Andrew Hinh

Landon Maupin

Sprint 2 Submission

1.Sprint 1 Corrections: This includes the Sprint 1 correction, as well as a list of changes we made at the bottom.

IT 261 Agile Sprint 1
Abandoned House Ranchers - Landon Maupin & Andrew Hinh

2. Verbal update, 1-2 page single space memo

Our goal is to create a system that can handle reading and writing hours worked by employees. The system should be able to recall information saved in the past at any time as long as you request it from a date. In the system, there is the Employee class. Each employee has a unique identifier known as the employee ID. ID's are used by the manager which is an extension of employee. In order to keep ID's unique to each employee, an ArrayList of employee objects is used. The index of the employee in the ArrayList will be their unique identifier. This also means that employees with the same name is ensured different ID's along with different jobs and wages. Each employee carries their name, position name, job description, and an hourly wage. Employees are only able to read their recorded times, not write. If an hour is recorded incorrectly, it is up to them to contact a manager to correct it.

Managers have all the attributes of an employee of employee ID, job name, position, and wage; however they have added responsibilities than an employee. They are responsible for adding new employees into the system. Managers are expected to correct any mistakes for employee's entered hours as well as giving employees raises when needed. Managers are also responsible for creating multiple Schedules for employees to work on certain days. They are able to print recorded hours that the system has for any employee and print paychecks accordingly. The system can handle paying overtime after 40 hours in a week. After a paycheck is printed, the employee's hours for that week is reset back to 0, however the record of that employee is saved on a document. That same document contains employee name, ID, hours worked, and date.

Schedules are created by managers, and are identified by three key attributes: employee number, the date, and which Shift the employee will take. Once created, they will be appended into the document for future access. This document must be appended rather than overwritten every time a Schedule object is called. PrintStream is used for he needs of the company to append files rather than FileWriter. Schedules have the option to read from this document using the Java Scanner and String Tokenizer classes. It will be able to read from the same format it is to write in, from left to right: Employee ID, employee name, the date, and hours worked that day. The hours worked will be in decimal format rather than carry two separate integers of minutes

and hours. This is easier to understand for the user on the paycheck while being easier to follow on code.

Finally, a Shift object is identified by a shift number, and will determine how many hours an employee is supposed to work, along with which hours of the day they will work. Shift ID's are created unique similarly to how Employees are unique. They are given an ArrayList that holds Shift objects, in which the corresponding index is the ID. Using a shift ID, along with an employee ID and date will let you access hours worked by which employee from the Schedule objects.

4. Essential Use Case List

-Here is a use case description table, which shows the actors for each specific use case, as well as what type of event is relevant to the system.

Use Case	Type of event	Description
Punch In	External	All employees are able to enter when they start their work for a given day.
Punch Out	External	All employees are able to enter what time they finish that shift for that given day.
View hours in pay period	External	All employees have access to view how many hours they have worked on their upcoming paycheck, which can be a maximum of two weeks.
Add new employee	Internal	A manager can add an employee to the system and assign them an ID. From there, any manager can schedule them for work.
Correct punch hours	Internal	A manager can look for hours worked in the system and correct the number of hours worked in a given day with an employee ID.
Create schedule	Interal	A manager is able to create a schedule for an employee to work given an employee ID, the date, and the shift they are supposed to work.
Record weekly hours	Temporal	The system will save the number of hours worked by each employee after they have entered in the actual time they have worked.
Print hours	External	Managers have access to recalling all hours worked on any given day for the company.
Reset hours	External	Managers can call the system to reset hours worked in a given week for all employees.
Set wages	Internal	Managers can update any employees wages to a specified amount.

5. System vision with UI dialog

-This section basically tells what the project is all about. It describes the company, its goals, its problems, and its goal on how to solve those problems. It also gives an idea of what the proposed solution is capable of and how it can benefit the company.

Acme Bottling Company, Consolidated

System Vision Document

Problem Description:

Acme Bottling Company has many problems with its clock in/out system. First, employees have to physically punch in/out on an old machine that has a less-than-functional clock, and it doesn't print the times correctly half the time because the clock is slow. Second, at the beginning of a shift, it takes employees a longer-than-normal time to all clock in due to the time it takes to line up the ink stamp on the time cards. Third, it's the twentieth century, and the President has recently mentioned increasing efforts to make the company more technologically advanced; a recent spike in demand and profits has inspired Acme to become faster and, overall, more efficient. Many efforts have been made as far as production is concerned, but the power of Acme lies in its work force. Plans have been made to implement an electronic clock in/out system. This system will significantly speed up the process of clocking in and out. Employees will now swipe a personal identification card instead of manually punching a clock. More importantly, the managers' hours will be decreased dramatically. A new system for keeping track of hours will also be included in this electronic time card system, and reports of employee's hours will be printed out on a computer, rather than having the manager manually look at all the time cards and calculate the amount of time worked for each employee. Acme has decided to hire the Abandoned House Ranchers, a small independent consulting firm, to implement this new system.

