Bilkent University

Department of Computer Engineering

Object-Oriented Software Engineering Term Project

TrackIn: Intern Tracking System

Design Report

**Group 3G:**

Muammer TAN, Ali Semi YENİMOL, I. Kayacan KAYA, Eniselda TUSKU

Contents

[1 Introduction 1](#_Toc496090066)

[1.1 Purpose of The System 1](#_Toc496090067)

[1.2 Design Goals 1](#_Toc496090068)

[2 Software Architecture 1](#_Toc496090069)

[2.1 Subsystem decomposition 1](#_Toc496090070)

[2.2 Hardware/software mapping 1](#_Toc496090071)

[2.3 Persistent data management 1](#_Toc496090072)

[2.4 Access control and security 1](#_Toc496090073)

[2.5 Boundary conditions 1](#_Toc496090074)

[3 Subsystem Services 1](#_Toc496090075)

[4 Low-level Design 1](#_Toc496090076)

[4.1 Object design trade-offs 1](#_Toc496090077)

[4.2 Final Object Design 1](#_Toc496090078)

[4.3 Packages 1](#_Toc496090079)

[4.4 Class Interfaces 1](#_Toc496090080)

[5 Glossary & References 1](#_Toc496090081)

Analysis Report

Project short-name: project title

# Introduction

TrackIn is a web based application that provides companies, interns and supervisors with an easier way to manage the internship procedure and facilitate the communication between them.

## Purpose of The System

The system aims to achieve the following features:

Task Management: Managing and tracking the tasks given to interns, see the status and workload, who supervises to which intern, announcing general events to interns.

Privacy: Providing the privacy for companies as they might not want to share all details and information with the interns.

Document Conversation: Keeping the files within the system to save files, documents integrated to a internship and eliminating possible document losses.

Flexibility: Providing the flexibility to companies each has its own requirements from interns and operations to do.

## Design Goals

**Usability:**

One of the most important design goals that we aim is to develop a user friendly program that will be easily used by the customer. Weather you enter our program as an intern, supervisor or a company, all the functionalities and the relation with other users will be clearly stated in the interface so that the user will not have any difficulties in understanding and putting his functions to use. Since this system intends to facilitate the internship tracking process the program will be as user friendly as possible in order to contribute to this purpose.

**Efficiency:**

We aim to make our program as efficient as possible so the user will not have any difficulties or delays while using it. We will try to guarantee a response time that will not exceed a certain threshold value. In this way the communication between the users, regarding the tasks or other elements will be fast and will not cause the user any concerns.

**Extensibility:**

Since we are developing a platform for different companies to keep track of their internship processes, it may later be extended and further developed so it can cover more functionalities and be adopted without any problem to certain company profiles. In order for this to be achieved we indeed to build a system that can be further modified and extended without causing any problems to the current one. This will involve building a system which is clear and very well structured.

**Reliability:**

We intend to design a system that will be reliable and stable and will not crush due to irrelevant inputs or actions from the user. In order for this to be achieved frequent test will be conducted to every part so that stability can be assured.

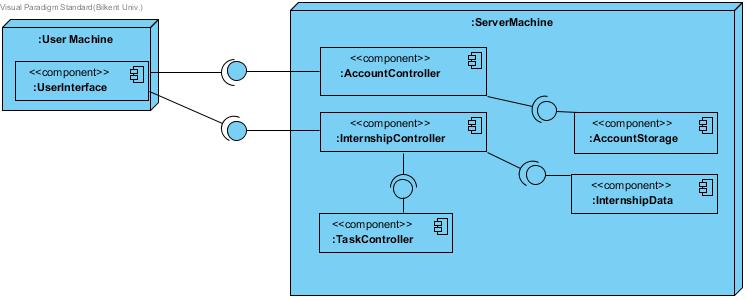
**Well defined interface:**

In order for the user to use the program as efficiently as possible and for him not to have any inconveniences, we aim to design a well defined interface which will not only make things easier for the user but for the developers as well. A well designed interface would create facilities.

