Packages 2

Class Interfaces 2

Global 2

SunActionPerformer 2

Server 3

SUN 3

CommunicationHandler 3

SerializationManager 4

JSONSerializationManager 4

Logic 4

UserManager 4

NotificationManager 4

NewsManager 4

EventManager 5

CheckInManager 5

Data 5

DatabaseManager 5

HibernateDatabaseManager 6

User 6

Student 6

ClubRepresentative 6

Joinable 7

Course 7

StudentClub 7

Hobby 7

FeedItem 7

News 8

Event 8

Notification 8

CheckIn 8

Mobile 9

SUNApp 9

LoginPage 9

SignupPage 9

ProfilePage 9

MainPage 10

JoinableDetailPage 10

CreateEventPage 10

SearchInterestsPage 11

FindFriendsPage 11

SUNClient 12

User 12

Student 12

ClubRepresentative 12

Joinable 12

Course 13

StudentClub 13

Hobby 13

FeedItem 13

News 13

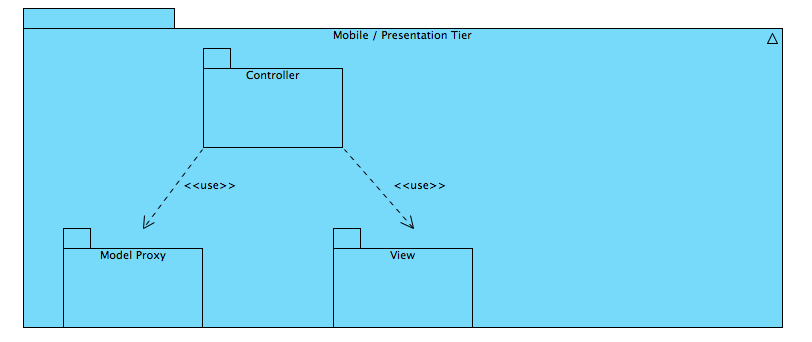
Event 13

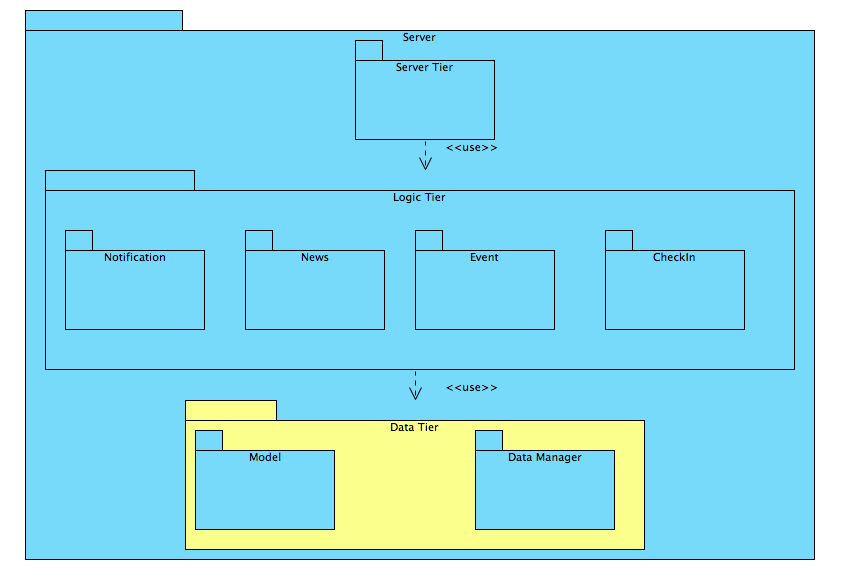
Notification 14

CheckIn 14

Packages

There are 2 main packages of the system. These are mobile and server packages, and they handle the android application and the backend of the system respectively.





