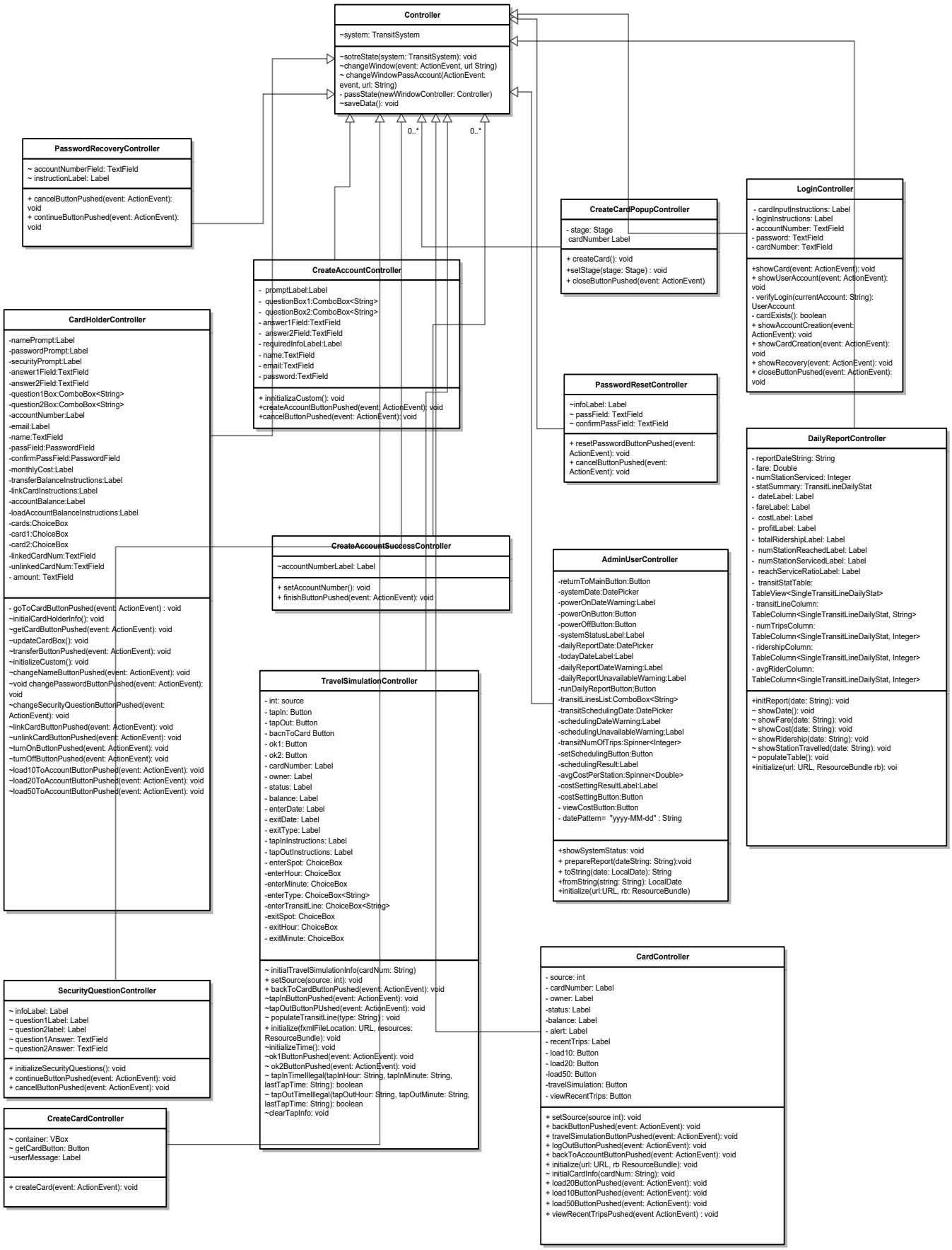
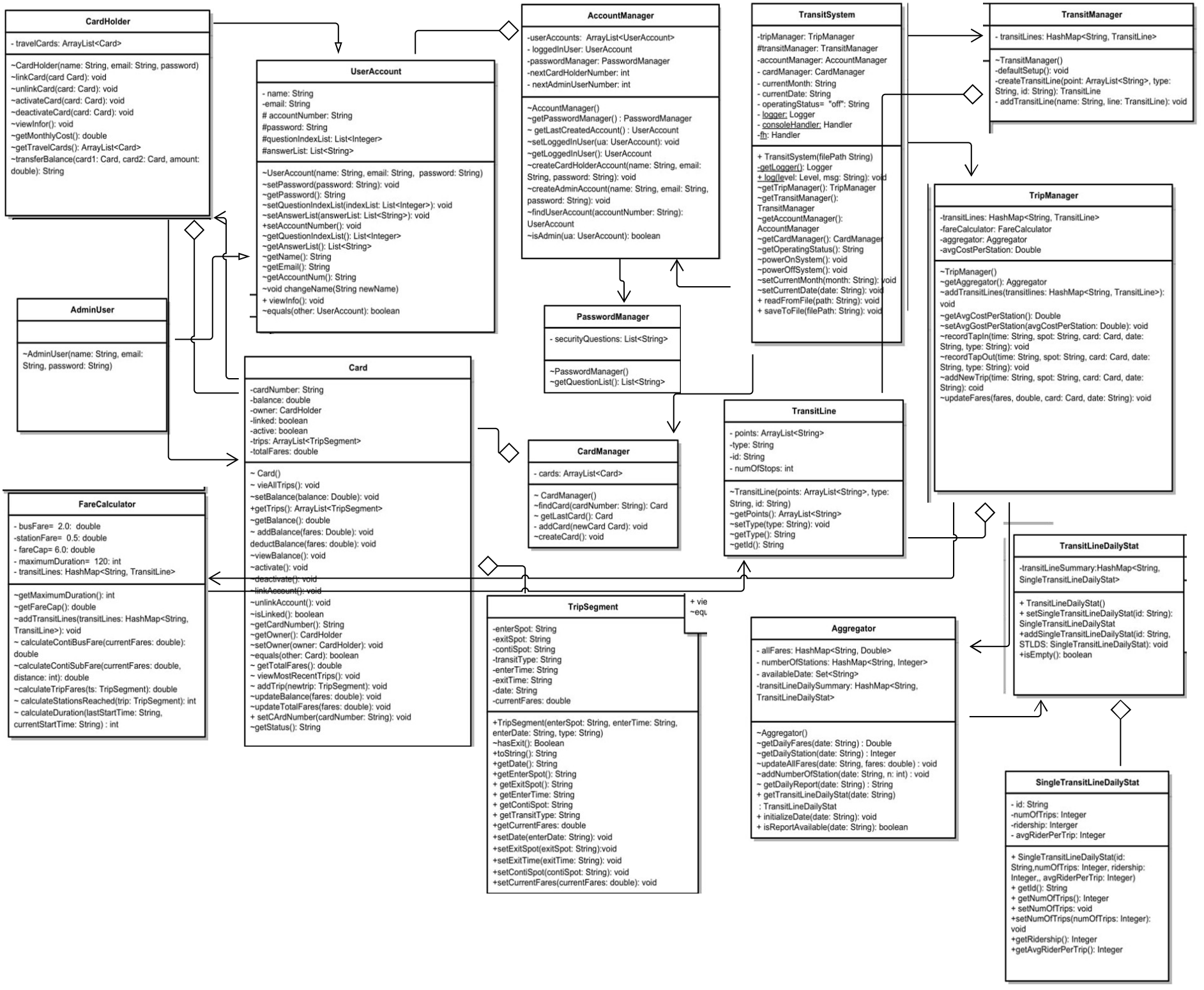

Controller
~system: TransitSystem
~sotreState(system: TransitSystem): void
~changeWindow(event: ActionEvent, url String)
~ changeWindowPassAccount(ActionEvent: event, url: String)
- passState(newWindowController: Controller)
~saveData(): void
0..*
0..*
PasswordRecoveryController
~ accountNumberField: TextField
~ instructionLabel: Label
+ cancelButtonPushed(event: ActionEvent): void
+ continueButtonPushed(event: ActionEvent): void
CreateCardPopupController
- stage: Stage
cardNumber Label
+ createCard(): void
+setStage(stage: Stage) : void
+ closeButtonPushed(event: ActionEvent)
LoginController
- cardInputInstructions: Label
- loginInstructions: Label
- accountNumber: TextField
- password: TextField
- cardNumber: TextField
+showCard(event: ActionEvent): void
+ showUserAccount(event: ActionEvent): void
- verifyLogin(currentAccount: String): UserAccount
- cardExists(): boolean
+ showAccountCreation(event: ActionEvent): void
+ showCardCreation(event: ActionEvent): void
+ showRecovery(event: ActionEvent): void
+ closeButtonPushed(event: ActionEvent): void
CreateAccountController
- promptLabel:Label
- questionBox1:ComboBox<String>
- questionBox2:ComboBox<String>
- answer1Field:TextField
- answer2Field:TextField
- requiredInfoLabel:Label
- name:TextField
- email:TextField
- password:TextField
+ initializaCustom(): void
+createAccountButtonPushed(event: ActionEvent): void
+cancelButtonPushed(event: ActionEvent): void
CardHolderController
-namePrompt:Label
-passwordPrompt:Label
-securityPrompt:Label
-answer1Field:TextField
-answer2Field:TextField
-question1Box:ComboBox<String>
-question2Box:ComboBox<String>
-accountNumber:Label
-email:Label
-name:TextField
-passField:PasswordField
-confirmPassField:PasswordField
-monthlyCost:Label
-transferBalanceInstructions:Label
-linkCardInstructions:Label
-accountBalance:Label
-loadAccountBalanceInstructions:Label
-cards:ChoiceBox
-card1:ChoiceBox
-card2:ChoiceBox
-linkedCardNum:TextField
-unlinkedCardNum:TextField
- amount: TextField
- goToCardButtonPushed(event: ActionEvent) : void
~initialCardHolderInfo(): void
~getCardButtonPushed(event: ActionEvent): void
~updateCardBox(): void
~transferButtonPushed(event: ActionEvent): void
~initializeCustom(): void
~changeNameButtonPushed(event: ActionEvent): void