# Software Architecture

## detailedsubsystemSubsystem decomposition

## Hardware/software mapping:



As TrackIn is proposed to be a web application, hardware/software mapping is important for the design. For efficiency goals, the system has Client/Server architecture as it can be seen from the Deployment Diagram above. The Server Virtual Machine must support the Java environment and as we decided to use Apache Tomcat for server-side implementation, the server also should support this library. For the AccountStorage and InternshipStorage systems, we are planning to use MySQL to construct the database. Hence the server must also have a support for MySQL. On the client side, a modern web browser is needed where PrimeFace Framework(for user-Interface) is compatible with. In addition, the user machine must obviously have a mouse and keyboard to input to the system.

## Persistent data management

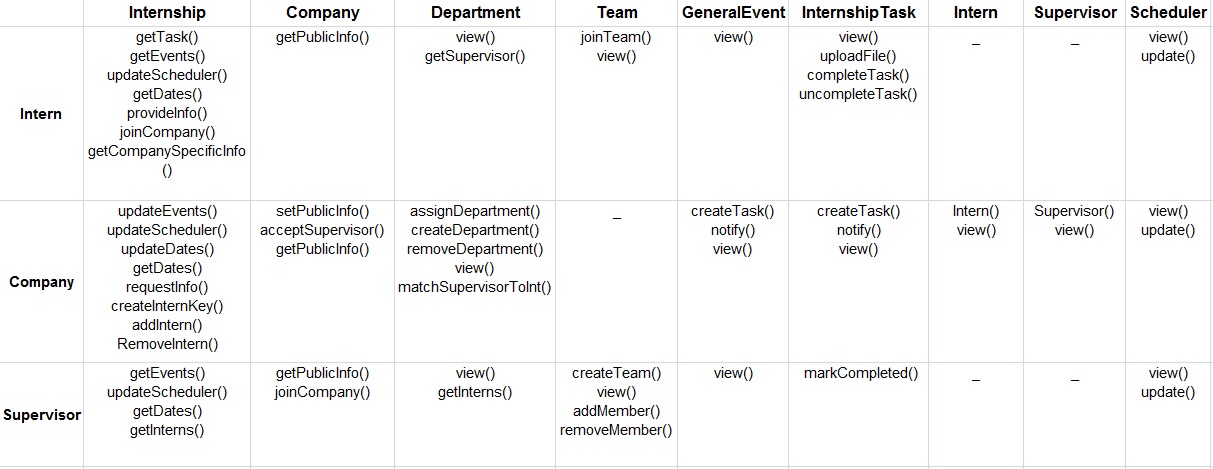
Data Management is one of the most important things we will deal with since we are going to implement a user-based web application. We are going to use MySQL to store personal information, username, password and user-related data of Intern, Supervisor and Company user types. Also we are going to keep events, tasks and other Internship related data as well. So, losing or manipulating all those data would be very easy if we would keep them in .txt files. Because of the concern of security and convenience, we are going to use MySQL database.

## Access control and security

TrackIn has 3 different types of users: Intern, Company, Supervisor. Each user has different functionalities within the system and should not act as one another. The central actor of the system is company, which is able to manage the other actors and their roles in the system and acts as a host whereas Intern and Supervisor actors are users in terms of hierarchical user pattern. That’s why, Interns and Supervisors must not interfere with company specific functionalities and each others’ functionalities. Also, the companies must not access other companies’ control rights. The restrictions for such access control mechanisms is provided with Controller classes inside subsystems.

Another concern will be security for user accounts. For that reason, we need to use proper cryptographic protocols such as hashing and authentication methods to ensure the security for possible cyber-attacks.

**Access Matrix:**



## Boundary conditions

* **Initialization:**

The system is initialized when the user launches the program …. When it is firstly initialized the initialization files provide the default state. Firstly the user will be faced with the logging in process.