# Class Interfaces

## Global

|  |  |
| --- | --- |
| Interface | SunActionPerformer |
| Description | This interface declares every possible action that a client can request. So API clients and CommunicationHandler’s must realize these methods to work properly. |
| Package | sun |
| Attributes | - |
| Operations | createEvent(name: String, date: Date, joinable: Joinable): Event  deleteEvent(event: Event): Boolean  getNewsFeed():FeedItem[]  checkIn(event: Event): CheckIn  joinEvent(event: Event): Boolean  leaveEvent(event: Event): Boolean  addInterest(interest: Joinable): Boolean  removeInterest(interest: Joinable): Boolean  setProfilePicture(avatar: String): Boolean  searchUser(keyword: String): User[]  addFriend(friend: Student): Boolean  getNotifications(): Notification[]  getJoinable(id: int): Joinable  getJoinableEvents(joinable: Joinable): Event[] |

## Server

|  |  |
| --- | --- |
| Class | SUN |
| Description | This class is the main manager of the backend system. It has the main method and instantiates the other managers and manages the communication between them. |
| Package | server.server |
| Attributes | communicationHandlers: CommunicationHandler[]  // Keeps the currently active communication handlers for clients  serializers: SerializationManager[]  userManager: UserManager |
| Operations | main(args: String[])  getHandlerForUser(user:User): ConnectionHandler  performAction(serializedAction:String):Object  // Requests from the client arrives as a serialized string. This method is the entry point of the request. It deserializes the request and delegates it.  createUserHandlerForLogin(user:User):ConnectionHandler |

|  |  |
| --- | --- |
| Class | CommunicationHandler |
| Description | Manages the relation between users and their assigned server sockets |
| Package | server.server |
| Attributes | loggedInUser : User  // User that this handler belongs to  clientSocket : Socket |
| Operations | - |

|  |  |
| --- | --- |
| Class | SerializationManager |
| Description | An abstract class that defines the needs of a SerializationManager instance. Each SerializationManager subclass need to be able to deserialize the given String and understand the request and serialize the result |
| Package | server.server |
| Attributes | - |
| Operations | SerializationManager(sun: SUN): SerialiazationManager  deserializeAndPerform(action: String): Object  // Method that understands the request and delegates it to other helpers.  serialize(object: Serializable) : String |

|  |  |
| --- | --- |
| Class | JSONSerializationManager |
| Description | A subclass of SerializationManager, which handles requests that serialized using JSON notation. |
| Package | server.server |
| Attributes | - |
| Operations | - |

### Logic

|  |  |
| --- | --- |
| Class | UserManager |
| Description | Handles client requests, which are related to user operations like login and signup. |
| Package | server.logic |
| Attributes | - |
| Operations | login(email: String, password: String) : User  signup(email: String, password: String, name: String) : Student  UserManager(sun: SUN) : UserManager  // Only constructor of this manager gets the reference to the SUN object to work. |

|  |  |
| --- | --- |
| Class | NotificationManager |
| Description | Handles the logic behind sending notifications to the students. |
| Package | server.logic.notification |
| Attributes | - |
| Operations | sendNotification(notification: Notification): Boolean  getNotificationsOfUser(user: Student): Notification[] |

|  |  |
| --- | --- |
| Class | NewsManager |
| Description | This manager handles the items that students see on their news feeds. |
| Package | server.logic.news |
| Attributes | - |
| Operations | addFeedItem(feedItem: FeedItem): Boolean  getNewsFeed(user: Student): FeedItem[] |

|  |  |
| --- | --- |
| Class | EventManager |
| Description | Handles the logic of creating an event and all actions that are performed on events |
| Package | server.logic.event |
| Attributes | - |
| Operations | createEvent(name: String, date: Date, joinable: Joinable, user: User): Boolean  deleteEvent(event: Event, user: User): Boolean  // User parameter indicates the user that sends the delete request. Only the creator of the event may delete it.  joinEvent(event: Event, user: User): Boolean  leaveEvent(event: Event, user: User): Boolean  getEventsOfUser(user: User, requester: User): Event[]  // Some events can only be seen by the participator. |

|  |  |
| --- | --- |
| Class | CheckInManager |
| Description | Handles the check-in logic of the application |
| Package | server.logic.checkin |
| Attributes | - |
| Operations | checkIn(user: User, event: Event): CheckIn |