System Capabilities:

- Manager will be able to correct hours for any employee. This will be useful, for instance, if an employee clocks in/out at the incorrect time, or if the employee forgets to clock in/out altogether.
- Manager is able to create a new employee
- Manager will be able to set employees' wages, or make changes to them
- Supports real-time tracking of employees
- The system is capable of accumulating hours at any given time
- System allows user flexible and easy entry of data

Benefits to Plant Operation:

- Reports on hours will be quickly and easily generated, thus saving the manager valuable time
- Employees will be able to quickly clock in/out, also saving valuable time
- Reports will be neat an organized, allowing managers to work more efficiently
- Less time will be wasted on things like clocking in/out, manually counting and calculating hours and pay, which will save costs
- Plant production will increase significantly, which will allow the plant to meet demands and increase profits.
- Plant will become more technologically advanced, leading to other more advanced projects

6. Work Breakdown:

-This outlines how, exactly, each task of our project was tackled by each of its respective solution team members.

Landon

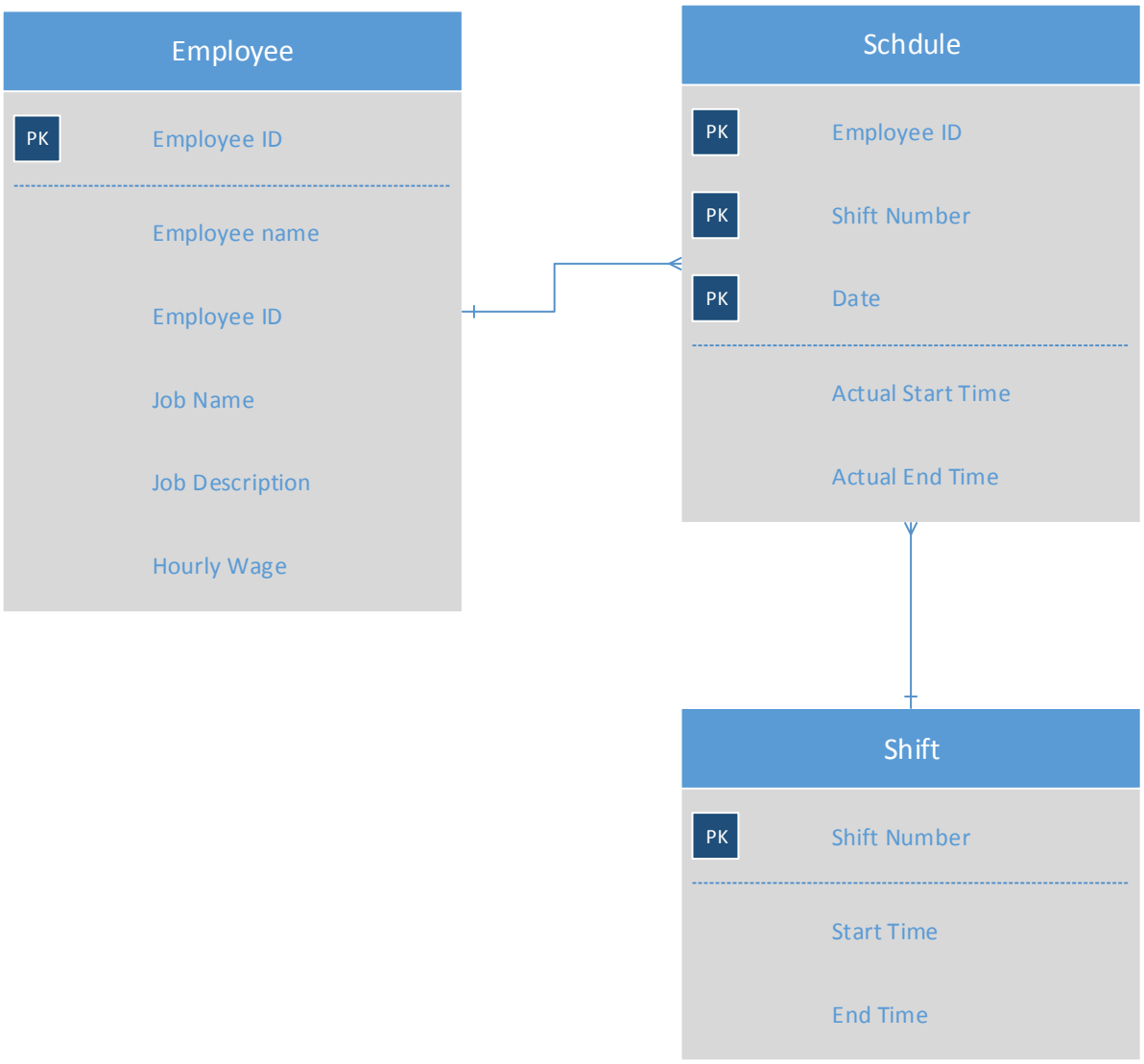
- Contributed to the creation and design of the system sequence diagrams
- Created use cases including a use case description
- Contributed to creation of Java code
- Created and drew problem domain class diagram
- Created system vision document

