572 576
573 577
574 578
575 579
576 580
577 581
578 582
579 583
580 584
581 585
582 586
583 587
584 588
585 589
586 590
587 591
588 592
589 593
590 594
591 595
592 596
593 597
594 598
595 599
596 600
597 601
598 602
599 603
600 604
601 605
602 606
603 607
604 608
605 609
606 610
607 611
608 612
609 613
610 614
611 615
612 616
613 617
614 618
615 619
616 620
617 621
618 622
619 623
620 624
621 625
622 626
623 627
624 628
625 629
626 630
627 631
628 632
629 633
630 634
631 635
632 636
633 637
634 638
635 639
636 640
637 641
638 642
639 643
640 644
641 645
642 646
643 647
644 648
645 649
646 650
647 651
648 652
649 653
650 654
651 655
652 656
653 657
654 658
655 659
656 660
657 661
658 662
659 663
660 664
661 665
662 666
663 667
664 668
665 669
666 670
667 671
668 672
669 673
670 674
671 675
672 676
673 677
674 678
675 679
676 680
677 681
678 682
679 683
680 684
681 685
682 686
683 687
684 688
685 689
686 690
687 691
688 692
689 693
690 694
691 695
692 696
693 697
694 698
695 699
696 700
697 701
698 702
699 703
700 704
701 705
702 706
703 707
704 708
705 709
706 710
707 711
708 712
709 713
710 714
711 715
712 716
713 717
714 718
715 719
716 720
717 721
718 722
719 723
720 724
721 725
722 726
723 727
724 728
725 729
726 730
727 731
728 732
729 733
730 734
731 735
732 736
733 737
734 738
735 739
736 740
737 741
738 742
739 743
740 744
741 745
742 746
743 747
744 748
745 749
746 750
747 751
748 752
749 753
750 754
751 755
752 756
753 757
754 758
755 759
756 760
757 761
758 762
759 763
760 764
761 765
762 766
763 767
764 768
765 769
766 770
767 771
768 772
769 773
770 774
771 775
772 776
773 777
774 778
775 779
776 780
777 781
778 782
779 783
780 784
781 785
782 786
783 787
784 788
785 789
786 790
787 791
788 792
789 793
790 794
791 795
792 796
793 797
794 798
795 799
796 800
797 801
798 802
799 803
800 804
801 805
802 806
803 807
804 808
805 809
806 810
807 811
808 812
809 813
810 814
811 815
812 816
813 817
814 818
815 819
816 820
817 821
818 822
819 823
820 824
821 825
822 826
823 827
824 828
825 829
826 830
827 831
828 832
829 833
830 834
831 835
832 836
833 837
834 838
835 839
836 840
837 841
838 842
839 843
840 844
841 845
842 846
843 847
844 848
845 849
846 850
847 851
848 852
849 853
850 854
851 855
852 856
853 857
854 858
855 859
856 860
857 861
858 862
859 863
860 864
861 865
862 866
863 867
864 868
865 869
866 870
867 871
868 872
869 873
870 874
871 875
872 876
873 877
874 878
875 879
876 880
877 881
878 882
879 883
880 884
881 885
882 886
883 887
884 888
885 889
886 890
887 891
888 892
889 893
890 894
891 895
892 896
893 897
894 898
895 899
896 900
897 901
898 902
899 903
900 904
901 905
902 906
903 907
904 908
905 909
906 910
907 911
908 912
909 913
910 914
911 915
912 916
913 917
914 918
915 919
916 920
917 921
918 922
919 923
920 924
921 925
922 926
923 927
924 928
925 929
926 930
927 931
928 932
929 933
930 934
931 935
932 936
933 937
934 938
935 939
936 940
937 941
938 942
939 943
940 944
941 945
942 946
943 947
944 948
945 949
946 950
947 951
948 952
949 953
950 954
951 955
952 956
953 957
954 958
955 959
956 960
957 961
958 962
959 963
960 964
961 965
962 966
963 967
964 968
965 969
966 970
967 971
968 972
969 973
970 974
971 975
972 976
973 977
974 978
975 979
976 980
977 981
978 982
979 983
980 984
981 985
982 986
983 987
984 988
985 989
986 990
987 991
988 992
989 993
990 994
991 995
992 996
993 997
994 998
995 999
996 1000
997 1001
998 1002
999 1003
1000 1004
1001 1005
1002 1006
1003 1007
1004 1008
1005 1009
1006 1010
1007 1011
1008 1012
1009 1013
1010 1014
1011 1015
1012 1016
1013 1017
1014 1018
1015 1019
1016 1020
1017 1021
