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. 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:

Name	BUPT Number	QM number
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
✓ Attending all the meetings and joining in discussing and making decisions. ✓ The route assignment of the system GUI ✓ Project Management of the final report.	✓ The code of route assignment of the system GUI ✓ Project Management of the final report.

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
✓ Keeping track on important information of the regular meetings. ✓ Design the structure and details of user interface. ✓ Data processing ✓ Finishing subject content and frame structure of final report.	✓ Code: ✓ Backlog: story details. ✓ Report: Introduction and conclusion.; Part of project management, requirement and implement and testing.

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
✓ write user GUI ✓ Design demonstration GUI ✓ test	✓ Report:6.2.3 ✓ Code: Main GUI designer, function developement

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 th e system, as well as accomplishing the corresponding part of the report. Every tim e 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 functio ns.

In my view, this coursework played an important part in learning software engineeri ng. At first, I know nothing practical about agile software development. After readin g 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 j obs to proper members, all of us worked effectively and efficiently.

2.2.4 Xiaoqian HUANG_130804992_JP2013213507

Piece of Work	Relevant outcomes
✓ Discussed the functions and the construction of the management system. ✓ Analyzed the framework of the system. ✓ Be responsible for designing the algorithm and programming the control classes and entity classes. ✓ Tested part of the classes using Junit. ✓ Wrote part of the report in "Desgin and Analysis .	✓ Codes of control class and entity class - AssignDriver, AssignTrain, ScheduleRoute , ScheduleJourney, ControlOnBoardScreen, ControlStopScreen, ControlSynchronize, Train, Driver, Route and Journey. ✓ Junit codes – AssignTrain , ScheduleJourney ✓ Descriptions and Diagrams of the classes – 5.1, 5.2, 5.3 in the report.

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
✓ Design the GUI frame, including welcome menu, synchronize, on-board screen and stop screen frame. Besides, write code to achieve these functions. ✓ Write some reports, including 4.1 and 4.2.	✓ Report : 4.1.2 , 4.1.3 and 4.2 ✓ Code: welcome class, menu class, synchronizes class, on-board class and stop class.

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
✓ Implement the function of read and update the information from file, write the class of OnBoard and OnScreen. ✓ Write report with 6.1.	✓ Report : 6.1 ✓ Code: Class of OnBoard and OnScreen. Implement of I/O of the system.

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 and 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

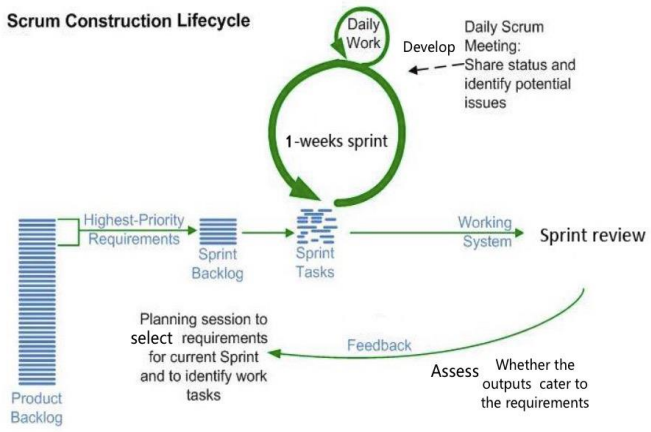
  In this part, the operation mangers are able to start or stop the train remotely with the help of the system.

- ✓ 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 priory 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 moths, 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) trough 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	Priority	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

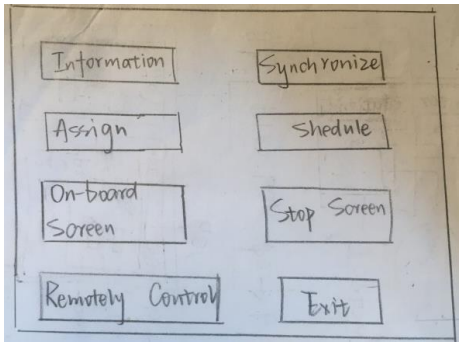
4.4 Software Prototypes

4.4.1 Logical user-interface design

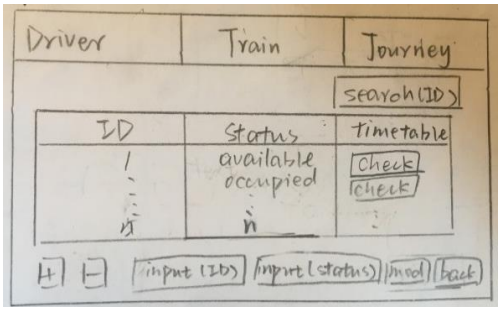
- ✓ **Welcome Menu**: Welcome Menu is the first page of the system interface. The operation manager can go to any functions of the system by clicking the specific button in this page.
- ✓ **Information**: The information page offers the operation manager a way to maintain the details and status information 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 serval 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 chose 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 statu 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 it's 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