Andrew

- Helped design and implement Java code, including read and write capabilities
- Created and drew problem domain class diagram
- Created and drew activity diagram
- Created 2 page updated memo

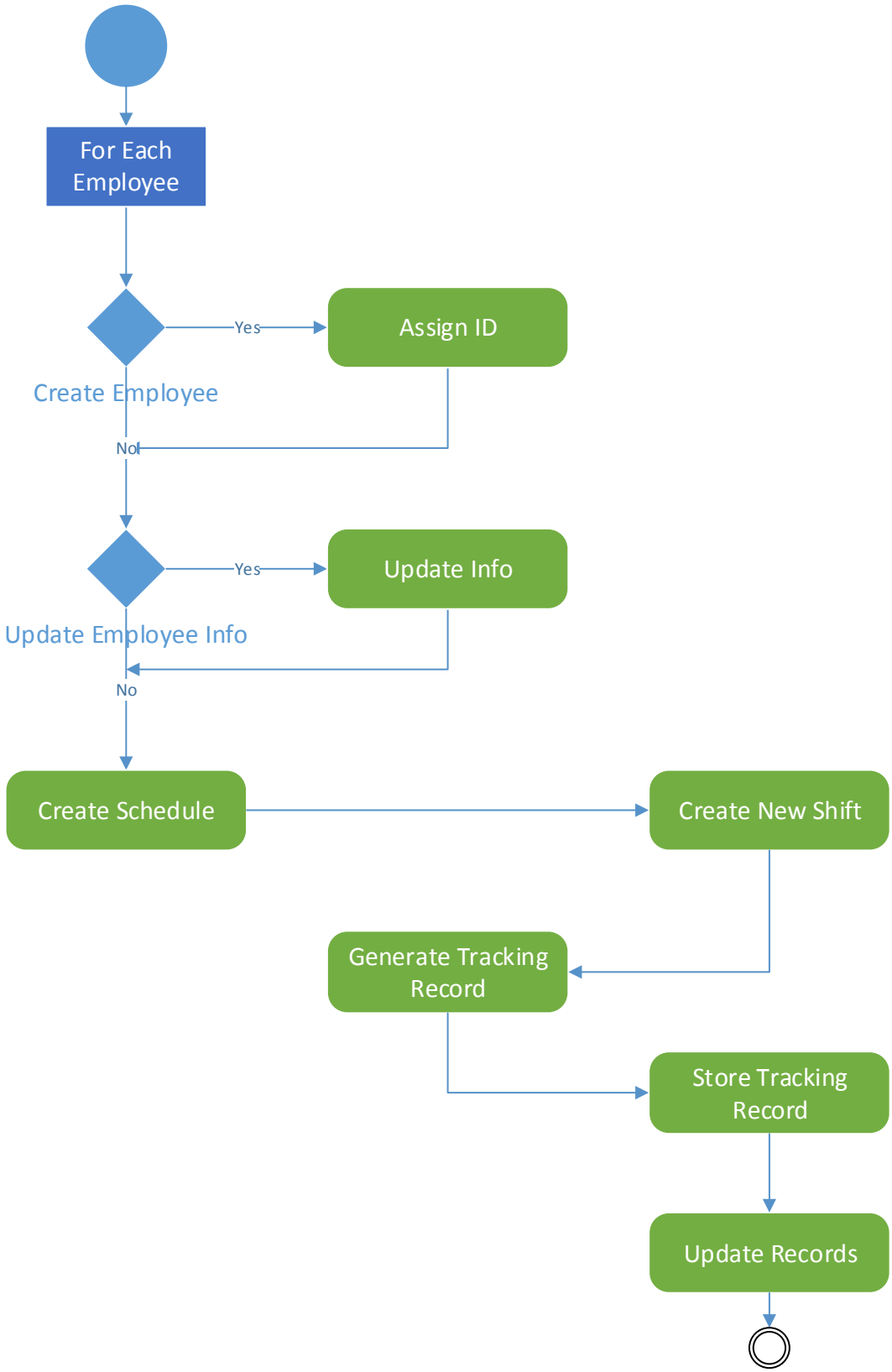
7. Problem Domain Class Diagram

-In our domain class diagram, we see how attributes and operations of the different classes are related to each other.