* **Termination:**

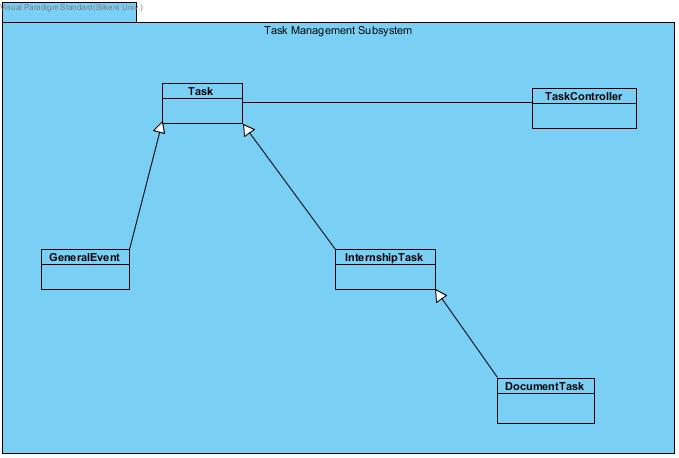
The program terminates when the user chooses Log Out of their account.

* **Failure:**

In case of a failure, since the information is saved in a database it will not be lost. However if there is information procession which has not been saved yet during the time when the failure occurs, the system experiences loss of information.

# Subsystem Services

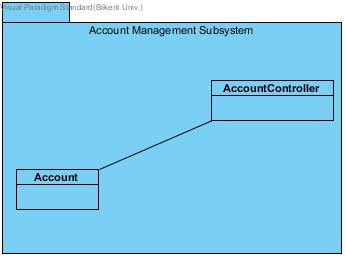
**TaskManagement**

This subsystem is responsible for Task related controls and acts as a inter-mediator between Database related to the storage of Tasks and Events. For easy-development purposes, Events are also designed to be a subclass of Task class within the system. When a company or supervisor adds a task to an intern, this subsystem will handle the types of the task and update the database and then notify the Internship Subsystem.

**Internship**

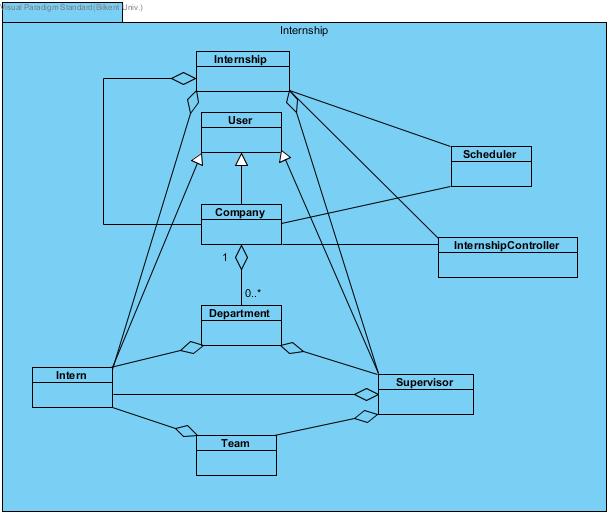
Internship Subsystem has the functionalities for the core mechanics of the internship procedure like relationship between interns, companies, supervisor, and teams. It is a central subsystem which communicates with TaskManagement and AccountManagement Subsystems. It is also responsible for UserInterface communication within the other subsystems.

**AccountManagement**

The role of the AccountManagement class is to control various types of users and provide the security of accounts. It is responsible for registration and log-in mechanisms and after a user enters to the system, it will active the Internship Subsystem according to the user type. Additionally, it is responsible for controlling the storage of the accounts.

**Internship Data**

Internship data subsystem will be the connection between MySQL and our Java codes. We will keep here functions that send MySQL queries to the DB Service. All the queries’ and functions’ main task is to keep the data of an internship such as lists, names, surnames, events, tasks, dates, calendar details etc.



**Account Directory**

Account Directory Subsystem’s main function is to send queries to MySQL DB Service to keep, update and change the data of user accounts such as name, surname, school, dates etc.

