Documentation
Java and Software Final Project

Alli Jones and Aidan Edwards

April 24, 2018

Table of Contents

1. Outline
2. Gantt Diagram
3. Vision Statement, Requirements, Buisness Rules
4. Use Cases 1 - 4
5. Use Cases 5 - 6
6. Use Cases 7 - 8
7. Use Case 0
8. Use Case Diagram
9. Domain Model
10. System Dequence Diagram
11. System Sequence Diagram Continued
12. System Operations and Contracts
13. Design Model
14. Sequence Diagram 1: Saving Data to the Database
15. Sequence Diagram 2: Loading Test Data
16. Sequence Diagram 3: Clearing Data
17. Design Patterns Used (Explained further in comments in code)

Link to the GitHub: https://github.com/allijones/Java-Project

Alli Jones and Aidan Edwards
3/15/2018

Java and Software Project Outline

Goal: Make a weekly schedule for a restaurant

1. Layer 1: A database with two tables:

   A table with employees and corresponding ID numbers

101	Name
102	Name
103	Name

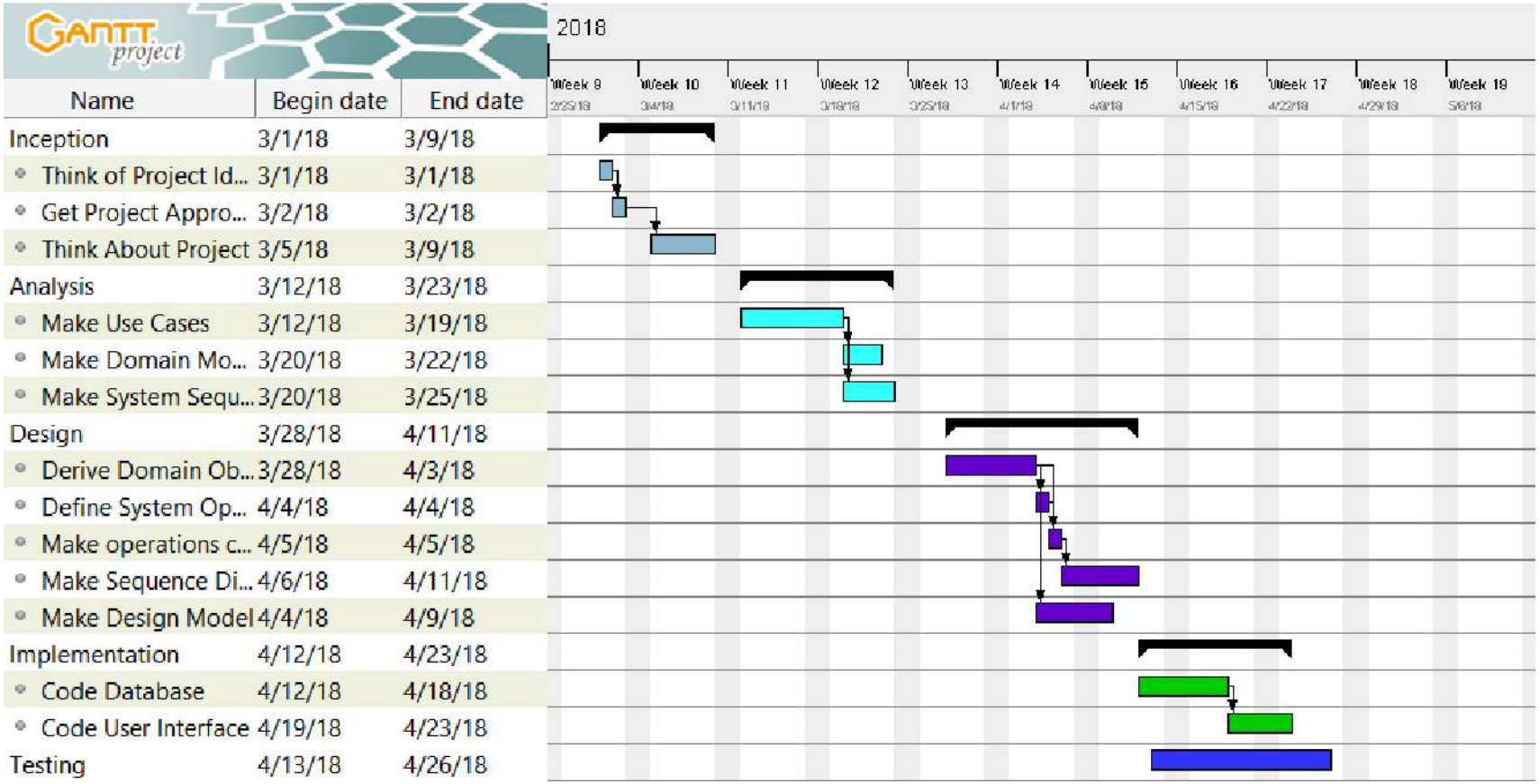
   A table with IDs of employees corresponding to the days of the week that they are available to work.