### Data

|  |  |
| --- | --- |
| Class | DatabaseManager |
| Description | This abstract class defines the actions that a DatabaseManager instance must handle; data saving and retrieval functions of every action. |
| Package | server.data |
| Attributes | - |
| Operations | login(email: String, password: String): User  signup(email: String, password: String, name: String): User  saveEvent(event: Event): Boolean  deleteEvent(event: Event): Boolean  getNewsFeedForUser(user: User): FeedItem[]  saveCheckIn(checkin: CheckIn): Boolean  addUserToEvent(user: User, event: Event): Boolean  removeUserFromEvent(user: User, event: Event): Boolean  addUserInterest(user: User, interest: Joinable): Boolean  removeUserInterest(user: User, interest: Joinable): Boolean  saveUser(user: User): Boolean  // Saves the changes that have been made on the user instance  findUser(keyword: String): User[]  addFriend(adder: User, target: User): Boolean  removeFriend(remover: User, target: User): Boolean  getNotificationsOfUser(user: User): Notification[]  addNotificationToUser(notification: Notification, user: User): Boolean  getJoinable(id: int): Joinable |

|  |  |
| --- | --- |
| Class | HibernateDatabaseManager |
| Description | A subclass of DatabaseManager which uses Hibernate to handle database operations. |
| Package | server.data |
| Attributes | - |
| Operations | - |

|  |  |
| --- | --- |
| Class | User |
| Description | An abstract class that represents minimum requirements of a User object |
| Package | server.data.model |
| Attributes | id: int  email: String  password: String  name: String  avatar: String  contactInfo: String  lastLogin: Date |
| Operations | - |

|  |  |
| --- | --- |
| Class | Student |
| Description | A subclass of User that keeps student informations |
| Package | server.data.model |
| Attributes | friends: Student[]  events: Event[]  interests: Joinable[]  points: int  notifications: Notification[]  department: String  // Departments of the Students will help the system to offer events and friends to them |
| Operations | - |

|  |  |
| --- | --- |
| Class | ClubRepresentative |
| Description | A subclass of User which keeps information about club representatives. |
| Package | server.data.model |
| Attributes | represents: StudentClub  numberOfPosts: int  validUntil: Date  // Representative accounts may expire at the end of the semester. |
| Operations | - |

|  |  |
| --- | --- |
| Class | Joinable |
| Description | An abstract class, which keeps basic information about concepts that, a Students can join or track. |
| Package | server.data.model |
| Attributes | id: int  name: String  info: String  followerCount: int  latestEventAdded: Date  // In order to sort the Joinable |
| Operations | - |

|  |  |
| --- | --- |
| Class | Course |
| Description | A subclass of Joinable, which represents a course (e.g. CS101). |
| Package | server.data.model |
| Attributes | code: String |
| Operations | - |

|  |  |
| --- | --- |
| Class | StudentClub |
| Description | A subclass of Joinable, which represents a student club (e.g. IEEE). |
| Package | server.data.model |
| Attributes | - |
| Operations | - |

|  |  |
| --- | --- |
| Class | Hobby |
| Description | A subclass of Joinable, which represents a hobby that may be done on campus (e.g. Biking). |
| Package | server.data.model |
| Attributes | - |
| Operations | - |

|  |  |
| --- | --- |
| Class | FeedItem |
| Description | An abstract class that represents everything that can be displayed on users’ news feeds. |
| Package | server.data.model |
| Attributes | id: int  date: Date  creator: User  showCount: int  // How many students has seen this item |
| Operations | - |

|  |  |
| --- | --- |
| Class | News |
| Description | A subclass of FeedItem that keeps news about events and activities. |
| Package | server.data.model |
| Attributes | text: String |
| Operations | - |

