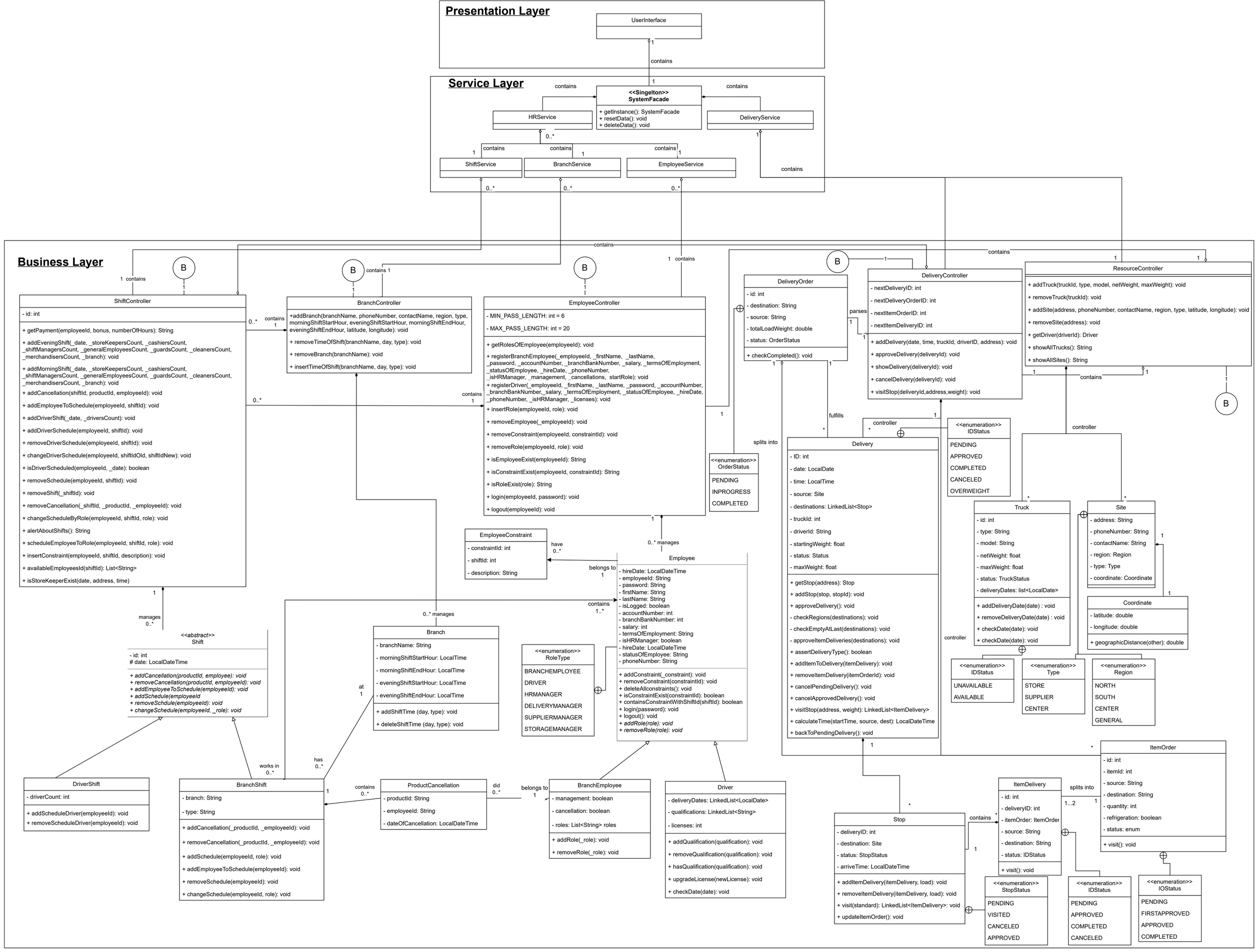
Presentation Layer
UserInterface
Service Layer
<<Singelton>>
SystemFacade
+ getInstance(): SystemFacade
+ resetData(): void
+ deleteData(): void
HRService
DeliveryService
ShiftService
BranchService
EmployeeService
Business Layer
ShiftController
- id: int
+ getPayment(employeeId, bonus, numberOfHours): String
+ addEveningShift(_date, _storeKeepersCount, _cashiersCount, _shiftManagersCount, _generalEmployeesCount, _guardsCount, _cleanersCount, _merchandisersCount, _branch): void
+ addMorningShift(_date, _storeKeepersCount, _cashiersCount, _shiftManagersCount, _generalEmployeesCount, _guardsCount, _cleanersCount, _merchandisersCount, _branch): void
+ addCancellation(shiftId, productId, employeeId): void
+ addEmployeeToSchedule(employeeId, shiftId): void
+ addDriverShift(_date, _driversCount): void
+ addDriverSchedule(employeeId, shiftId): void
+ removeDriverSchedule(employeeId, shiftId): void
+ changeDriverSchedule(employeeId, shiftIdOld, shiftIdNew): void
+ isDriverScheduled(employeeId, _date): boolean
+ removeSchedule(employeeId, shiftId): void
+ removeShift(_shiftId): void
+ removeCancellation(_shiftId, _productId, _employeeId): void
+ changeScheduleByRole(employeeId, shiftId, role): void
+ alertAboutShifts(): String
+ scheduleEmployeeToRole(employeeId, shiftId, role): void
+ insertConstraint(employeeId, shiftId, description): void
+ availableEmployeesId(shiftId): List<String>
+ isStoreKeeperExist(date, address, time)
BranchController
+addBranch(branchName, phoneNumber, contactName, region, type, morningShiftStartHour, eveningShiftStartHour, morningShiftEndHour, eveningShiftEndHour, latitude, longitude): void
+ removeTimeOfShift(branchName, day, type): void
+ removeBranch(branchName): void
+ insertTimeOfShift(branchName, day, type): void
EmployeeController
- MIN_PASS_LENGTH: int = 6
- MAX_PASS_LENGTH: int = 20
+ getRolesOfEmployee(employeeId): void
+ registerBranchEmployee(_employeeId, _firstName, _lastName, _password, _accountNumber, _branchBankNumber, _salary, _termsOfEmployment, _statusOfEmployee, _hireDate, _phoneNumber, _isHRManager, _management, _cancellations, startRole): void
+ registerDriver(_employeeId, _firstName, _lastName, _password, _accountNumber, _branchBankNumber, _salary, _termsOfEmployment, _statusOfEmployee, _hireDate, _phoneNumber, _isHRManager, _licenses): void
+ insertRole(employeeId, role): void
+ removeEmployee(_employeeId): void
+ removeConstraint(employeeId, constraintId): void
+ removeRole(employeeId, role): void
+ isEmployeeExist(employeeId): String
+ isConstraintExist(employeeId, constraintId): String
+ isRoleExist(role): String
+ login(employeeId, password): void
+ logout(employeeId): void
DeliveryOrder
- id: int
- destination: String
- source: String
- totalLoadWeight: double
- status: OrderStatus
+ checkCompleted(): void
DeliveryController
- nextDeliveryID: int
