IController

run(): int

LoginSystem

- userMan: UserManager
- lp: LoginSystemPresenter
- input: Scanner

+ run(): int
+ createAccount(): int
+ loginToAccount(): int

LoginSystemPresenter

- userMan: UserManager
- lp: LoginSystemPresenter
- input: Scanner = new Scanner(System.in);

+ LoginCreateActPrompt(): void
+ enterUsernamePrompt(): void
+ goBackOptionPrompt(): void
+ displayLoginSuccess(accountType: String, userName: String): void
+ enterActTypePrompt(): void
+ invalidActTypePrompt(accountType: String): void
+ enterActNamePrompt(): void
+ invalidActNamePrompt(): void
+ displaySuccCreation(accountType: String, accountName: String): void
+ invalidInputPrompt(): void

AttendeePanel

- userMan: UserManager
- msgMan: MessageManager
- roomMan: RoomManager
- inputFilter: InputFilter
- input: Scanner = new Scanner(System.in);

- displayCommands(): void
- Message(currUserID: UUID): Integer
- viewMessages(allUserIds: ArrayList<UUID>, currUserID: UUID): Integer
- viewAllEvents(): void
- viewSignedUpEvents(currUserID: UUID): void
- joinLeaveEvent(currUserID: UUID, userMan: UserManager, joinOrLeave: String):Integer
- userExists(username: String): boolean

+ run(): int

SpeakerPanel

- userMan: UserManager
- roomMan: RoomManager
- msgMan: MessageManager
- speakerPres: SpeakerPresenter
- input: Scanner = new Scanner(System.in);

- Message(currUserID: UUID, allUserIds: ArrayList<UUID>): Integer
- userExists(username: String): boolean
- viewMessages(allusers: ArrayList<UUID>, currUser: UUID): void
- viewAllEvents(): void
- viewSpeakingEvents(speakerID: UUID): void
- sendMessageToAttendees(speakerID: UUID): int

+ run(): int

Definitions

+ QUIT = int = -3
+ BACK = int = -2
+ REMAIN_IN_STATE = int = -1

Interface
Inheritance
Entity
Use Case
Controller

Created by @Liam Ogilvie

OrganizerPanel

- sc: Scanner = new Scanner(System.in);
- um: UserManager
- rm: RoomManager
- op: OrganizerPresenter
- filter: InputFilter
- START_HOUR_EARLIEST: int = 9
- START_HOUR_LATEST: int = 16

- cancelRequested(input: String): boolean
- createEvent(): void
- createRoom(): void
- cancelEvent(): void
- rescheduleEvent(): void
- createSpeaker(): void

+ run(): int

SpeakerPresenter

- userMan: UserManager
- roomMan: RoomManager
- msgMan: MessageManager

+ viewAllEvents(): void
+ viewSpeakingEvents(speakerID: UUID): void
+ sendAllAttendeesIntro(speakerID: UUID): void
+ viewMessages(allusers: ArrayList<UUID>, currUser: UUID): void
+ messageSelectedTalks(talks: ArrayList): void
+ messagePrompt(): void
+ messageSuccess(msgid: UUID, event: UUID): void
+ errorNoSpeakingEvents(): void
+ errorInvalidInput(command: String): void
+ errorNoAttendees(): void
+ errorGeneral(): void
+ errorUserNotFound(): void
+ dmPrompt(): void
+ typeMsgPrompt(): void
+ msgSentPrompt(username: String): void
+ listAllUsers(): void

InputFilter

- um: UserManager
- rm: RoomManager
- ip: InputPresenter
- sc: Scanner = new Scanner(System.in);
- START_HOUR_EARLIEST: int = 9
- START_HOUR_LATEST: int = 16

+ inputRoom(): int
+ inputEventNumber(roomNumber: int): int
+ inputSpeakerUsername(): String
+ inputNewSpeakerUsername(): String
+ inputYear(): int
+ inputMonth(year: int): int
+ inputDay(year: int, month: int)
+ inputHour(year: int, month: int, day: int)
+ inputMinute(): int

InputPresenter

+ inputRoomPrompt(): void
+ inputRoomStatus(): void
+ inputEventPrompt(): void
+ inputEventStatus(): void
+ inputSpeakerNamePrompt(): void
+ inputSpeakerNameStatusNonSpeaker(speakerName: String): void
+ inputSpeakerNameStatusDNE(speakerName: String): void
+ inputNewSpeakerNamePrompt(): void
+ inputNewSpeakerNameStatus(speakerName: String): void
+ inputYearPrompt(): void
+ inputYearStatus(): void
+ inputMonthPrompt(): void
+ inputMonthStatus(month: int): void
+ inputDayPrompt(): void
+ inputDayStatus(day: int): void
+ inputHourPrompt(startHourEarliest: int, startHourLatest: int): void
+ inputHourStatus(hour: int): void
+ inputMinutePrompt(): void
+ inputMinuteStatus(minute: int): void
+ invalidInputNotification(): void