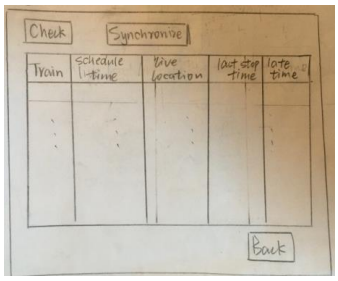
✓ Welcome Menu:



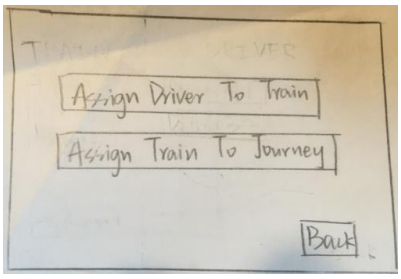
✓ Information:



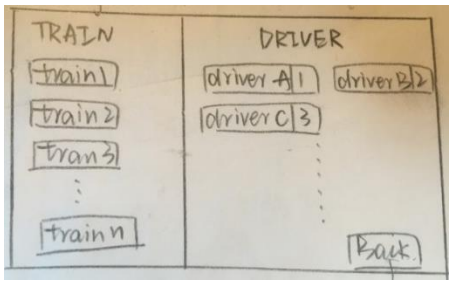
✓ Synchronize:



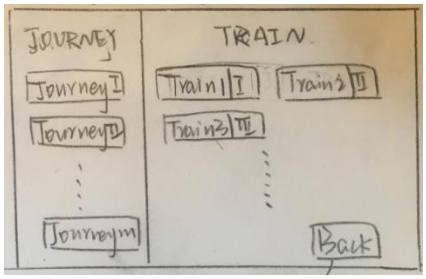
✓ Assignment:



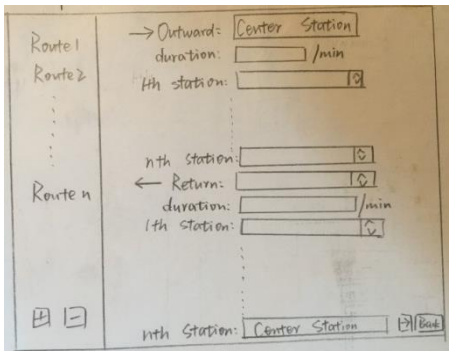
✓ Assign driver to train



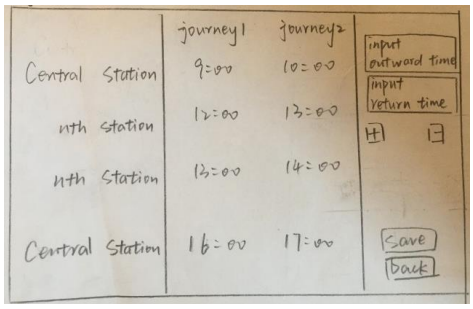
✓ Assign train to journey



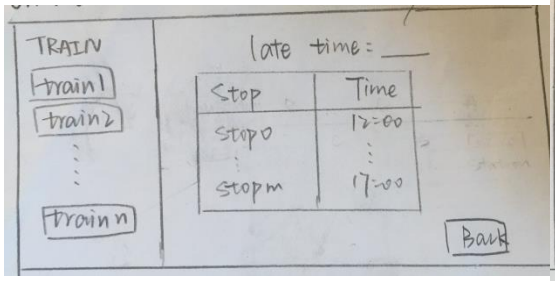
✓ Schedule: Select stops for route



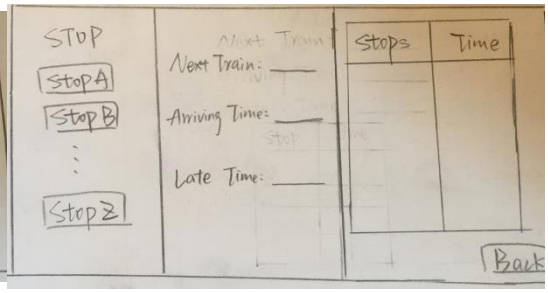
✓ Schedule: Select journey for route



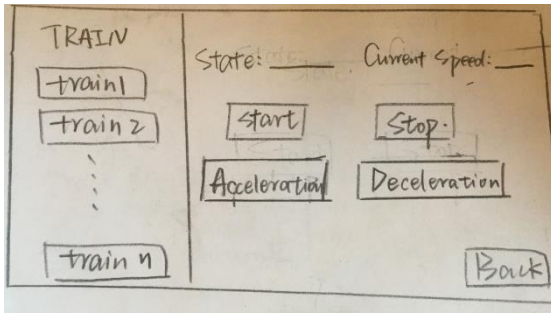
✓ On-board screen: for tuorist



✓ Stop screen: for tourist



✓ Remotely Control



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	✓ Deciding all necessary user and fundamental functions of the system. ✓ Making group schedule and establishing milestones. ✓ Iteration of stories with aspects of user, developer and companies. ✓ Conclude the details of requirements. ✓ Dividing our group members into three subgroups and separating specific tasks of each group.	All group members
Iteration 2 【3.29-4.15】 GUI design in java	✓ Design the software prototype. ✓ Modify the prototype by concerning about the detail of requirements and convenience of users. ✓ Realize those GUI interfaces in JAVA.	3 group members
Iteration 3 【4.16-4.29】 Function realization	✓ Realize the serval functions of the system such as assignment, schedule, management and synchronize. ✓ Realize other additional functions such like displaying timetable of stop screen and train screen.	2 group members
Iteration 4 【4.30-5.7】 System completion and data Implement	✓ Combine all classes together and make the whole system run without error. ✓ Design implication strategy and built plan. ✓ Finish the implementation by separate the whole system of serval parts.	2 group members
Iteration 5 【5.7-5.14】 Test of the whole system	✓ Test the system part and part. ✓ Maximize the fault tolerant. ✓ Detecting the errors of system and revise them. ✓ Finish the last iteration of the stories to improve the functions thoroughly.	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

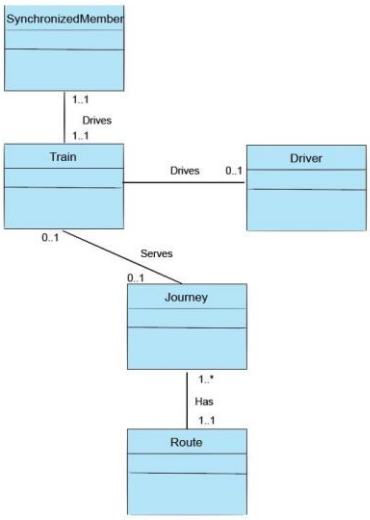
5 Analysis and Design

5.1 Analysis and Architecture Design

Analysis classes	Class	Detail
Boundary class: *(We use Train class to define all the attributes of train.)*	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.
Entity class: 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.
Control class: 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