- nextDeliveryOrderID: int
- nextItemOrderID: int
- nextItemDeliveryID: int
+ addDelivery(date, time, truckId, driverID, address): void
+ approveDelivery(deliveryId): void
+ showDelivery(deliveryId): void
+ cancelDelivery(deliveryId): void
+ visitStop(deliveryId,address,weight): void
ResourceController
+ addTruck(truckId, type, model, netWeight, maxWeight): void
+ removeTruck(truckId): void
+ addSite(address, phoneNumber, contactName, region, type, latitude, longitude): void
+ removeSite(address): void
+ getDriver(driverId): Driver
+ showAllTrucks(): String
+ showAllSites(): String
<<enumeration>>
OrderStatus
PENDING
INPROGRESS
COMPLETED
Delivery
- ID: int
- date: LocalDate
- time: LocalTime
- source: Site
- destinations: LinkedList<Stop>
- truckId: int
- driverId: String
- startingWeight: float
- status: Status
- maxWeight: float
+ getStop(address): Stop
+ addStop(stop, stopId): void
+ approveDelivery(): void
- checkRegions(destinations): void
- checkEmptyAtLast(destinations): void
- approveItemDeliveries(destinations): void
+ assertDeliveryType(): boolean
+ addItemToDelivery(itemDelivery): void
+ removeItemDelivery(itemOrderId): void
+ cancelPendingDelivery(): void
+ cancelApprovedDelivery(): void
+ visitStop(address, weight): LinkedList<ItemDelivery>
+ calculateTime(startTime, source, dest): LocalDateTime
+ backToPendingDelivery(): void
<<enumeration>>
IDStatus
PENDING
APPROVED
COMPLETED
CANCELED
OVERWEIGHT
Truck
- id: int
- type: String
- model: String
- netWeight: float
- maxWeight: float
- status: TruckStatus
- deliveryDates: list<LocalDate>
+ addDeliveryDate(date) : void
+ removeDeliveryDate(date) : void
+ checkDate(date): void
+ checkDate(date): void
Site
- address: String
- phoneNumber: String
- contactName: String
- region: Region
- type: Type
- coordinate: Coordinate
Coordinate
- latitude: double
- longitude: double
+ geographicDistance(other): double
<<enumeration>>
IDStatus
UNAVAILABLE
AVAILABLE
<<enumeration>>
Type
STORE
SUPPLIER
CENTER
<<enumeration>>
Region
NORTH
SOUTH
CENTER
GENERAL
EmployeeConstraint
- constraintId: int
- shiftId: int
- description: String
Employee
- hireDate: LocalDateTime
- employeeId: String
- password: String
- firstName: String
- lastName: String
- isLogged: boolean
- accountNumber: int
- branchBankNumber: int
- salary: int
- termsOfEmployment: String
- isHRManager: boolean
- hireDate: LocalDateTime
- statusOfEmployee: String
- phoneNumber: String
+ addConstraint(_constraint): void
+ removeConstraint(constraintId): void
+ deleteAllconstraints(): void
+ isConstraintExist(constraintId): boolean
+ containsConstraintWithShiftId(shiftId): boolean
+ login(password): void
+ logout(): void
+ addRole(role): void
+ removeRole(role): void
<<enumeration>>
RoleType
BRANCHEMPLOYEE
DRIVER
HRMANAGER
DELIVERYMANAGER
SUPPLIERMANAGER
STORAGEMANAGER
Branch
- branchName: String
- morningShiftStartHour: LocalTime
- morningShiftEndHour: LocalTime