|  |  |
| --- | --- |
| Class | Event |
| Description | This class holds information about events that students can join. |
| Package | server.data.model |
| Attributes | id: int  name: String  joinable: Joinable  // subject of the event, such as CS101 or Biking  joinedStudents: Student[]  eventInfo: String  owner: Student  displayCount: int |
| Operations | - |

|  |  |
| --- | --- |
| Class | Notification |
| Description | Keeps the information of notifications that the students will see. |
| Package | server.data.model |
| Attributes | id: int  source: User  target: Student  type: NotificationType  date: Date  data: String  reached: Boolean  opened: Boolean |
| Operations | - |

|  |  |
| --- | --- |
| Class | CheckIn |
| Description | Model object that keeps the information of a checking including rewarded points. |
| Package | server.data.model |
| Attributes | id: int  student: Student  event: Event  date: Date  rewardedPoints: int |
| Operations | - |

## Mobile

|  |  |
| --- | --- |
| Class | SUNApp |
| Description | Main class of the Android application. |
| Package | mobile.controller |
| Attributes | - |
| Operations | handleNotification(notification: Notification)  // When a push notification arrive at the mobile application this method will be invoked to display it to the user |

|  |  |
| --- | --- |
| Class | LoginPage |
| Description | Displays the username and password inputs to get credentials of the user and tries to login with the given information using the SUNClient |
| Package | mobile.controller |
| Attributes | username: EditText  password: EditText  loginButton: Button  signupButton: Button |
| Operations | loginPressed()  toSignup() |

|  |  |
| --- | --- |
| Class | SignupPage |
| Description | Handles signup process by asking the name and the credentials of the user. |
| Package | mobile.controller |
| Attributes | name: EditText  username: EditText  password: EditText  signupButton: Button  backButton: Button |
| Operations | back()  signup() |

|  |  |
| --- | --- |
| Class | ProfilePage |
| Description | Displays personal information of any user. |
| Package | mobile.controller |
| Attributes | events: Event[]  user: User  list: ListView  // Changes the displayed events depending on the users’ choices. |
| Operations | showMyEvents()  // Shows the events that the user marked as he will join  showNextEvents()  // Shows the upcoming events of the Joinables that the user follow  showPastEvents()  // Displays the events that the user participated in.  showContactInfo()  changeAvatar()  // Show picture selection dialog  setAvatar(file: Image) |

|  |  |
| --- | --- |
| Class | MainPage |
| Description | This is the dashboard page of the application, which will be opened when the application launches. |
| Package | mobile.controller |
| Attributes | showProfileButton: Button  showEventsButton: Button  findFriendsButton: Button  bumpButton: Button |
| Operations | showProfile()  showEvents()  findFriends()  bump()  // Opens the dialog that allows users’ to bump their phones to check in with a friend. |

|  |  |
| --- | --- |
| Class | JoinableDetailPage |
| Description | Displays details of a Joinable instance. |
| Package | mobile.controller |
| Attributes | name: TextView  info: TextView  eventList: ListView  events: Event[]  createEventButton: Button  backButton: Button |
| Operations | showCreateEventPage()  showInfo()  back() |

|  |  |
| --- | --- |
| Class | CreateEventPage |
| Description | This page collects necessary information to create a new event about a Joinable |
| Package | mobile.controller |
| Attributes | joinable: Joinable  name: EditText  info: EditText  saveButton: Button  backButton: Button |
| Operations | showConfirmation()  createEvent()  // Sends the new event information to server and tries to create the event  back() |

|  |  |
| --- | --- |
| Class | SearchInterestsPage |
| Description | This is the page where users follow or un-follow Joinables. |
| Package | mobile.controller |
| Attributes | filterBox: EditText  // The textbox that users enter keywords to filter the results  typeSelection: SegmentedButton  // Allows users to select the kind of Joinable that they want to follow, selections are; Course, Club and Hobby  list: ListView  backButton: Button |
| Operations | showCourseList()  showClubList()  showHobbyList()  addInterest(interest: Joinable)  // This method will be called when the user select a Joinable to follow  removeInterest(interest: Joinable)  back() |