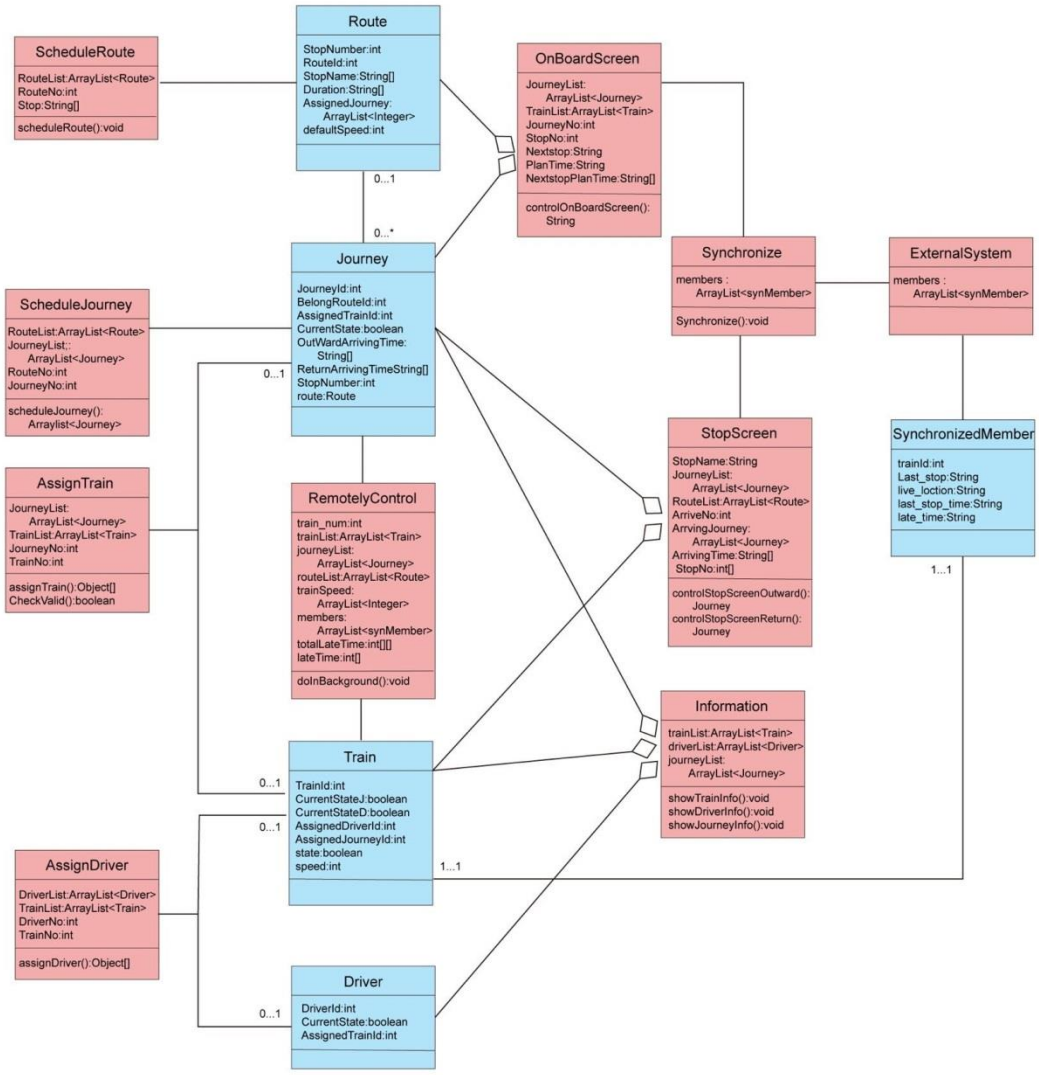


5.3 Re-usability of system components

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

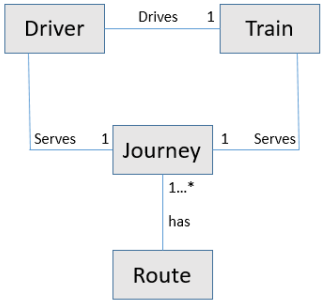
6.1.2.1 Integration Build Plan

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.

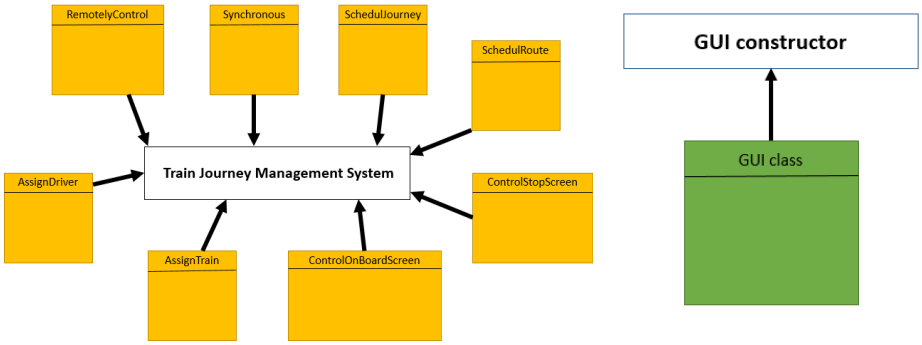
The GUI is used for better interaction between users and system. It also meets our expectation as first. The entity classes define the attributes of important things and some methods. The kernel of the system is the control classes. The operation of users will produce some data that to be record and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