~void changePasswordButtonPushed(event: ActionEvent): void
~changeSecurityQuestionButtonPushed(event: ActionEvent): void
~linkCardButtonPushed(event: ActionEvent): void
~unlinkCardButtonPushed(event: ActionEvent): void
~turnOnButtonPushed(event: ActionEvent): void
~turnOffButtonPushed(event: ActionEvent): void
~load10ToAccountButtonPushed(event: ActionEvent): void
~load20ToAccountButtonPushed(event: ActionEvent): void
~load50ToAccountButtonPushed(event: ActionEvent): void
PasswordResetController
~infoLabel: Label
~ passField: TextField
~ confirmPassField: TextField
+ resetPasswordButtonPushed(event: ActionEvent): void
+ cancelButtonPushed(event: ActionEvent): void
DailyReportController
- reportDateString: String
- fare: Double
- numStationServiced: Integer
- statSummary: TransitLineDailyStat
- dateLabel: Label
- fareLabel: Label
- costLabel: Label
- profitLabel: Label
- totalRidershipLabel: Label
- numStationReachedLabel: Label
- numStationServicedLabel: Label
- reachServiceRatioLabel: Label
- transitStatTable: TableView<SingleTransitLineDailyStat>
- transitLineColumn: TableColumn<SingleTransitLineDailyStat, String>
- numTripsColumn: TableColumn<SingleTransitLineDailyStat, Integer>
- ridershipColumn: TableColumn<SingleTransitLineDailyStat, Integer>
- avgRiderColumn: TableColumn<SingleTransitLineDailyStat, Integer>
+initReport(date: String): void
~ showDate(): void
~ showFare(date: String): void
~ showCost(date: String): void
~ showRidership(date: String): void
~ showStationTravelled(date: String): void
~ populateTable(): void
+initialize(url: URL, ResourceBundle rb): voi
CreateAccountSuccessController
~accountNumberLabel: Label
+ setAccountNumber(): void
+ finishButtonPushed(event: ActionEvent): void
AdminUserController
-returnToMainButton:Button
-systemDate:DatePicker
-powerOnDateWarning:Label
-powerOnButton:Button
-powerOffButton:Button
-systemStatusLabel:Label
-dailyReportDate:DatePicker
-todayDateLabel:Label
-dailyReportDateWarning:Label
-dailyReportUnavailableWarning:Label
-runDailyReportButton;Button
-transitLinesList:ComboBox<String>
-transitSchedulingDate:DatePicker
-schedulingDateWarning:Label
-schedulingUnavailableWarning;Label
-transitNumOfTrips:Spinner<Integer>
-setSchedulingButton:Button
-schedulingResult:Label
-avgCostPerStation:Spinner<Double>
-costSettingResultLabel:Label
-costSettingButton:Button
- viewCostButton:Button