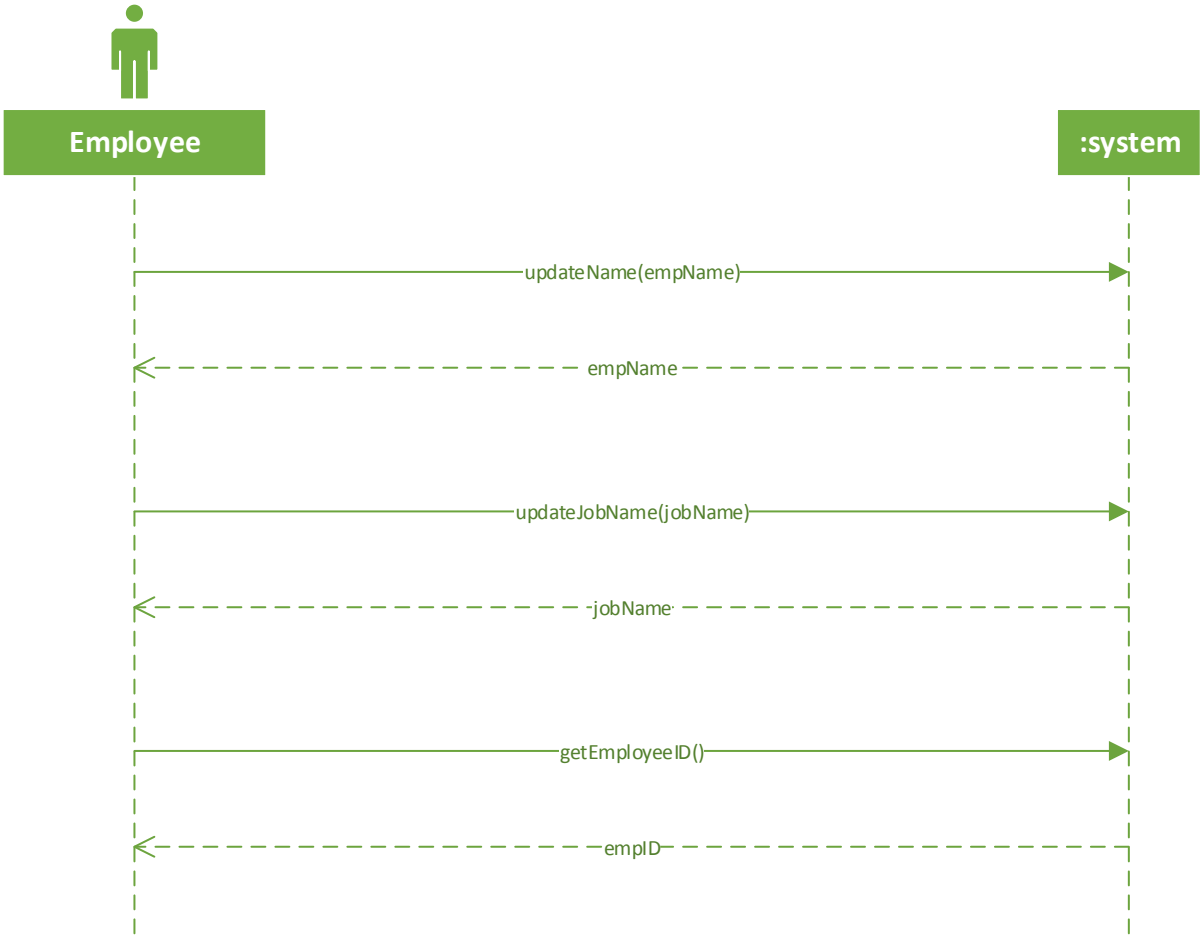
8. Activity Diagram

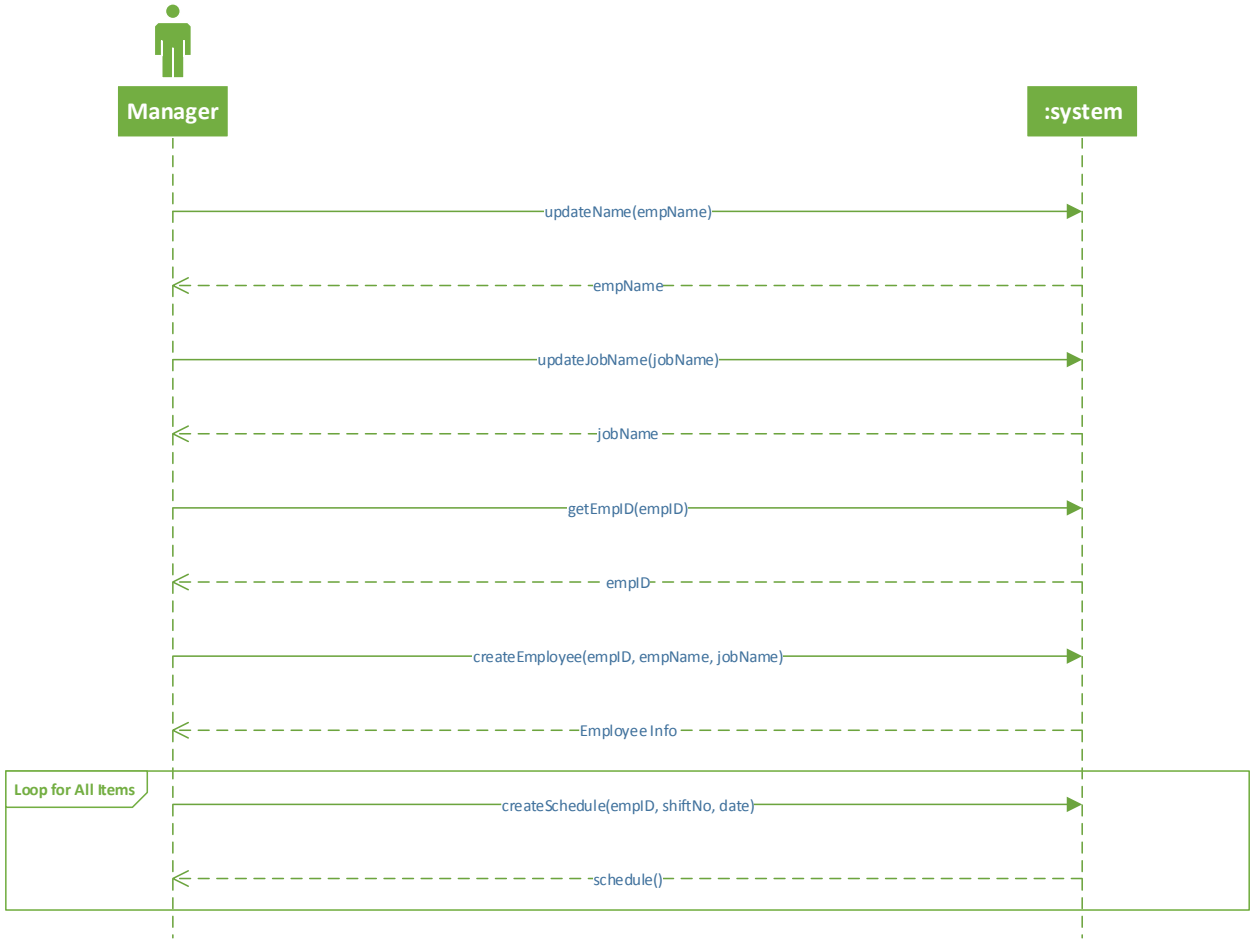
-This diagram gives a rundown of how the activities play out in the overall process. It gives the order, and if certain activities are done at the same time as others, it displays them as such vertically.