- eveningShiftStartHour: LocalTime
- eveningShiftEndHour: LocalTime
+ addShiftTime (day, type): void
+ deleteShiftTime (day, type): void
<<abstract>>
Shift
- id: int
date: LocalDateTime
+ addCancellation(productId, employee): void
+ removeCancellation(productId, employeeId): void
+ addEmployeeToSchedule(employeeId): void
+ addSchedule(employeeId
+ removeSchdule(employeeId): void
+ changeSchedule(employeeId, _role): void
DriverShift
- driverCount: int
+ addScheduleDriver(employeeId): void
+ removeScheduleDriver(employeeId): void
BranchShift
- branch: String
- type: String
+ addCancellation(_productId, _employeeId): void
+ removeCancellation(_productId, _employeeId): void
+ addSchedule(employeeId, role): void
+ addEmployeeToSchedule(employeeId): void
+ removeSchedule(employeeId): void
+ changeSchedule(employeeId, role): void
ProductCancellation
- productId: String
- employeeId: String
- dateOfCancellation: LocalDateTime
BranchEmployee
- management: boolean
- cancellation: boolean
- roles: List<String> roles
+ addRole(_role): void
+ removeRole(_role): void
Driver
- deliveryDates: LinkedList<LocalDate>
- qualifications: LinkedList<String>
- licenses: int
+ addQualification(qualification): void
+ removeQualification(qualification): void
+ hasQualification(qualification): void
+ upgradeLicense(newLicense): void
+ checkDate(date): void
Stop
- deliveryID: int
- destination: Site
- status: StopStatus
- arriveTime: LocalDateTime
+ addItemDelivery(itemDelivery, load): void
+ removeItemDelivery(itemDelivery, load): void
+ visit(standard): LinkedList<ItemDelivery>: void
+ updateItemOrder(): void
ItemDelivery
- id: int
- deliveryID: int
- itemOrder: ItemOrder
- source: String
- destination: String
- status: IDStatus
+ visit(): void
ItemOrder
- id: int
- itemId: int
- source: String
- destination: String
- quantity: int
- refrigeration: boolean
- status: enum
+ visit(): void
<<enumeration>>
StopStatus
PENDING
VISITED
CANCELED
APPROVED
<<enumeration>>
IDStatus
PENDING
APPROVED
COMPLETED
CANCELED
<<enumeration>>
IOStatus
PENDING
FIRSTAPPROVED
APPROVED
COMPLETED
DataAccessLayer
* For improved readability, all DAO classes inherit from AbstractDAO
<<Singelton>>
BranchDAO
- identityMap: Map<Identifier, Branch>
- instance: BranchDAO
+ getInstance(): BranchDAO
+ getBranch(_address): Branch
+ insert(_address,_phoneNumber,_contactNumber,_region, _type,_morningShiftStartHour,_eveningShiftStartHour, _morningShiftEndHour,_eveningShiftEndHour,_latitude, _longitude)
+ update(id,column,val)
+findAllBranches(): HashMap<String, Branch>
<<Private Class>>
Identifier
- address: String
+ same(_address): boolean
<<Singelton>>
EmployeeDAO
- identityMap: Map<Identifier, Employee>
- instance: EmployeeDAO
- constraintsToEmployeeDAO: ConstraintsToEmployeeDAO
+ getInstance(): EmployeeDAO
+ getEmployee(_employeeId): BranchEmployee