- datePattern= "yyyy-MM-dd" : String
+showSystemStatus: void
+ prepareReport(dateString: String):void
+ toString(date: LocalDate): String
+fromString(string: String): LocalDate
+initialize(url:URL, rb: ResourceBundle)
TravelSimulationController
- int: source
- tapIn: Button
- tapOut: Button
- bacnToCard Button
- ok1: Button
- ok2: Button
- cardNumber: Label
- owner: Label
- status: Label
- balance: Label
- enterDate: Label
- exitDate: Label
- exitType: Label
- tapInInstructions: Label
- tapOutInstructions: Label
- enterSpot: ChoiceBox
-enterHour: ChoiceBox
-enterMinute: ChoiceBox
-enterType: ChoiceBox<String>
-enterTransitLine: ChoiceBox<String>
-exitSpot: ChoiceBox
- exitHour: ChoiceBox
- exitMinute: ChoiceBox
~ initialTravelSimulationInfo(cardNum: String)
+ setSource(source: int): void
+ backToCardButtonPushed(event: ActionEvent): void
~tapInButtonPushed(event: ActionEvent): void
~tapOutButtonPUshed(event: ActionEvent): void
~ populateTransitLine(type: String) : void
+ initialize(fxmlFileLocation: URL, resources: ResourceBundle): void
~initializeTime(): void
~ok1ButtonPushed(event: ActionEvent): void
~ ok2ButtonPushed(event: ActionEvent): void
~ tapInTimeIllegal(tapInHour: String, tapInMinute: String, lastTapTime: String): boolean
~ tapOutTimeIllegal(tapOutHour: String, tapOutMinute: String, lastTapTime: String): boolean
~clearTapInfo: void
SecurityQuestionController
~ infoLabel: Label
~ question1Label: Label
~ question2label: Label
~ question1Answer: TextField
~ question2Answer: TextField
+ initializeSecurityQuestions(): void
+ continueButtonPushed(event: ActionEvent): void
+ cancelButtonPushed(event: ActionEvent): void
CardController
- source: int
- cardNumber: Label
- owner: Label
-status: Label
-balance: Label
- alert: Label
- recentTrips: Label
- load10: Button
- load20: Button
-load50: Button
-travelSimulation: Button
- viewRecentTrips: Button
+ setSource(source int): void
+ backButtonPushed(event: ActionEvent): void
+ travelSimulationButtonPushed(event: ActionEvent): void
+ logOutButtonPushed(event: ActionEvent): void
+ backToAccountButtonPushed(event: ActionEvent): void
+ initialize(url: URL, rb ResourceBundle): void
~ initialCardInfo(cardNum: String): void
+ load20ButtonPushed(event: ActionEvent): void
+ load10ButtonPushed(event: ActionEvent): void
+ load50ButtonPushed(event: ActionEvent): void
+ viewRecentTripsPushed(event ActionEvent) : void