**User Interface:**

User interface subsystem is the subsystem responsible for all the graphical interface that occur during the program. It creates relations and exchanges data with all the other subsystem and manages the user interface of the program. It makes possible the user interaction with the system by taking input, returning data, displaying all the elements that the user needs and arranging the program according to their preferences.

* Registering:

The first view that appears when the program is initialized is the register Frame. Its primary aim is to provide user login according to his profile. Different panels are displayed if the user is signing up for the first time in which case he has to provide several information in order for his account to be activated, or if he is just signing up with an already existing account. This subsystem connects with the Account Management Subsystem in order for the corresponding profile to be displayed.

* Profiles:

After signing up according to the data collected from the user, the corresponding profile is displayed. This is made possible through the Account Management Subsystem which connects with the controller of the Internship subsystem and updates the user interface. Each profile is represented from a different frame and it contains the display of every element that composes that certain profile. Each user has access to different features so they include in their frames several panels. For example the supervisor also contains the “team” panel in which he can create and manage a team of interns. The company as an additional feature has “Matching Panel” where it makes possible the matching of supervisors with interns, etc.

# Low-level Design

## Object design trade-offs

**Response Time vs. Memory usage:** Memory usage is not that important in TrackIn system since it will not use the memory of the user's device. All the data will be held in database system and functionality of the program will be held in server. So, TrackIn will use only the RAM of user's device and hopefully it will not exceed the RAM of user's device. However, response time is really important for TrackIn system because it will be a web-based application and client will not want to wait when using TrackIn.

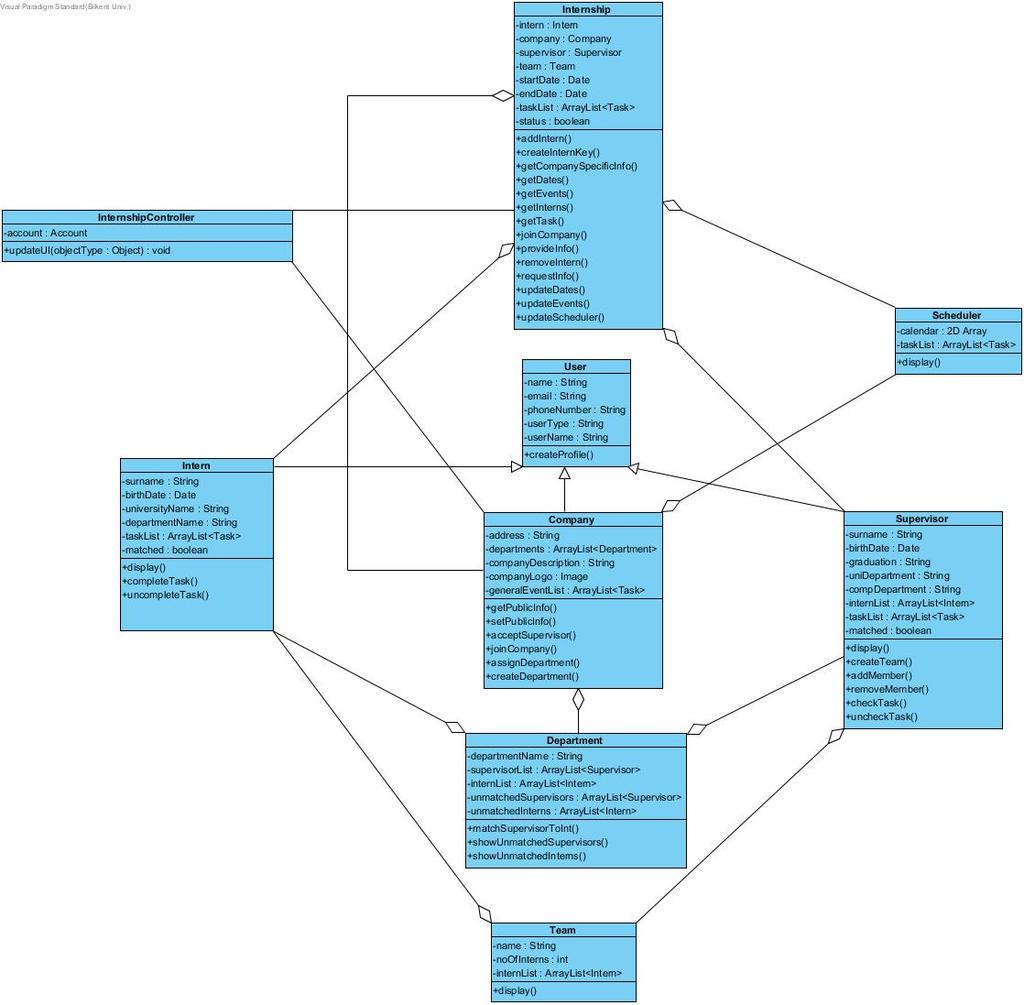
**Usability vs. Functionality:** Usability is a vital feature for our application because it is a business app and people will not want to use it if it is hard to use. Because of that we decided to use PrimeFaces Framework. But functionality is very important as well since again, it is a business app. So, this trade-off must be very balanced because these two features are very important for clients.