101	Mon, Tues, Wed, Thurs, Sat
102	Mon, Tues, Thurs, Sun
103	Mon, Thurs, Sun

2. Layer 2: Business Layer. This layer will have business rules such as:
   a) Each day of the final schedule needs to have at least 3 employees working that day
   b) Employees may not be removed from the schedule once they are scheduled
   c) The system will support reading from and writing to CSV files only

3. Layer 3: Presentation Layer. This layer will be the user-interface layer. The program will make a schedule based on the database and the business rules and print it to a file. The user will be able to interact with it by:
   a) Assigning shifts to employees
   b) Adding an employee to an already existing schedule
   c) Printing the entire database of employees to see each employee and what days they are available
   d) If the schedule violates the business rules, the user will be notified
   e) Goal table:

Names:	Mon	Tues	Wed	Thurs	Fri	Sat	Sun
E1	X	X	X	X		X	
E2	X	X		X			X
E3			X		X		X
E4	X			X			
E5			X		X	X	X
E6		X			X	X	



Name	Begin date	End date
Inception	3/1/18	3/9/18
Think of Project Id...	3/1/18	3/1/18
Get Project Appro...	3/2/18	3/2/18
Think About Project	3/5/18	3/9/18
Analysis	3/12/18	3/23/18
Make Use Cases	3/12/18	3/19/18
Make Domain Mo...	3/20/18	3/22/18
Make System Sequ...	3/20/18	3/25/18
Design	3/28/18	4/11/18
Derive Domain Ob...	3/28/18	4/3/18
Define System Op...	4/4/18	4/4/18
Make operations c...	4/5/18	4/5/18
Make Sequence Di...	4/6/18	4/11/18
Make Design Model	4/4/18	4/9/18
Implementation	4/12/18	4/23/18
Code Database	4/12/18	4/18/18
Code User Interface	4/19/18	4/23/18
Testing	4/13/18	4/26/18

Use Cases
Java and Software Project

Alli Jones and Aidan Edwards

April 23, 2018

Vision Statement

The goal of this project is to create a schedule-maker that can be used in a restruant or retail store to help the managers of the store create weekly schedules for their employees. The goal is not to create a schedule, but to design an interface that has access to a databse of current employees that will be able to interact with the manager as he/she creates the weekly schedule, or updates an exisiting schedule. The schedule-maker will provide the manager with lists of employee names and their ID number for each day, allowing the manager to easily select employees to work on every day of the week. After assisting the manager in creating or updating a schedule, this software application will send the schedule to a CSV file so that the manager can access it easily, and print it easily.

Requirements

1. Access a database of current employees
2. Provide options to create a schedule
3. Provide option to update an existing schedule
4. Send schedule to CSV file
5. Display all employees in the system

Business Rules

1. There must be at least 3 employees working on every day of the week
2. The Schedule will be read from and written to CSV files only
3. Once an employee is scheduled, he/she cannot be removed from the schedule

Use Case 1: Loading Data

1. System provides user with option to "Load test data" of employees
2. User selects option
3. System loads the employee data

Use Case 2: Loading Data from CSV

1. System provides option to load new employee data from CSV
2. User selects option
3. System prompts user to input the name of the CSV file
4. User inputs file name
5. System opens that CSV file and loads the data

   Alternative Paths:

   5* CSV file does not exist

   5.1 System informs user that file was invalid

   5.2 System provides option to load CSV file again

Use Case 3: Loading Data from SQL

1. System provides user with option to load data from SQL database
2. User selects option
3. System loads data from SQL

Use Case 4: Displaying all Employees

1. System provides user with option to display all employees in the database
2. User selects option
3. System displays chart of all emloyees and when they are available

Use Case 5: Creating a Schedule

1. System provides user with option to create a schedule
2. User selects option
3. System automatically starts with Sunday. System provides user with a list of employee names and ID numbers that are avalaible to work on current day.
4. System asks user to select and enter 3 employee ID numbers to fill that day. (System will not allow user to continue to the next day without selecting at leaset 3.)
5. User selects and enters at least 3 employee ID numbers into the system.
6. When 3 have been entered, user now has the option to continue to the next day.
7. User selects option to continue.
8. System then repeats this same process for Tuesday-Saturday.
9. After each day is filled, the system asks the user to enter a name for the CSV file that the schedule will be printed to
10. User enters a name for their CSV file
11. System creates CSV file and writes data to it
12. System returns to main menu

    Alternative Paths:

    3. *No employees are avalaible to work on current

    3.1 System informs user that no employees can work on that day and asks user if he/she would like to continue to Tuesday or quit the process

    3.1.A User selects the option to continue making the schedule and the system continues normally

    3.1.B User selects the option to quit the process and the system returns to the main menu

Use Case 6: Adding an Employee to the Schedule

1. System provides user with option to update existing schedule
2. User selects option
3. System asks user to enter the title of the CSV file that he/she want to update
4. User enters the CSV file name

5. Staring with Sunday, system provides user with a list of employee names and ID numbers that are avalaible to work on current day.
6. User can select an employee to add to that day, or select an option "next" that will take him/her to the next day.
7. Process repeats for days Monday-Saturday
8. System asks user to enter a title for the newly updated CSV file
9. User enters title
10. System creates new CSV file
11. System returns to main menu

Use Case 7: Saving Loaded Data to SQL

1. System provides user with option to "save data"
2. User selects option
3. System saves data to the SQL database

   (Additional Note: This assigns employees an ID number! Prior to saving to the SQL database, all employee ID numbers are zero.)