For Each Employee
Create Employee
Yes
Assign ID
No
Update Employee Info
Yes
Update Info
No
Create Schedule
Create New Shift
Generate Tracking Record
Store Tracking Record
Update Records

9. System Sequence Diagram

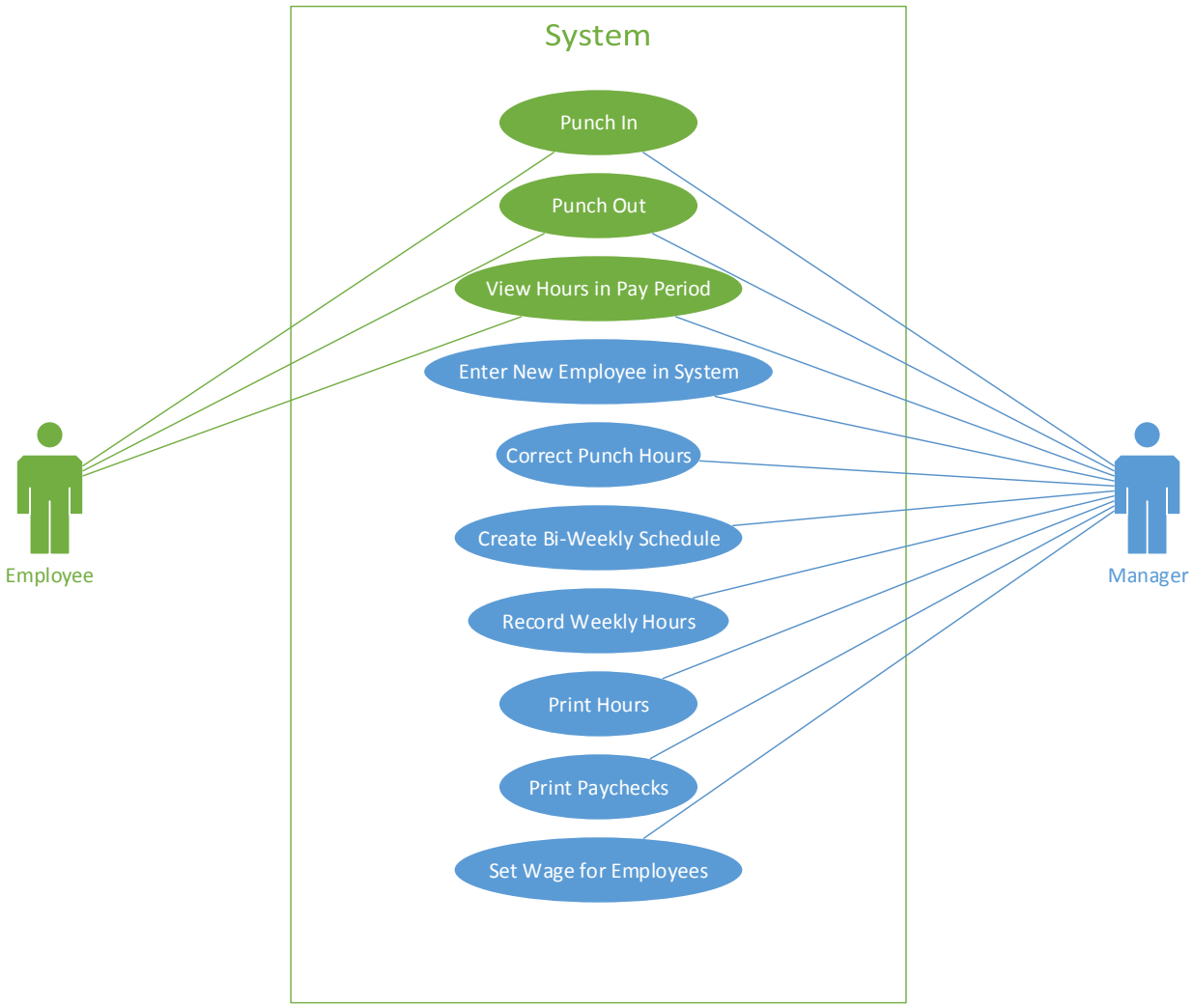
-This diagram show the methods between the actor and the system, and also what is returned. Each method the actor calls is a use case from the essential use case diagram.




Manager
:system
updateName(empName)
empName
updateJobName(jobName)
jobName
getEmpID(empID)
empID
createEmployee(empID, empName, jobName)
Employee Info
Loop for All Items
createSchedule(empID, shiftNo, date)
schedule()

10. Use case diagram + use case description

-This diagram shows the actors and the use cases.