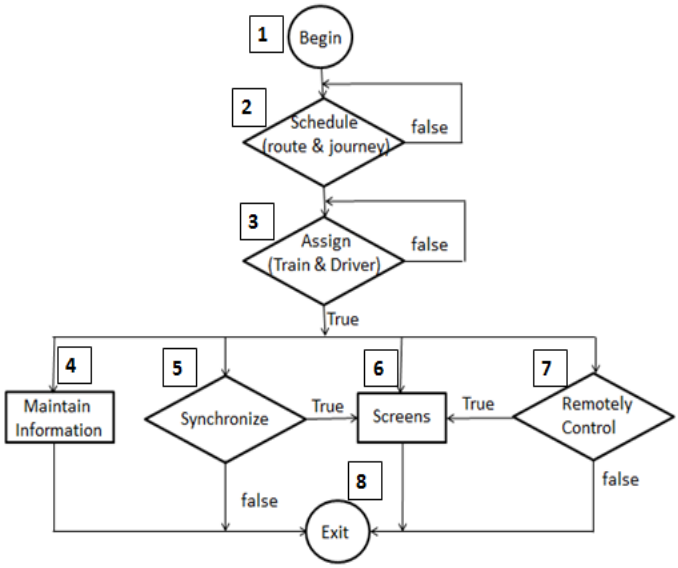
- ✓ This part is to answer the question how to determine whether the testing effort is successful
- ✓ Success criteria – 85% of test cases passed and no high priority defects unsolved.

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 waring 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.

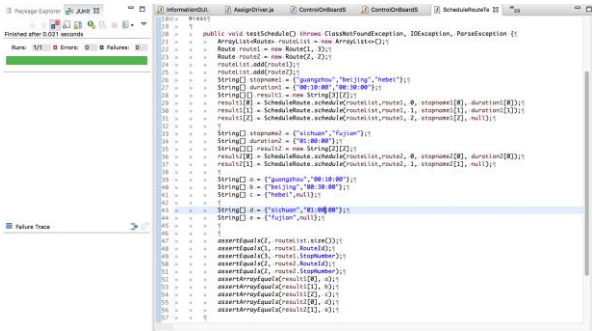
6.2.2.2 Object-Oriented Testing

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

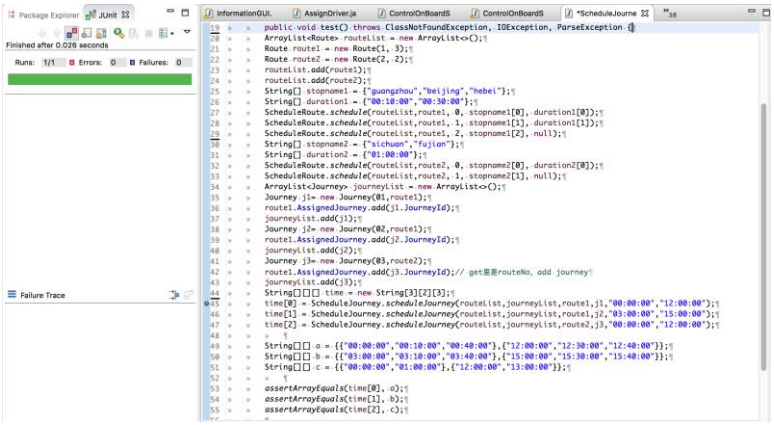
- ✓ **Schedule route**

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.



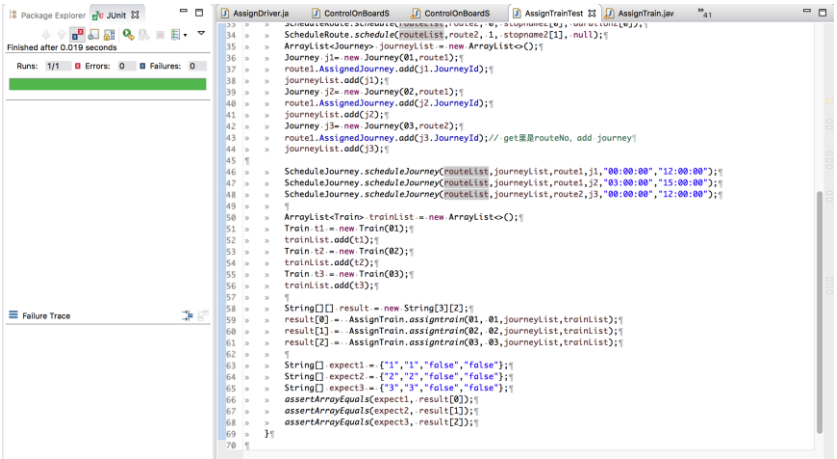
- ✓ **Schedule journey**

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.