Use Case 8: Clear Loaded Data

1. System provides user with option to clear the previously loaded employee data
2. User selects option
3. Loaded data is cleared. New data can now be loaded (see use cases 1, 2, and 3).

Use Case 0: Choosing Interface

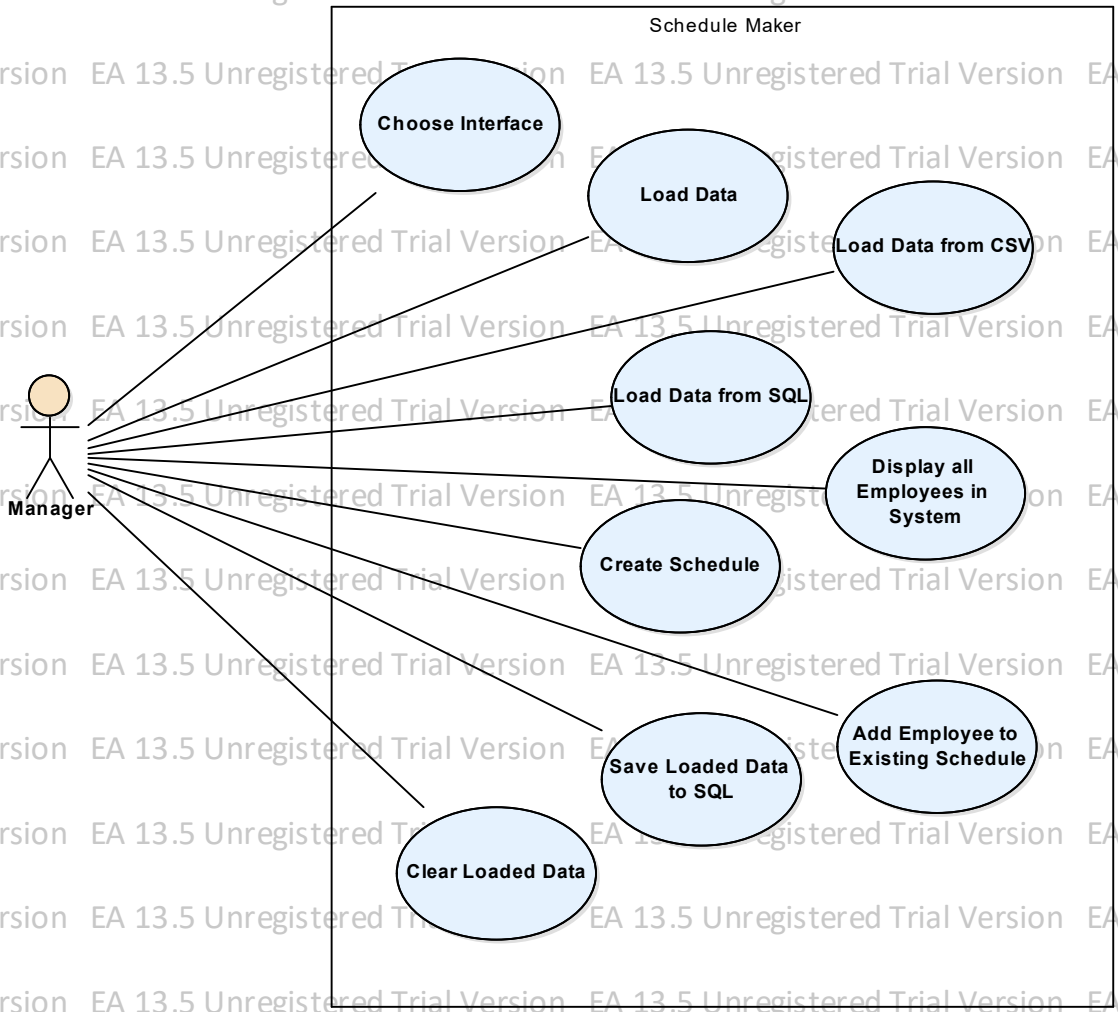
1. System provides option to use GUI interface
2. User selects option
3. System will now use GUI interface

   Alternative Paths:

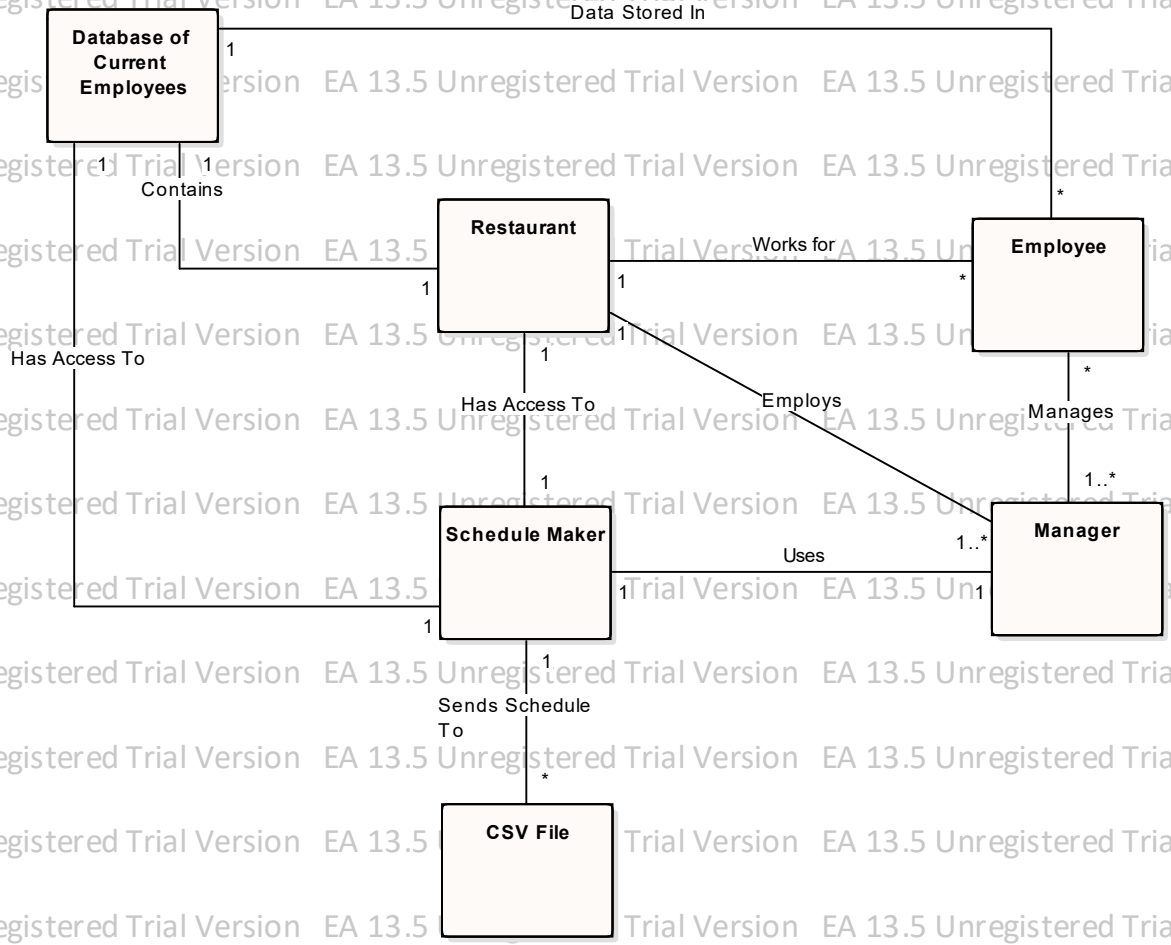
   2* User enters invalid input

   2.1 System repeats question until user enters valid input

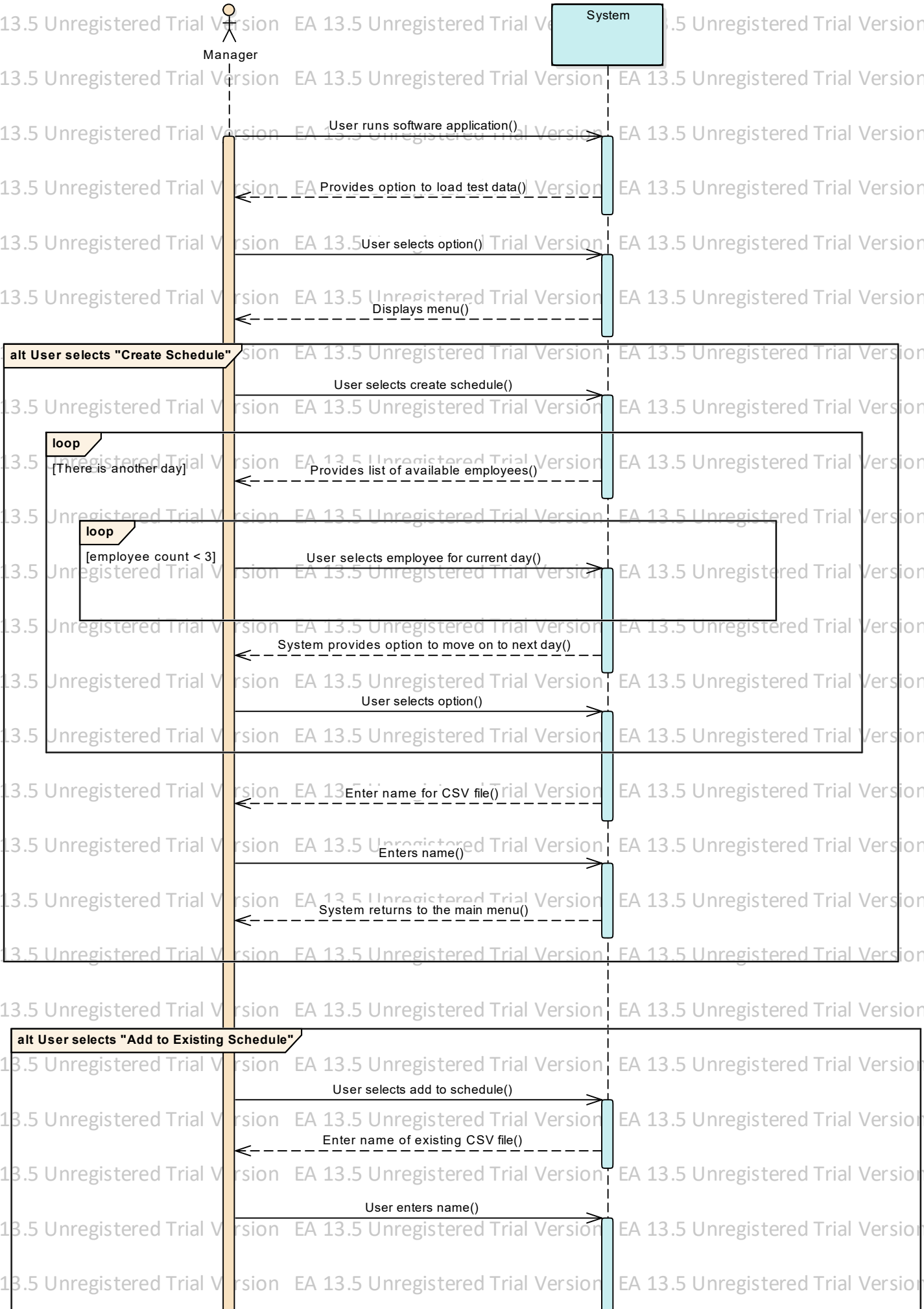
EA 13.5 Unregistered Trial Version

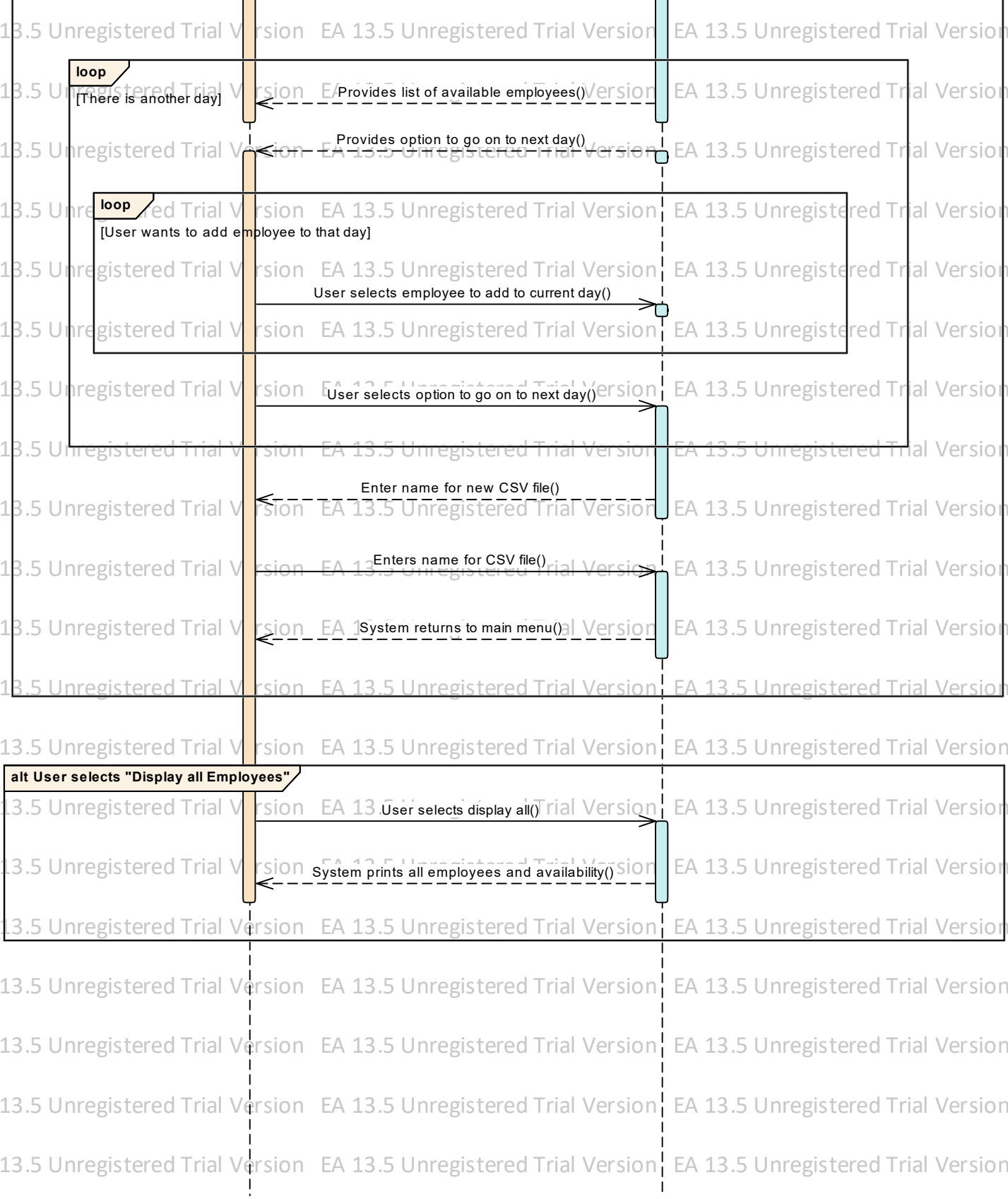


EA 13.5 Unregistered Trial Version



EA 13.5 Unregistered Trial Version





EA 13.5 Unregistered Trial Version

EA 13.5 Unregistered Trial Version

System Operations

+ availableDay(): ArrayList<Integer>
+ buildSchedule(): void
+ getId(): int
+ getName(): String
+ getSchedule(): Boolean[ ]
+ isAvailable(): Boolean
+ loadAll(): Map<Integer, String>
+ loadAll(): Map<Integer, Boolean>
+ run(): void
+ save(): void
+ save(): void