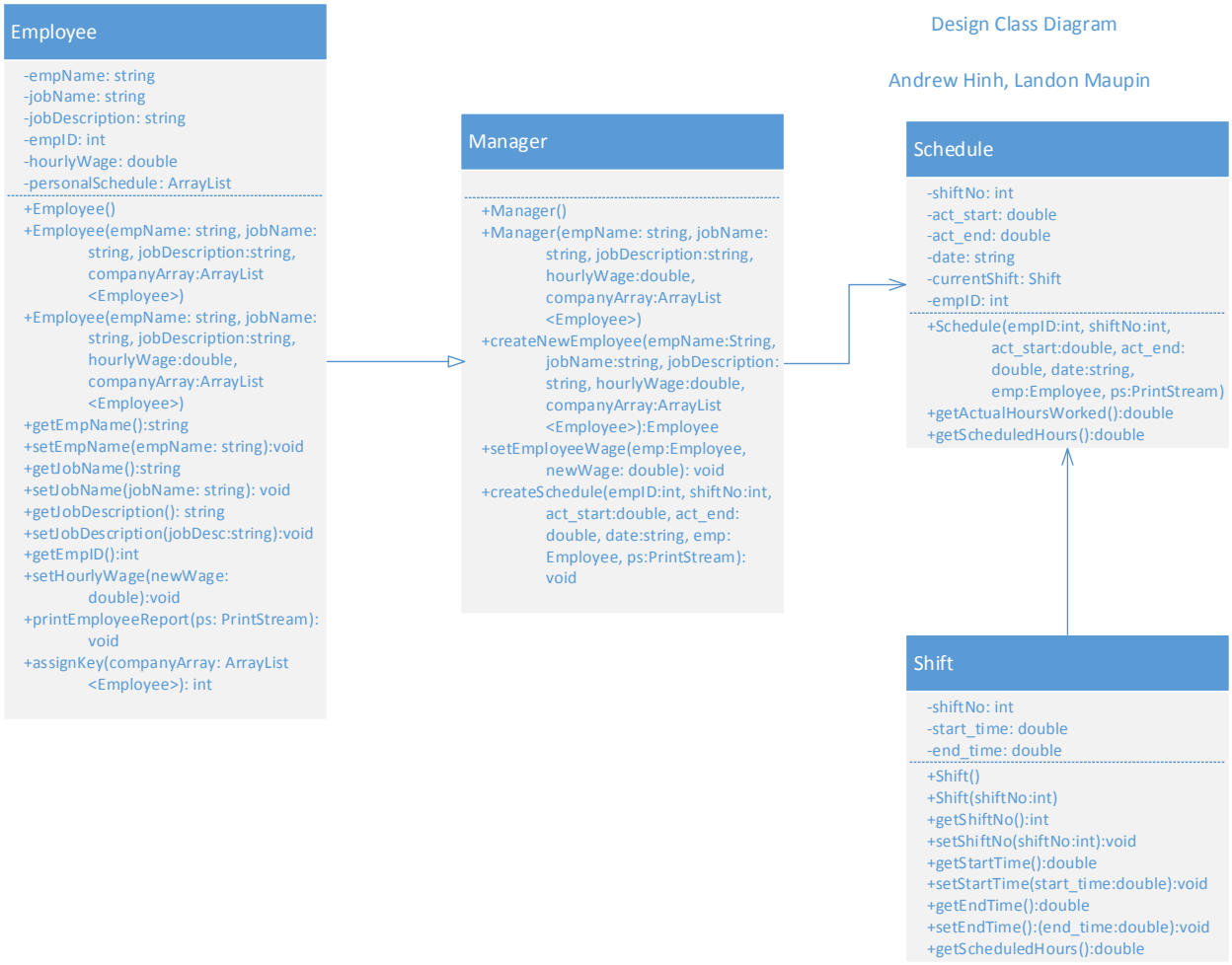
END OF SPRINT 1 CORRECTIONS

List of Sprint 1 Corrections

- Added introductions to each section
- Corrected the Domain Class Diagram
- Corrected the Activity Diagram
- Added used case descriptions to the SSD