1018 1022
1019 1023
1020 1024
1021 1025
1022 1026
1023 1027
1024 1028
1025 1029
1026 1030
1027 1031
1028 1032
1029 1033
1030 1034
1031 1035
1032 1036
1033 1037
1034 1038
1035 1039
1036 1040
1037 1041
1038 1042
1039 1043
1040 1044
1041 1045
1042 1046
1043 1047
1044 1048
1045 1049
1046 1050
1047 1051
1048 1052
1049 1053
1050 1054
1051 1055
1052 1056
1053 1057
1054 1058
1055 1059
1056 1060
1057 1061
1058 1062
1059 1063
1060 1064
1061 1065
1062 1066
1063 1067
1064 1068
1065 1069
1066 1070
1067 1071
1068 1072
1069 1073
1070 1074
1071 1075
1072 1076
1073 1077
1074 1078
1075 1079
1076 1080
1077 1081
1078 1082
1079 1083
1080 1084
1081 1085
1082 1086
1083 1087
1084 1088
1085 1089
1086 1090
1087 1091
1088 1092
1089 1093
1090 1094
1091 1095
1092 1096
1093 1097
1094 1098
1095 1099
1096 1100
1097 1101
1098 1102
1099 1103
1100 1104
1101 1105
1102 1106
1103 1107
1104 1108
1105 1109
1106 1110
1107 1111
1108 1112
1109 1113
1110 1114
1111 1115
1112 1116
1113 1117
1114 1118
1115 1119
1116 1120
1117 1121
1118 1122
1119 1123
1120 1124
1121 1125
1122 1126
1123 1127
1124 1128
1125 1129
1126 1130
1127 1131
1128 1132
1129 1133
1130 1134
1131 1135
1132 1136
1133 1137
1134 1138
1135 1139
1136 1140
1137 1141
1138 1142
1139 1143
1140 1144
1141 1145
1142 1146
1143 1147
1144 1148
1145 1149
1146 1150
1147 1151
1148 1152
1149 1153
1150 1154
1151 1155
1152 1156
1153 1157
1154 1158
1155 1159
1156 1160
1157 1161
1158 1162
1159 1163
1160 1164
1161 1165
1162 1166
1163 1167
1164 1168
1165 1169
1166 1170
1167 1171
1168 1172
1169 1173
1170 1174
1171 1175
1172 1176
1173 1177
1174 1178
1175 1179
1176 1180
1177 1181
1178 1182
1179 1183
1180 1184
1181 1185
1182 1186
1183 1187
1184 1188
1185 1189
1186 1190
1187 1191
1188 1192
1189 1193
1190 1194
1191 1195
1192 1196
1193 1197
1194 1198
1195 1199
1196 1200
1197 1201
1198 1202
1199 1203
1200 1204
1201 1205
1202 1206
1203 1207
1204 1208
1205 1209
1206 1210
1207 1211
1208 1212
1209 1213
1210 1214
1211 1215
1212 1216
1213 1217
1214 1218
1215 1219
1216 1220
1217 1221
1218 1222
1219 1223
1220 1224
1221 1225
1222 1226
1223 1227
1224 1228
1225 1229
1226 1230
1227 1231
1228 1232
1229 1233
1230 1234
1231 1235
1232 1236
1233 1237
1234 1238
1235 1239
1236 1240
1237 1241
1238 1242
1239 1243
1240 1244
1241 1245
1242 1246
1243 1247
1244 1248
1245 1249
1246 1250
1247 1251
1248 1252
1249 1253
1250 1254
1251 1255
1252 1256
1253 1257
1254 1258
1255 1259
1256 1260
1257 1261
1258 1262
1259 1263
1260 1264
1261 1265
1262 1266
1263 1267
1264 1268
1265 1269
1266 1270
1267 1271
1268 1272
1269 1273
1270 1274
1271 1275
1272 1276
1273 1277
1274 1278
1275 1279
1276 1280
1277 1281
1278 1282
1279 1283
1280 1284
1281 1285
1282 1286
1283 1287
1284 1288
1285 1289
1286 1290
1287 1291
1288 1292
1289 1293
1290 1294
1291 1295
1292 1296
1293 1297
1294 1298
1295 1299
1296 1300
1297 1301
1298 1302
1299 1303
1300 1304
1301 1305
1302 1306
1303 1307
1304 1308
1305 1309
1306 1310
1307 1311
1308 1312
1309 1313
1310 1314
1311 1315
1312 1316
1313 1317
1314 1318
1315 1319
1316 1320
1317 1321
1318 1322
1319 1323
1320 1324
1321 1325
1322 1326
1323 1327
1324 1328
1325 1329
1326 1330
1327 1331
1328 1332
1329 1333
1330 1334
1331 1335
1332 1336
1333 1337
1334 1338
1335 1339
1336 1340
1337 1341
1338 1342
1339 1343
1340 1344
1341 1345
1342 1346
1343 1347
1344 1348
1345 1349
1346 1350
1347 1351
1348 1352
1349 1353
1350 1354
1351 1355
1352 1356
1353 1357
1354 1358
1355 1359
1356 1360
1357 1361
1358 1362
1359 1363
1360 1364
1361 1365
1362 1366
1363 1367
1364 1368
1365 1369
1366 1370
1367 1371
1368 1372
1369 1373
1370 1374
1371 1375
1372 1376
1373 1377
1374 1378
1375 1379
1376 1380
1377 1381
1378 1382