buildSchdule( ): void

Functionality for building a schedule
Prints final schedule to CSV

save( ): void

saving days to the days database

getId( ): int

Part of the Employee class
returns the ID number of an employee

run( ): void

Prints the menu

isAvailable( ): Boolean

Part of the Employee class
Returns if an employee is available for a specific day

save( ): void

saving employees to the employees database

loadAll( ): Map<Integer, String>

Retrieves employee info from the database

getName( ): String

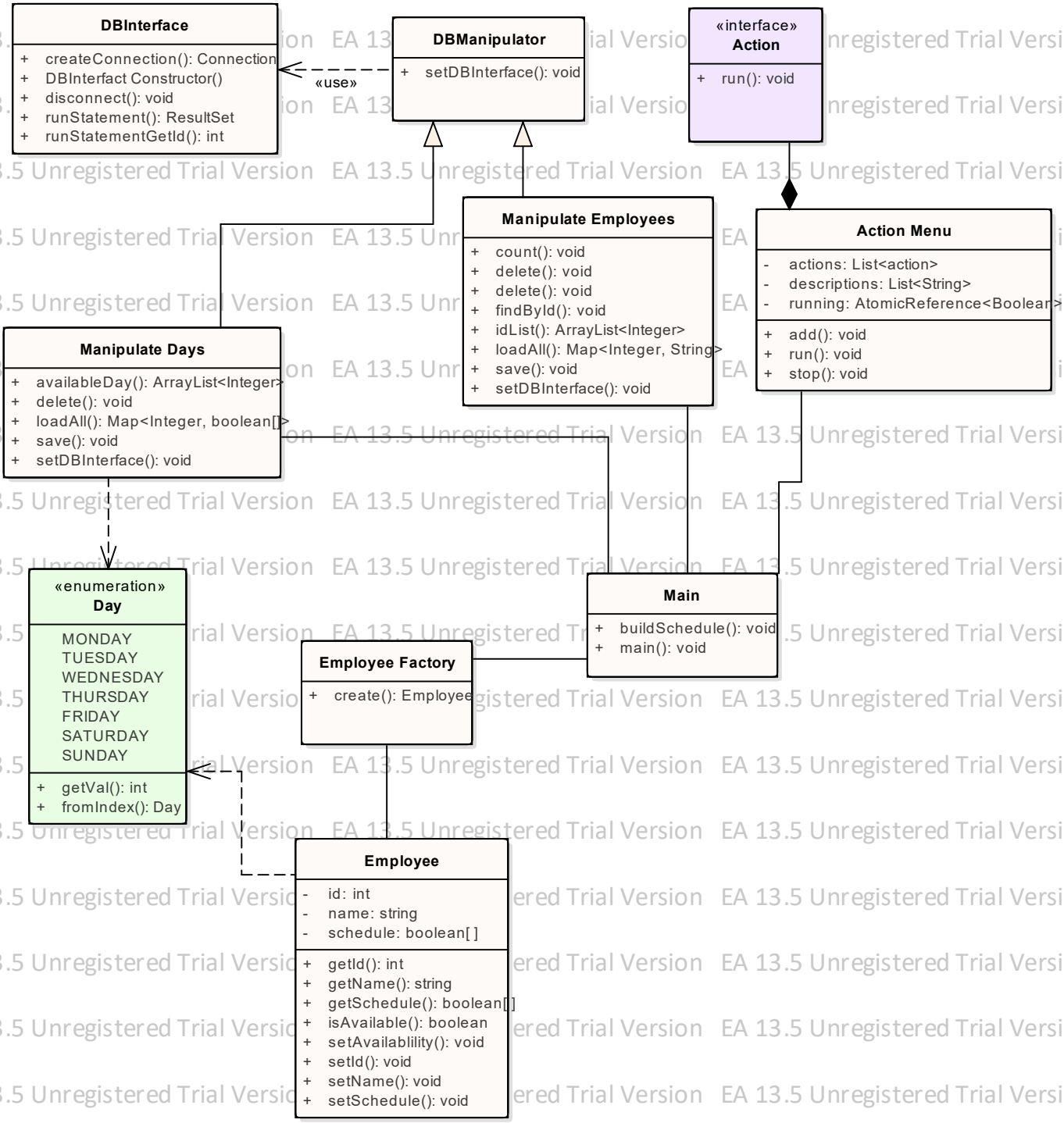
Part of the Employee class
Returns the name of an employee