|  |  |
| --- | --- |
| Class | FindFriendsPage |
| Description | This page allows users to find and add new friends to their friend lists. |
| Package | mobile.controller |
| Attributes | filterBox: EditText  // An input box that will be used to enter a keyword to find a friend. The keyword may be the part of the name or the username of the friend.  list: ListView  searchButton: Button  back: Button |
| Operations | searchFriend(keyword: String)  // Sends a request to find relevant Students for the keyword and updates the list according to the results  sendFriendRequest(target: Student)  back() |

|  |  |
| --- | --- |
| Class | SUNClient |
| Description | A singleton class that handles the communication between the mobile application and the server. This class implements the SUNActionPerformer interface and can handle every possible request that the application components need to make. |
| Package | mobile.controller |
| Attributes | instance: SUNClient |
| Operations | getInstance(): SUNClient |

|  |  |
| --- | --- |
| Class | User |
| Description | An abstract class that represents minimum requirements of a User object |
| Package | mobile.modelproxy |
| Attributes | id: int  email: String  name: String  avatar: String  contactInfo: String |
| Operations | - |

|  |  |
| --- | --- |
| Class | Student |
| Description | A subclass of User that keeps student informations |
| Package | mobile.modelproxy |
| Attributes | friends: Student[]  events: Event[]  interests: Joinable[]  points: int  notifications: Notification[] |
| Operations | - |

|  |  |
| --- | --- |
| Class | ClubRepresentative |
| Description | A subclass of User which keeps information about club representatives. |
| Package | mobile.modelproxy |
| Attributes | represents: StudentClub |
| Operations | - |

|  |  |
| --- | --- |
| Class | Joinable |
| Description | An abstract class, which keeps basic information about concepts that, a Students can join or track. |
| Package | mobile.modelproxy |
| Attributes | id: int  name: String  info: String |
| Operations | - |

|  |  |
| --- | --- |
| Class | Course |
| Description | A subclass of Joinable, which represents a course (e.g. CS101). |
| Package | mobile.modelproxy |
| Attributes | code: String |
| Operations | - |

|  |  |
| --- | --- |
| Class | StudentClub |
| Description | A subclass of Joinable, which represents a student club (e.g. IEEE). |
| Package | mobile.modelproxy |
| Attributes | - |
| Operations | - |

|  |  |
| --- | --- |
| Class | Hobby |
| Description | A subclass of Joinable, which represents a hobby that may be done on campus (e.g. Biking). |
| Package | mobile.modelproxy |
| Attributes | - |
| Operations | - |

|  |  |
| --- | --- |
| Class | FeedItem |
| Description | An abstract class that represents everything that can be displayed on users’ news feeds. |
| Package | mobile.modelproxy |
| Attributes | id: int  date: Date  creator: User |
| Operations | - |

|  |  |
| --- | --- |
| Class | News |
| Description | A subclass of FeedItem that keeps news about events and activities. |
| Package | mobile.modelproxy |
| Attributes | text: String |
| Operations | - |

|  |  |
| --- | --- |
| Class | Event |
| Description | This class holds information about events that students can join. |
| Package | mobile.modelproxy |
| Attributes | id: int  name: String  joinable: Joinable  // subject of the event, such as CS101 or Biking  joinedStudents: Student[]  eventInfo: String |
| Operations | - |

|  |  |
| --- | --- |
| Class | Notification |
| Description | Keeps the information of notifications that the students will see. |
| Package | mobile.modelproxy |
| Attributes | id: int  source: User  target: Student  type: NotificationType  date: Date  data: String |
| Operations | - |

|  |  |
| --- | --- |
| Class | CheckIn |
| Description | Model object that keeps the information of a checking including rewarded points. |
| Package | mobile.modelproxy |
| Attributes | id: int  student: Student  event: Event  date: Date  rewardedPoints: int |
| Operations | - |