1379 1383
1380 1384
1381 1385
1382 1386
1383 1387
1384 1388
1385 1389
1386 1390
1387 1391
1388 1392
1389 1393
1390 1394
1391 1395
1392 1396
1393 1397
1394 1398
1395 1399
1396 1400
1397 1401
1398 1402
1399 
```

and journeyList in which all the routes and the journeys belong to them are recorded, the method Outward() and Return() return the correct information.(i.e. the journey, from which we can tell the next train by checking its assigned\_trainId. ) Because the time we executed this test is 15:25:00, we set the depart/return to 15:20:00, thereby the train assigned to journey1 will arrive at “Beijing” station at 15:30:00(outward)/15:50:00(return). Thus the next journey to “Beijing” will be journey1.

```

22 public void testOutward() throws ClassNotFoundException, IOException, ParseException {
23     routelist = new ArrayList<>();
24     route1 = new Route(1, 3);
25     routelist.add(route1);
26     String[] stopname1 = {"Guangzhou", "Beijing", "Hebei"};
27     String[] duration1 = {"00:10:00", "00:30:00"};
28     ScheduleRoute.schedule(routelist, route1, 0, stopname[0], duration[0]);
29     ScheduleRoute.schedule(routelist, route1, 1, stopname[1], duration[1]);
30     ScheduleRoute.schedule(routelist, route1, 2, stopname[2], null);
31     journeyList = new ArrayList<>();
32     j1 = new Journey(0, route1);
33     route1.AssignJourney.add(j1, JourneyId);
34     journeyList.add(j1);
35     ScheduleJourney.scheduleJourney(routelist, journeyList, j1, "15:20:00", "16:00:00");
36     Journey result;
37     result = ControlStopScreen.Outward("Beijing", routelist, journeyList);
38     assertEquals(j1, result);
39 }
40
41 @Test
42 public void testReturn() throws ClassNotFoundException, IOException, ParseException {
43     routelist = new ArrayList<>();
44     route1 = new Route(1, 3);
45     routelist.add(route1);
46     String[] stopname1 = {"Guangzhou", "Beijing", "Hebei"};
47     String[] duration1 = {"00:10:00", "00:30:00"};
48     ScheduleRoute.schedule(routelist, route1, 0, stopname[0], duration[0]);
49     ScheduleRoute.schedule(routelist, route1, 1, stopname[1], duration[1]);
50     ScheduleRoute.schedule(routelist, route1, 2, stopname[2], null);
51     journeyList = new ArrayList<>();
52     j1 = new Journey(0, route1);
53     route1.AssignJourney.add(j1, JourneyId);
54     journeyList.add(j1);
55     ScheduleJourney.scheduleJourney(routelist, journeyList, j1, "14:10:00", "15:20:00");
56     Journey result;
57     result = ControlStopScreen.Return("Beijing", routelist, journeyList);
58     assertEquals(j1, result);
59 }

```

## 7 Conclusion

During the assignment of developing this journey management system, we, the Agile software development team, following with agile process step by step, are gradually produced the final software system and description report.

After the developing process, the final software system is able to meet majority requirements in requirement-finding stage and implement fundamental operations of system. In addition, to offer users a more friendly and convenient interface, we added some other functions such as searching, automatic calculation and fault-tolerant.

In our group, the aspects which every member is skilled in are totally different. We cooperate with each other and handle all problems successfully. By doing this project, all members had better understanding of software engineering.

## 8 Reference

- [1] Ian Sommerville. (2009). *Software Engineering, Ninth Edition*
- [2] Kathy Sierra, Bert Bates. (2006). *Head First Java.*
- [3] Dan Pilone, Russell Miles. (2007). *Head First Software Development*