loadAll( ): Map<Integer, Boolean>

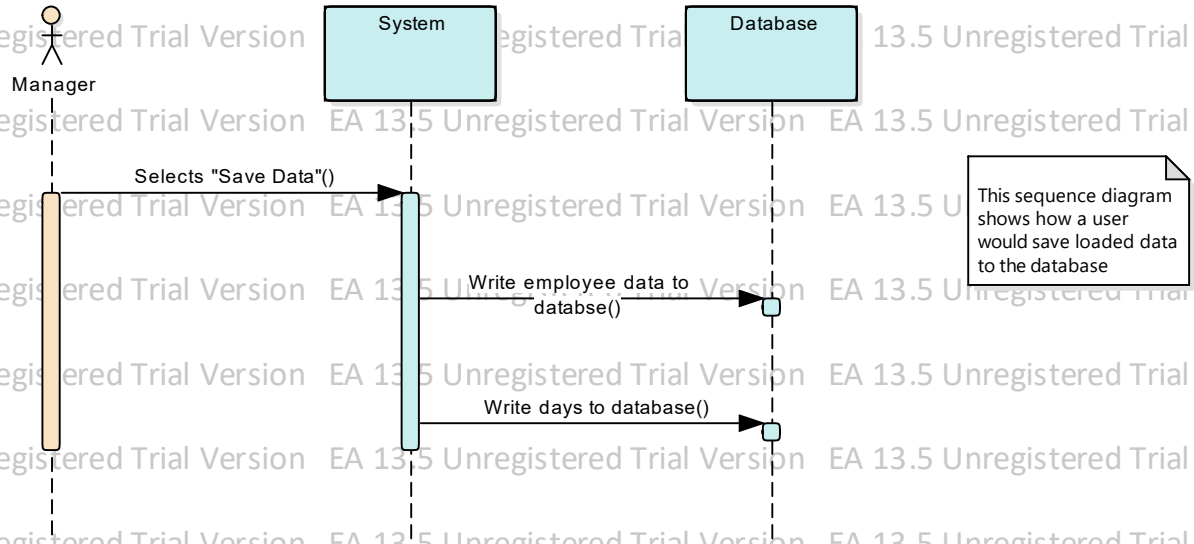
Retrieves schedule of an employee from the days database

getSchedule( ): Boolean[ ]

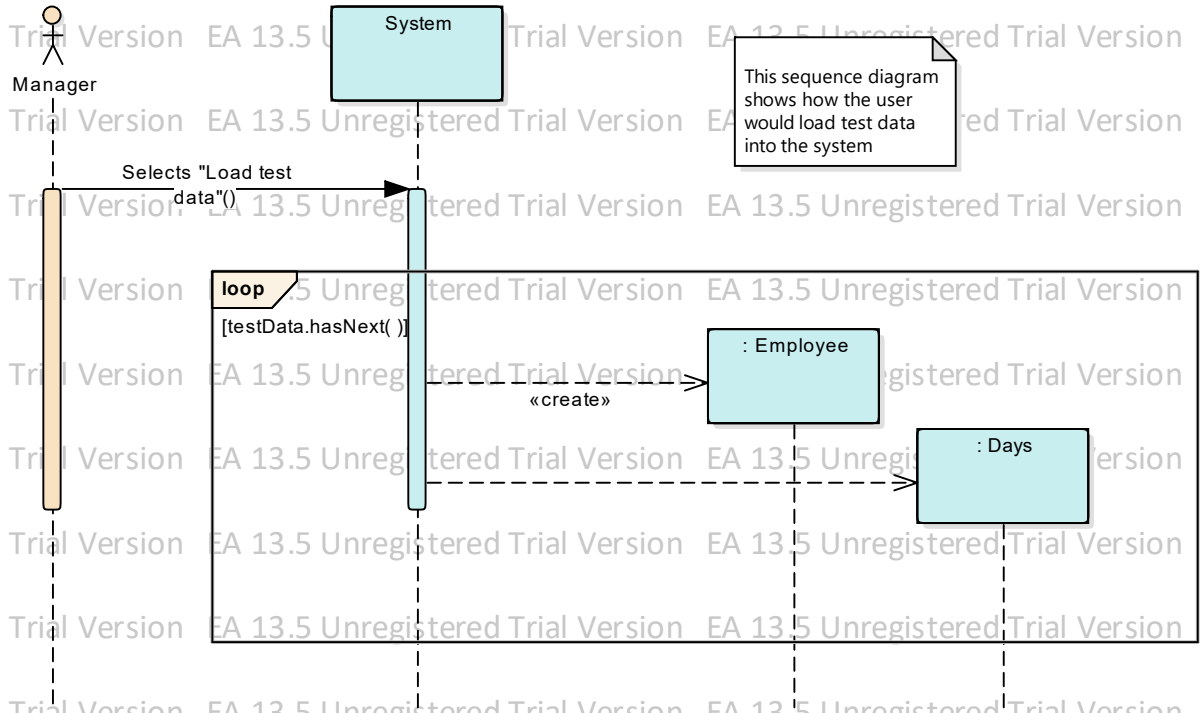
Part of the Employee class
Returns an array of booleans for when an employee is available to work