+ insert(_employeeId,_firstName,_lastName,_password,_accountNumber, _branchBankNumber,_salary,_termsOfEmployment,_statusOfEmployee, _hireDate,_phoneNumber,_isLogged,type)
+delete(_employeeId)
+findAllEmployees()
<<Private Class>>
Identifier
- employeeId: String
+ same(_employeeId): boolean
<<Singelton>>
DriverDeliveryDateDAO
- identityMap: Map<Identifier, List<LocalDate>>
- instance: DriverDeliveryDateDAO
+ getInstance(): DriverDeliveryDateDAO
+ getDriverDeliveyDates(_employeeId): List<LocalDate>>
+ insert(_employeeId,_deliveryDate)
+ delete(_employeeId,_deliveryDate)
+ loadData()
+ getDriverDeliveryDatesIM(_employeeId): LinkedList<LocalDate>
<<Singelton>>
DalFacade
+ getInstance(): DalFacade
+ deleteData(): void
<<abstract>>
AbstractDAO
path: String
connection: Connection
+ connect(): void
+ disconnect(): void
+ deleteData(): void
<<Singelton>>
DeliveryDAO
- identityMap: Map<Identifier, Delivery>
+ getInstance(): DeliveryDAO
+ insert(id, date, time, source, truckId, driverId, startingWeight, status, maxWeight): void
+ update(id, column, val): void
+ findAllDeliveries(): HashMap<Integer, Delivery>
<<Private Class>>
Identifier
- deliveryId: int
+ same(deliveryId): boolean
<<Singelton>>
DeliveryOrderDAO
- identityMap: Map<Identifier, DeliveryOrder>
+ getInstance(): DeliveryOrderDAO
+ insert(id, destination, source, totalLoadWeight, status): void
+ update(id, column, val): void
+ delete (deliveryOrderId): void
+ findallDeliveriesOrders(): HashMap<Integer, DeliveryOrder>
<<Private Class>>
Identifier
- deliveryOrderId: int
+ same(deliveryOrderId): boolean
<<Singelton>>
DriverQualificationDAO
- identityMap: Map<Identifier, List<String>>
- instance: DriverQualificationDAO
+ getInstrance(): DriverQualificationDAO
+ getDriverQualifications(_employeeId): List<String>>
+ insert(_employeeId,_qualification)
+ delete(_employeeId,_qualification)
+loadData()
+getDriverQualificationsIM(_employeeId) LinkedList<String>
<<Private Class>>
Identifier
- employeeId: String
- qualification: String
+same(_employeeId,_qualification): boolean
<<Singelton>>
DriverDAO
- identityMap: Map<Identifier, Driver>
- instance: DriverDAO
+ getInstance(): DriverDAO
+ getDriver(_employeeId): Driver
+ insert(_employeeId,_licenses)
+ update(id,column,val)
+ findAllDrivers()
<<Private Class>>
Identifier
- employeeId: String
+ same(_employeeId): boolean
<<Singelton>>
DriverShiftDAO
- identityMap: Map<Identifier, DriverShift>
- instance: DriverShiftDAO
+ getInstance(): DriverShift
+ getDriverShift(_shiftId): DriverShift
+ insert(_shiftId,_countDrivers)
+ update(id,column,val)
+ findAllDriverShifts(): HashMap<Integer, DriverShift>
<<Private Class>>
Identifier
- shiftId: int
+ same(_shiftId): boolean
<<Private Class>>
Identifier
- employeeId: String
- constraintId: int
+ same(_employeeId,_constraintId): boolean
<<Singelton>>
DriverScheduleDAO
- identityMap: Map<Identifier, List<String>>
- instance: DriverSchedule
+ getInstance(): DriverSchedule