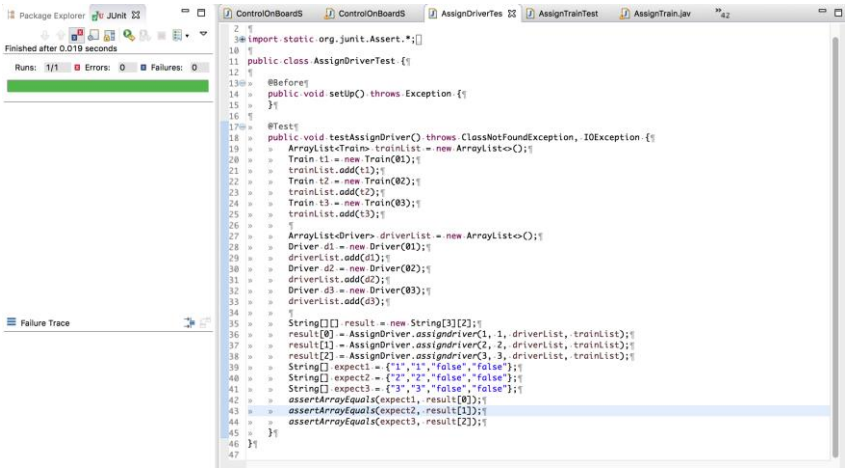
- ✓ **Assign train:**

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.



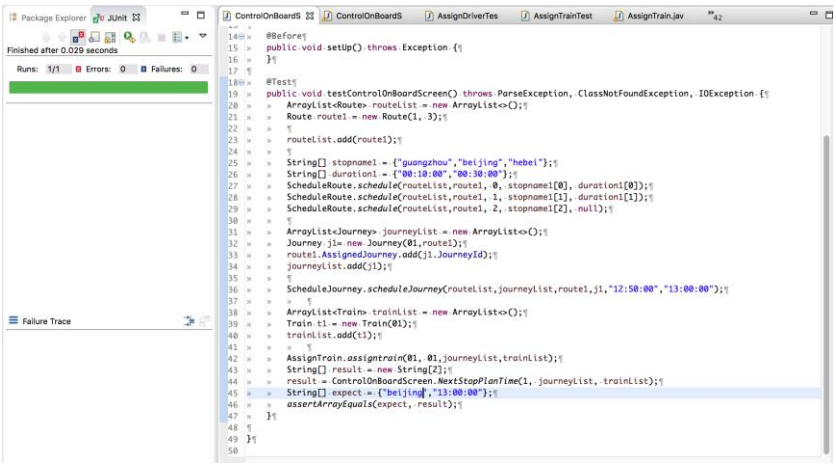
✓ **Assign driver:**

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.



✓ **Control on board:**

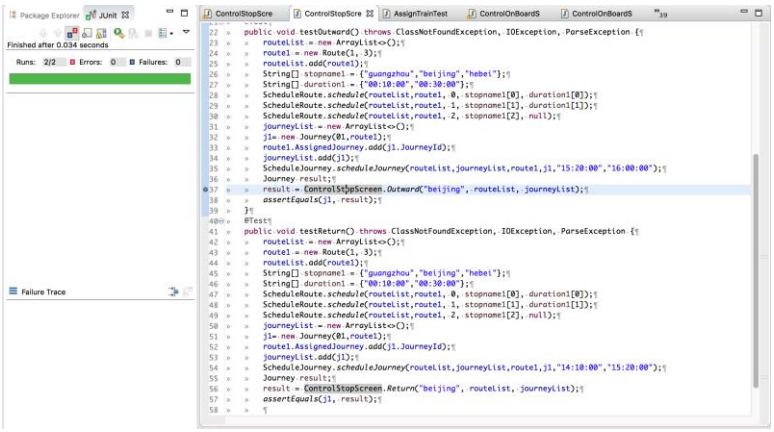
Once the train is on its journey, it should display the information about the journey it assigned on the board so that the passenger can check where they are and the next stops. In the test, the train is assigned to journey, which will stop at “beijing” at 13:00:00. Since the time we execute this test is 12:58:00, we expect to see that after invoke NextStopPlanTime(), the result it return is “Beijing—13:00:00”.



✓ **Control stop screen:**

Since there will be several trains on the routes at the same time, the stop screen should display the information about the next arriving train and its arriving time. We separate the stop screen to the outward and the return according to the real experience. In this test, we try to track the trains about to arriving at “Beijing” station. And after checking the routeList

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.



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