DBInterface
+ createConnection(): Connection
+ DBInterfact Constructor()
+ disconnect(): void
+ runStatement(): ResultSet
+ runStatementGetId(): int
«use»
DBManipulator
+ setDBInterface(): void
«interface»
Action
+ run(): void
Manipulate Employees
+ count(): void
+ delete(): void
+ delete(): void
+ findById(): void
+ idList(): ArrayList<Integer>
+ loadAll(): Map<Integer, String>
+ save(): void
+ setDBInterface(): void
Action Menu
- actions: List<action>
- descriptions: List<String>
- running: AtomicReference<Boolean>
+ add(): void
+ run(): void
+ stop(): void
Manipulate Days
+ availableDay(): ArrayList<Integer>
+ delete(): void
+ loadAll(): Map<Integer, boolean[]>
+ save(): void
+ setDBInterface(): void
«enumeration»
Day
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
SUNDAY
+ getVal(): int
+ fromIndex(): Day
Main
+ buildSchedule(): void
+ main(): void
Employee Factory
+ create(): Employee
Employee
- id: int
- name: string
- schedule: boolean[]
+ getId(): int
+ getName(): string
+ getSchedule(): boolean[]
+ isAvailable(): boolean
+ setAvailablility(): void
+ setId(): void
+ setName(): void
+ setSchedule(): void

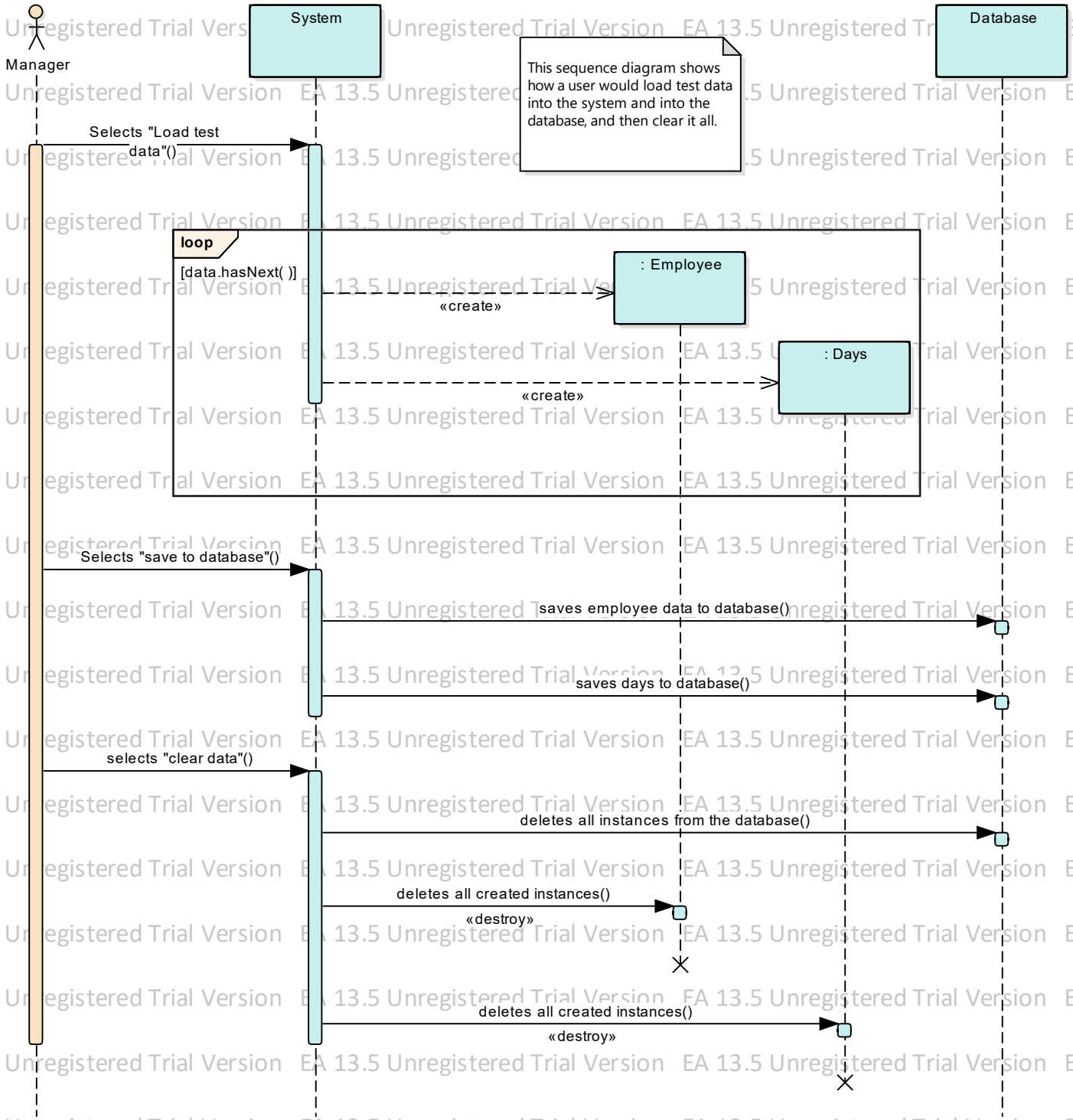
EA 13.5 Unregistered Trial Version



EA 13.5 Unregistered Trial Version



EA 13.5 Unregistered Trial Version



Design Patterns Used

1. Singleton: DBInterface
2. Factory: Employee Factory
3. Command: GUI Interface and action interface
4. Facade: DBInterface
5. Decorator: GUI Interface
6. Mediator: GUI Interface
7. State: Action Menu