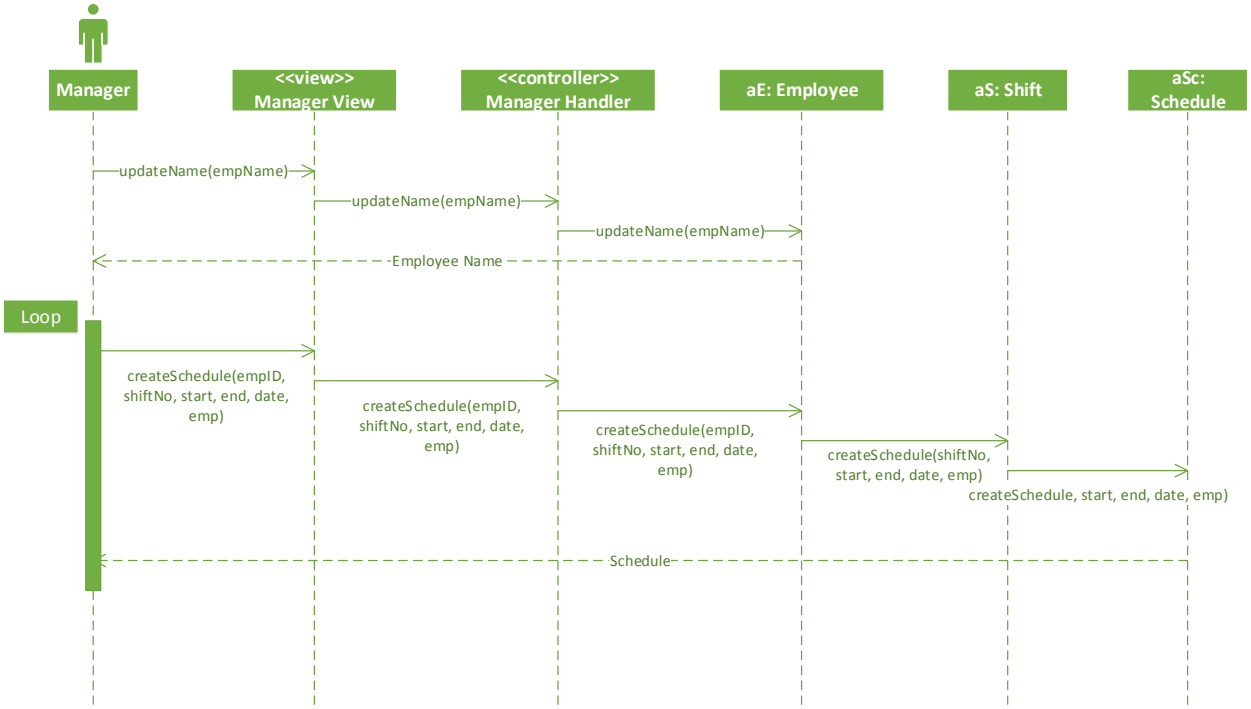
2. Design Class

- This shows the methods and attributes for all classes in our system.



3. Sequence Diagrams

-These diagrams are based on the design class and show how the data is accessed within the system as well as showing the view layers.



7. Completion Report

Save button implemented for user to determine when to save.

We have completed all 6 objectives that we originally planned from sprint 1 as well as having additional features implemented in the design. We added the ability to have the manager remove employees from the system without modifying the database directly (i.e. changing the .txt file), but instead using the user interface methods available. We also added a save button that wasn't planned from sprint 1 to let the manager decide when the employee list should be saved. The schedule will automatically be saved because in the event that taxes need to be filed, employees that worked there must be accounted for, even if they do not work there anymore. However, if there were changes done to the employees that weren't meant to be done, the manager can close the application without saving.

Plans for next iteration

Plans for an undo and redo feature can be implemented by creating two stacks: the 'undo' stack contains every single user interaction with the system leading up to opening that program. The second stack, redo, contains anything popped from the 'undo' stack. If you are in the middle of calling 'undo' but then make a change that wasn't part of the 'redo' stack, then the 'redo' stack becomes unavailable until you 'undo' once more. 'Redo' is only available immediately after calling 'undo'.

Team Reflection

Landon and I found it difficult to make time to meet up to work on the project since we had different schedules. Given this, we had to split up work and then use the little time we had together to either put our work together and discuss what we need to change/what we have to do next. The second sprint was more interesting in the parts where we had creative liberties while at the same time, have requirements to meet from sprint 1 objectives. The annotations from the sprint 1 submission for corrections was unclear, so we made assumptions as to what Dr. Hosack would like, trying to follow to the most correct submission.