CreateCardController
~ container: VBox
~ getCardButton: Button
~userMessage: Label
+ createCard(event: ActionEvent): void


CardHolder
- travelCards: ArrayList<Card>
~CardHolder(name: String, email: String, password)
~linkCard(card Card): void
~unlinkCard(card: Card): void
~activateCard(card: Card): void
~deactivateCard(card: Card): void
~viewInfor(): void
~getMonthlyCost(): double
~getTravelCards(): ArrayList<Card>
~transferBalance(card1: Card, card2: Card, amount: double): String
UserAccount
- name: String
-email: String
accountNumber: String
#password: String
#questionIndexList: List<Integer>
#answerList: List<String>
~UserAccount(name: String, email: String, password: String)
~setPassword(password: String): void
~getPassword(): String
~setQuestionIndexList(indexList: List<Integer>): void
~setAnswerList(answerList: List<String>): void
+setAccountNumber(): void
~getQuestionIndexList(): List<Integer>
~getAnswerList(): List<String>
~getName(): String
~getEmail(): String
~getAccountNum(): String
~void changeName(String newName)
+ viewInfo(): void
~equals(other: UserAccount): boolean
AccountManager
-userAccounts: ArrayList<UserAccount>
- loggedInUser: UserAccount
-passwordManager: PasswordManager
-nextCardHolderNumber: int
- nextAdminUserNumber: int
~AccountManager()
~getPasswordManager() : PasswordManager
~ getLastCreatedAccount() : UserAccount
~setLoggedInUser(ua: UserAccount): void
~getLoggedInUser(): UserAccount
~createCardHolderAccount(name: String, email: String, password: String): void
~createAdminAccount(name: String, email: String, password: String): void
~findUserAccount(accountNumber: String): UserAccount
~isAdmin(ua: UserAccount): boolean
TransitSystem
-tripManager: TripManager
#transitManager: TransitManager
-accountManager: AccountManager
- cardManager: CardManager
- currentMonth: String
- currentDate: String
- operatingStatus= "off": String
- logger: Logger
- consoleHandler: Handler
-fh: Handler
+ TransitSystem(filePath String)
-getLogger(): Logger
+ log(level: Level, msg: String): void
~getTripManager(): TripManager
~getTransitManager(): TransitManager
~getAccountManager(): AccountManager
~getCardManager(): CardManager
~getOperatingStatus(): String
~powerOnSystem(): void
~powerOffSystem(): void
~setCurrentMonth(month: String): void
~setCurrentDate(date: String): void
+ readFromFile(path: String): void