OrganizerPresenter

- userMan: UserManager
- msgMan: MessageManager

+ welcomePrompt(): void
+ commandPrompt(): void
+ commandNotRecognized(command: String): void
+ rescheduleEventWelcome(): void
+ rescheduleEventStatus(status: boolean): void
+ cancelEventWelcome(): void
+ cancelEventStatus(): void
+ createSpeakerWelcome(): void
+ createSpeakerStatus(speakerName: String): void
+ createRoomStatus(numRooms: int): void
+ createEventWelcome(): void
+ createEventTitlePrompt(): void
+ createEventStatus(status: boolean, event: String): void
+ printSpeakerEvents(speakerName: String): void
+ printRoomEvents(roomNumber: int): void
+ printAllEvents(): void
+ printAvailableSpeakers(): void
+ cancelNotification(): void
+ printAvailableRooms(): void

AttendeePresenter

- userMan: UserManager
- msgMan: MessageManager

+ displayWelcomeMsg(): void
+ displayCommands(): void
+ dmPrompt(): void
+ typeMsgPrompt(): void
+ msgSentPrompt(): void
+ whosMsgPrompt(): void
+ displayMsgs(allUserIds: ArrayList<UUID>, currUserID: UUID): void
+ displaySpecificUserMsg(content: List<String>): void
+ displayAllEvents(eventInfo: String): void
+ displayAttendingEvents(eventInfo: String): void
+ joinLeaveEventOrRoomPrompt(joinOrLeave: String, eventOrRoom: String): void
+ displayJoinLeaveSuccess(joinOrLeave: String): void
+ displayJoinLeaveError(joinOrLeave: String): void
+ displayUserDoesNotExistError(): void
+ displayInvalidCommandError(): void

User Manager

- attendees: ArrayList<Attendee>
- organizers: ArrayList<Organizer>
- speakers: ArrayList<Speaker>
- currentUser: User

- getUsers(): List<UUID>
- getUserIDToUser(): HashMap<UUID, User>
- getUsernameToUser(): HashMap<String, User>
- getUser(userID: UUID): User
- getUser(username: String): User
- getSpeakers(): ArrayList<Speaker>
- getOrganizers(): ArrayList<Organizer>

+ getUsernames(): ArrayList<String>
+ getAttendeeUUIDs(): ArrayList<UUID>
+ userExists(userID: UUID): boolean
+ usersExists(userIDs: List<UUID>): boolean
+ getCurrentUser(): UUID
+ setCurrentUser(username: String): boolean
+ setCurrentUserFromUserName(username: String): void
+ createAttendeeAccount(username: String): UUID
+ getOrganizerUUIDs(): ArrayList<UUID>
+ createOrganizerAccount(username: String): UUID
+ createSpeakerAccount(username: String): UUID
+ isValidUsername(username: String): boolean
+ userType(username: String): String
+ getSpeakerEventIDs(speakerName: String): ArrayList<UUID>
+ getSpeakerEventIDs(speakerID: UUID): ArrayList<UUID>
+ speakerAddEvent(speakerName: String, roomID: UUID, eventID: UUID): void
+ speakerRemoveEvent(speakerName: String, roomID: UUID, eventID: UUID): void
+ attendeeAddEvent(attendeeID: UUID, roomID: UUID, eventID: UUID): void
+ attendeeRemoveEvent(attendeeID: UUID, roomID: UUID, eventID: UUID): void
+ userExists(username: String): boolean
+ isSpeaker(username: String): boolean
+ stringAvailableSpeakers(): String
+ getMessagesFromUser(recipientID: UUID, senderID: UUID): List<UUID>
+ addMessage(recipientID: UUID, senderID: UUID): void
+ getUserID(username: String): UUID
+ getUsername(UserID: UUID): String

MessageManager

- messages: Map<UUID, Message>

- sendMessage(userManager: UserManager, senderID: UUID, recipientIDs: List<UUID>, messageContent: String): UUID
- getMessagesFromUser(userManager: UserManager, recipientID: UUID, senderID: UUID): ArrayList<Message>

+ sendMessage(userManager: UserManager, senderID: UUID, recipientID: UUID, messageContent: String): UUID
+ sendMessageToAllAttendees(userManager: UserManager, senderID: UUID, messageContent: String): UUID
+ getMessageContentsFromUser(userManager: UserManager, recipientID: UUID, senderID: UUID): List<String>
+ sendMessageToEventAttendees(userManager: UserManager, roomManager: RoomManager, senderID: UUID, eventID: UUID, messageContent: String): UUID