**Functionality vs. Difficulty of Code:** As it is mentioned above, functionality is very important for client. But it is nearly impossible for us to use all desired functional features since we, undergrad students, are not able to write that much functional and complex code. So we will do our best to make it closer to desired functionalities.

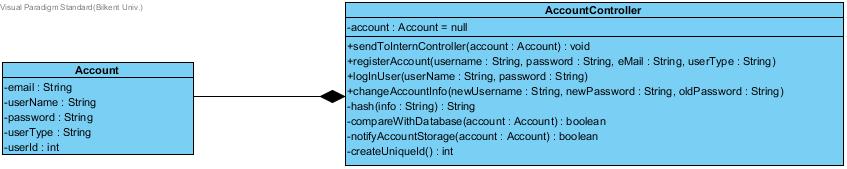
**Minimum Number of Errors vs. Functionality:** As functionality increases, the number of errors may increase because the code becomes more complicated and naturally becomes harder to handle. Since TrackIn is a web-based data storing application, a tiny error may cause huge problems such as losing, manipulating data, unwanted changes in internship process etc. Thus, minimum number of errors are more important than functionality of application.

## Final Object Design

* Internship Subsystem Classes



* Account Management Subsystem Classes



## task_management_subsystem_classes.jpgTask Management Subsystem Classes

## Packages

**4.3.1 View Package**

In this project, we are going to use JSF (Java Server Faces) Framework. So we are not going to use Java tools or packages for GUI (Graphical User Interface). Thus we are not going to have a view package.

**4.3.2 Controller Package**

Controller package's main role is to manage and the control the application. To manage and control the application, controller package uses five subsystems to manage internships, tasks, accounts, and database.

## Class Interfaces

* **AccountManagement Subsystem Classes**

**AccountController**

Controller class for the communication between UserInterface Subsytem, Internship Subsystem, and Database related to account information.

***Attributes:***

-account : Account - The Account object that this class is do the operations

***Methods:***

* +sendToInternController( Account)

Sends the required account information to Internship Subsystem that needs especially for the user type and user id for the operations

* +registerAccount(username : String, password : String, eMail : String, userType : String)

Creates a new account according to the given parameters, initializes the account member of the class, and activates the database(AccountData Subsystem) for storing it.

* +logInUser(userName : String, password : String)

Verifies the user information and initializes the account.

* +changeAccountInfo(newUsername : String, newPassword : String, oldPassword : String)

Make alterations on the account and activates the database(AccountData Subsystem) for storing it. newUsername is null if only password changes, newPassword is null if only username changes.