+ saveToFile(filePath: String): void
TransitManager
- transitLines: HashMap<String, TransitLine>
~TransitManager()
-defaultSetup(): void
-createTransitLine(point: ArrayList<String>, type: String, id: String): TransitLine
- addTransitLine(name: String, line: TransitLine): void
TripManager
-transitLines: HashMap<String, TransitLine>
-fareCalculator: FareCalculator
-aggregator: Aggregator
-avgCostPerStation: Double
~TripManager()
~getAggregator(): Aggregator
~addTransitLines(transitlines: HashMap<String, TransitLine>): void
~getAvgCostPerStation(): Double
~setAvgGostPerStation(avgCostPerStation: Double): void
~recordTapIn(time: String, spot: String, card: Card, date: String, type: String): void
~recordTapOut(time: String, spot: String, card: Card, date: String, type: String): void
~addNewTrip(time: String, spot: String, card: Card, date: String): coid
~updateFares(fares, double, card: Card, date: String): void
AdminUser
~AdminUser(name: String, email: String, password: String)
PasswordManager
- securityQuestions: List<String>
~PasswordManager()
~getQuestionList(): List<String>
Card
-cardNumber: String
-balance: double
-owner: CardHolder
-linked: boolean
-active: boolean
-trips: ArrayList<TripSegment>
-totalFares: double
~ Card()
~ vieAllTrips(): void
~setBalance(balance: Double): void
+getTrips(): ArrayList<TripSegment>
~getBalance(): double
~ addBalance(fares: Double): void
deductBalance(fares: double): void
~viewBalance(): void
~activate(): void
~deactivate(): void
~linkAccount(): void
~unlinkAccount(): void
~isLinked(): boolean
~getCardNumber(): String
~getOwner(): CardHolder
~setOwner(owner: CardHolder): void
~equals(other: Card): boolean
~ getTotalFares(): double
~ viewMostRecentTrips(): void
~ addTrip(newtrip: TripSegment): void
~updateBalance(fares: double): void
~updateTotalFares(fares: double): void
+ setCArdNumber(cardNumber: String): void
~getStatus(): String
CardManager
- cards: ArrayList<Card>
~ CardManager()
~findCard(cardNumber: String): Card
~ getLastCard(): Card
- addCard(newCard Card): void
~createCard(): void
TransitLine
- points: ArrayList<String>
-type: String
-id: String
- numOfStops: int
~TransitLine(points: ArrayList<String>, type: String, id: String)
~getPoints(): ArrayList<String>
~setType(type: String): void
~getType(): String
~getId(): String
FareCalculator
- busFare= 2.0: double
-stationFare= 0.5: double