RoomManager

- rooms: HashMap<UUID, Room>

- getRoom(roomID: UUID): Room
- getRoom(roomNumber: int): Room
- getEventRoom(event: Event): Room
- getEvent(eventID: UUID): Event
- getEvent(roomNumber: int, eventNumber: int): Event
- getRooms(): ArrayList<Room>
- getEvents(): ArrayList<Event>
- getEventsFromRoom(room: Room): ArrayList<Event>
- getEventsFromRoom(roomNumber: int): ArrayList<Event>
- eventIDsToEvents(eventIDs: ArrayList<UUID>): ArrayList<Event>
- getRoomIDToRoom(): HashMap<UUID, Room>
- getEventIDToEvent(): HashMap<UUID, Event>

+ getEventAttendeeIDs(eventID: UUID): ArrayList<UUID>
+ getNumRooms(): int
+ getNumEvents(): int
+ getNumEventsInRoom(roomNumber: int): int
+ getEventIDs(): ArrayList<UUID>
+ newRoom(): void
+ newEventValid(eventTitle: String, speakerName: String, startTime: Calendar, endTime: Calendar, roomNumber: int, um: UserManager): boolean
+ newEventValid(eventTitle: String, speakerName: String, startTime: Calendar, endTime: Calendar, room: Room, um: UserManager): boolean
+ newEvent(eventTitle: String, speakerName: String, startTime: Calendar, endTime: Calendar, roomNumber: int): UUID
+ rescheduleEvent(um: UserManager, roomNumber: int, eventNumber: int, startTime: Calendar, endTime: Calendar): boolean
+ removeEvent(um: UserManager, roomNumber: int, eventNumber: int): boolean
+ addEventAttendee(attendeeID: UUID, eventID: UUID, um: UserManager): boolean
+ removeEventAttendee(attendee: Attendee, event: Event): boolean
+ stringEventsOfSpeaker(um: UserManager, speakerName: String): String
+ stringEventsOfRoom(roomNumber: int): String
+ stringEventInfoAll(): String
+ stringEventInfoAttending(attendeeID: UUID): String
+ stringEvent(eventID: UUID): String

Message

- messageID: UUID
- messageContent: String

+ getMessageContent(): String
+ getMessageID(): UUID

Room

- roomID: UUID
- events: HashMap<UUID, Event>

- eventOutOfBounds(newEvent: Event): boolean

+ getRoomID(): UUID
+ getEvents(): ArrayList<Event>
+ getEventIDToEvent(): HashMap<UUID, Event>
+ getTimeSchedule(): HashMap<Calendar, Event>
+ getSpeakerIDSchedule(): HashMap<String, ArrayList<Event>>
+ getTitleSchedule(): HashMap<String, ArrayList<Event>>
+ getEventsByTime(startTime: GregorianCalendar, endTime: GregorianCalendar): ArrayList<Event>
+ getEventsByTitle(title: String): ArrayList<Event>
+ getEventsBySpeakerID(speaker: Speaker): ArrayList<Event>
+ getEventIDs(): ArrayList<UUID>
+ getEvent(eventID: UUID): Event
+ eventOverlapping(newEvent: Event, comparisonEvent: Event): boolean
+ eventIsValid(newEvent: Event):boolean
+ addEvent(eventToAdd: Event): boolean
+ removeEvent(eventToRemove: Event): boolean

User

- userID: UUID
- username: String
- conversations: ArrayList<UUID>

+ getUserID(): UUID
+ getUsername(): String
+ getMessages(): List<UUID>
+ addMessage(sender: User message: Message): void
+ *isOrganizer()*: boolean
+ *isAttendee()*: boolean
+ *isSpeaker()*: boolean
+ *getStringType*: String

Event

- title: String
- eventID: UUID
- startTime: Calendar
- endTime: Calendar
- speakerName: String
- attendeeIDs: ArrayList<UUID>

@Override
+ toString(): String

+ getTitle(): String
+ getEventID(): UUID
+ getStartTime(): Calendar
+ getEndTime(): Calendar
+ getAttendeeIDs(): ArrayList<UUID>
+ addAttendee(attendeeID: UUID): boolean
+ removeAttendee(attendeeToRemove : Attendee): boolean
+ getSpeakerName(): String

Attendee

- events: ArrayList<UUID>

@Override
+ isOrganizer(): boolean (False)
+ isAttendee(): boolean (True)
+ isSpeaker(): boolean (False)
+ getStringType(): String (“attendee”)

+ getEvents(): Map<UUID, ArrayList<UUID>>
+ addEvents(roomID: UUID, eventID: UUID): boolean
+ removeReservedEvents(roomID: UUID, eventID: Event): boolean

Speaker

- eventsSpeaking: ArrayList<UUID>

@Override
+ isOrganizer(): boolean (False)
+ isAttendee(): boolean (False)
+ isSpeaker(): boolean (True)
+ getStringType(): String (“speaker”)

+ getEventsSpeaking(): HashMap<UUID, ArrayList<UUID>>
+ addEvent(roomID: UUID, eventID: UUID): void
+ removeEvent(roomID: UUID, eventID: UUID): boolean

Organizer

@Override
+ isOrganizer(): boolean (True)
+ isAttendee(): boolean (False)
+ getStringType(): String (“organizer”)