* -hash(info : String) : String

Used to hash private info like password and user name before initializing the account and storing it into the database

* -compareWithDatabase(account : Account) : boolean

Do the comparison if the user has the access to continue

* -notifyAccountStorage(account : Account) : Boolean

notifying the database for storage, called by registerAccount, logInUser, and changeAccountInfo methods

* -createUniqueId() : int

Creates a unique id for new accounts, called by registerAccount

**Account**

Entity class to hold account information

***Attributes:***

-email : String – Stores the e-mail adress of the account

-userName : String – Stores the user name as hashed

-password : String – Stores the password as hashed

-userType : String – Stores the userType, can be INTERN, SUPERVISOR, or COMPANY

-userId : int – Stores the unique user id

* **Task Management Subsystem** **Classes**

**Attributes:**

* task: Task – The Task class is an class which will
* internshipTack: InternshipTask – Is the class that will contain all the tasks that are related with the internship such as the intern’s assigned and completed work. It has properties like:

<checked> - Will denote if the supervisor has checked the intern’s work or not.

<startDate> - To denote the starting date of the task.

* documentTask: DocumentTask – Is the class that will contain all the tasks that are related with different documents needed during the internship procedure. It has properties like:

<File Type> - Which specifies the type of file that the user uploads.

<Max Document Size> - Specifies the max size that the uploaded document must have.

**Methods:**

* view() – Method used by intern, supervisor and company in order to view their tasks.
* uploadFile()- Method used by intern class in order to upload different files such as internship task files, documents etc.
* completeTask() – Method use by the intern class in order to mark the task as completed. This method, after the task is marked as completed, notifies the supervisor that the intern has completed his assigned task.
* uncompleteTask()- Method use by the intern class in order to mark a certain task as uncompleted. This method is needed in case the intern marks a task as completed by mistake
* markCompleted () – Method used by the supervisor in order to mark the intern’s task as completed.
* createTask() – Method used by the supervisor in order to create a certain task for the intern.
* notify() – Method used from the company to notify information such as events to the intern and the supervisor.

**Internship Subsystem Classes**

**Internship**

***Attributes:***

intern : Intern: This class will use Intern objects in order to create an internship based on that particular Intern.

company : Company: This class will use Company objects in order to create an internship based on that particular Company.

supervisor : Supervisor: This class will use Supervisor objects in order to create an internship based on that particular Supervisor.

team : Team: This class will use Team objects in order to create an internship based on that particular Team.

startDate : Date: This class will use Date objects in order to create an internship based on that particular starting date of internship.

endDate : Date: This class will use Date objects in order to create an internship based on that particular ending date of internship.

taskList : ArrayList<Task>: This class will use Task objects in order to hold all of the tasks that are related with that particular internship.

status : boolean:This class will use a boolean status attribute which is going to indicate whether the internship is done or not.

***Methods:***

getTask(): This method will show the tasks to the intern so that the intern user will be able to see the tasks given by the supervisor.

getEvents():This method will show the general events to the intern and the supervisor so that the intern and the supervisor user will be able to see the general events given by the company.

updateScheduler(): This method will update the scheduler objects of each user, i.e. company, supervisor, student.

getDates(): This method will be used by the intern and supervisor users in order to see the tasks and the general events.

provideInfo(): This method will be used by the intern class in order to provide the internship information that is requested by the company.

joinCompany(): This method will be accessed by the intern in order to enroll to a company.

getCompanySpecificInfo(): This method will be used by the intern to see the detailed information of the company that it has its internship at.

updateEvents(): This method is going to be called by the company.

updateDates(): Company user will be able to update the dates of the general events and the supervisors will be able to update the dates of the tasks

requestInfo(): This method will be used by the company class in order to request the internship information that should be provided by the intern.

createInternKey(): Company user will create an intern key for the inter to use in order to enroll to the company.

addIntern(): This method will add an intern to the supervisor’s list and the list of interns in the department class.

removeIntern(): This method will remove an intern from the supervisor’s list and the list of interns in the department class.

getInterns(): This method will be called by the supervisor in order to get the interns of itself.