- fareCap= 6.0: double
- maximumDuration= 120: int
- transitLines: HashMap<String, TransitLine>
~getMaximumDuration(): int
~getFareCap(): double
~addTransitLines(transitLines: HashMap<String, TransitLine>): void
~ calculateContiBusFare(currentFares: double): double
~calculateContiSubFare(currentFares: double, distance: int): double
~calculateTripFares(ts: TripSegment): double
~ calculateStationsReached(trip: TripSegment): int
~ calculateDuration(lastStartTime: String, currentStartTime: String) : int
TripSegment
-enterSpot: String
-exitSpot: String
-contiSpot: String
-transitType: String
-enterTime: String
-exitTime: String
-date: String
-currentFares: double
+TripSegment(enterSpot: String, enterTime: String, enterDate: String, type: String)
~hasExit(): Boolean
+toString(): String
+getDate(): String
+getEnterSpot(): String
+ getExitSpot(): String
+ getEnterTime: String
+ getContiSpot: String
+ getTransitType: String
+getCurrentFares: double
+setDate(enterDate: String): void
+setExitSpot(exitSpot: String):void
+setExitTime(exitTime: String): void
+setContiSpot(contiSpot: String): void
+setCurrentFares(currentFares: double): void
+ vie
~equ
Aggregator
- allFares: HashMap<String, Double>
- numberOfStations: HashMap<String, Integer>
- availableDate: Set<String>
-transitLineDailySummary: HashMap<String, TransitLineDailyStat>
~Aggregator()
~getDailyFares(date: String) : Double
~getDailyStation(date: String) : Integer
~updateAllFares(date: String, fares: double) : void
~addNumberOfStation(date: String, n: int) : void
~ getDailyReport(date: String) : String
+ getTransitLineDailyStat(date: String) : TransitLineDailyStat
+ initializeDate(date: String): void
+ isReportAvailable(date: String): boolean
TransitLineDailyStat
-transitLineSummary:HashMap<String, SingleTransitLineDailyStat>
+ TransitLineDailyStat()
+ setSingleTransitLineDailyStat(id: String): SingleTransitLineDailyStat
+addSingleTransitLineDailyStat(id: String, STLDS: SingleTransitLineDailyStat): void
+isEmpty(): boolean
SingleTransitLineDailyStat
- id: String
-numOfTrips: Integer
-ridership: Interger
- avgRiderPerTrip: Integer
+ SingleTransitLineDailyStat(id: String,numOfTrips: Integer, ridership: Integer,, avgRiderPerTrip: Integer)
+ getId(): String
+ getNumOfTrips(): Integer
+ setNumOfTrips: void
+setNumOfTrips(numOfTrips: Integer): void
+getRidership(): Integer
+getAvgRiderPerTrip(): Integer