**User**

***Attributes:***

name : String: Name of the user.

email : String: E-mail of the user.

phoneNumber : String: Phone number of the user

userName : String: Username of the user

userType : String: Usertype of the user

***Methods:***

createProfile(): This method will able the user to create their profiles by means of writing their informations.

**Company**

***Attributes:***

companyDescription : String: Description of the company.

address : String: Address of the company

companyLogo : Image: The logo of the company

departments : ArratList<Department>: The departments of the company which is specified by the company user.

generalEventList : ArrayList<Task>: The general events that are created and manipulated by the company.

***Methods:***

getPublicInfo(): This method will be able to show the information of the company which is visible to the public in the system.

setPublicInfo(): This method will able the company to set the information of itself in the system.

acceptSupervisor(): This method will be used in the process of accepting the supervisor into the department of the company.

joinCompany(): This method will update the intern and supervisor lists of the departments when an intern or a supervisor enters into the system.

assignDepartment(): This method will assign a department for intern or supervisor classes.

createDepartment(): The company class will call this method in order to create the specified department.

**Department**

***Attributes:***

departmentName : String: The name of the department

supervisorList : ArrayList<Supervisor>: The list of all the supervisors belong to this department.

internList : ArrayList<Intern>: The list of all the inters belong to this department.

unmatchedSupervisor : ArrayList<Supervisor>: The list of unmatched supervisors in that department.

unmatchedIntern : <ArrayList<Intern>:The list of unmatched interns in that department.

***Methods:***

matchSupervisorToInt(): This method will match the intern with the supervisor.

showUnmatchedSupervisors(): This method returns the list of unmatched supervisors.

showUnmatchedInterns(): This method returns the list of unmatched interns.

**Intern**

***Attributes:***

surname : String: Surname of the intern

birthDate : Date: Surname of the intern

universityName : String: The university of the intern

departmentName : String: The department of the intern

taskList : ArrayList <Task>: The task of the intern which is given by the supervisor

matched : boolean: The status of the intern whether he is matched or not

***Methods:***

display(): This method will be able the other users to display the information of the intern.

completeTask(): This method will able the intern to mark the task as completed.

uncompleteTask(): This method will able the intern to mark back the task as uncompleted.

**Supervisor**

***Attributes:***

surname : String: Surname of the supervisor

birthDate : Date: Birthddate of the supervisor

graduation : String: The university of the supervisor that he/she had graduated

uniDepartmentName : String: The university department of the supervisor

compDepartment : String: The company department of the supervisor

internList : ArrayList <Intern>: The list of the interns that this particular supervisor has

taskList : ArrayList <Task>: The task of the intern which is given by the him/her

matched : boolean: The status of the supervisor whether he is matched or not

***Methods:***

display(): This method will be able the other users to display the information of the supervisor.

createTeam(): Supervisor will be able to create a team by interns in its list.

addMember(): Supervisor will be able to add new member to the team.

removeMember(): Supervisor will be able to remove a member from the team.

checkTask(): Task of the intern will be checked by the supervisor.

uncheckTask(): Task of the intern will be unchecked back by the supervisor.

**Team**

***Attributes:***

name : String: Name of the team

internList : ArrayList <Intern>: The interns that are belong to this particular team.

noOfInterns : int:

***Methods:***

display(): This method will be able the other users to display the information of the team.

**Scheduler**

***Attributes:***

calendar : Date[][]: 2D Array of date object to represent the calendar.

taskList : ArrayList <Task>: The tasks that will be shown in the calendar.

***Methods:***

display(): This method will be used by the other user to display the scheduler with its updated form.

**InternshipController**

***Attributes:***

account : Account: This instance will be used for the checking and displaying the user interface of the corresponding object.

***Methods:***

updateUI(): This method will notify the classes in the User Interface subsystem in order to display the corresponding user interface for the chosen account.

# Glossary & References

1. Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.