+ getDriverSchedules(_shiftId): List<String>>
+ insert(_shiftId,_employeeId,_role)
+ delete(_shiftId,_employeeId,_role)
+ update(_shiftId,_employeeId,val)
+ loadData()
+ getDriverSchdulesIM(_shiftId): LinkedList<String>
<<Private Class>>
Identifier
- shiftId: int
- employeeId: String
+ same(_shiftId,_employeeId): boolean
<<Singelton>>
ConstraintToEmployeeDAO
- identityMap: Map<Identifier, List<Integer>>
- instance: ConstraintToEmployee
+ getInstance(): ConstraintToEmployee
+ getConstraintToEmployee(_employeeId): List<Integer>>
+ insert(_employeeId,_constraintId)
+ delete(_employeeId,_constraintId)
+ loadData()
+ getEmployeeCondtraintIM(_constraintId): EmployeeConstraint
<<Private Class>>
Identifier
- address: String
- timeOfShift: String
+ same(_address,_timeOfShift): boolean
<<Singelton>>
TimeOfShiftDAO
- identityMap: Map<Identifier, List<String>>
- instance: TimeOfShiftDAO
+ getInstrance(): TimeOfShiftDAO
+ getTimesOfShiftsToBrance(_address): List<String>>
+ insert(_address,_timeOfShift)
+ delete(_address,_timeOfShift)
+loadData()
+ getTimesOfShiftsToBranceIM(_address): List<String>>
<<Private Class>>
Identifier
- employeeId: String
+ same(_employeeId): boolean
<<Singelton>>
SiteDAO
- identityMap: Map<Identifier, Site>
+ getInstance(): SiteDAO
+ insert(address, phoneNumber, contactName, region, type, latitude, longitude): void
+ update(address, column, val): void
+ delete(address): void
+ findAllSites(): HashMap<Integer, Site>
<<Private Class>>
Identifier
- address: String
+ same(address): boolean
<<Private Class>>
Identifier
- stopId: int
+ same(stopId): boolean
<<Singelton>>
ItemDeliveryDAO
- identityMap: Map<Identifier, ItemDelivery>
+ getInstance(): ItemDeliveryDAO
+ insert(id, deliveryId, itemOrderId, source, destination, status): void
+ update(id, column, val): void
+ delete(itemDeliveryId): void
+ findAllItemDeliveries(): HashMap<Integer, ItemDelivery>
<<Singelton>>
StopDAO
- identityMap: Map<Identifier, Stop>
+ getInstance(): StopDAO
+ insert(id, deliveryId, destination, stopStatus, arriveTime): void
+ update(id, column, val): void
+ delete(id): void
+ findAllStops(): void
<<Singelton>>
ItemOrderDAO
- identityMap: Map<Identifier, ItemOrder>
+ getInstance(): ItemOrderDAO
+ insert(id, deliveryOrderId, itemId, source, destination, quantity, refrigeration, itemOrderStatus, status, itemDelivery1, itemDelivery2): void
+ update(id, column, val): void
+ delete(itemOrderId): void
+ findAllItemOrders(): HashMap<Integer, ItemOrder>
<<Private Class>>
Identifier
- itemOrderId: int
+ same(itemOrderId): boolean
<<Singelton>>
ShiftDAO
- identityMap: Map<Identifier, Shift>
- instance: ShiftDAO
+ getInstance(): ShiftDAO
+ insert(_shiftId,_date)
+ update(id,column,val)
<<Private Class>>
Identifier
- shiftId: int
+ same(_shidtId): boolean
<<Singelton>>
BranchShiftDAO
- identityMap: Map<Identifier, EveningShift>
- roleCountDAO: RoleCountDAO
- scheduleToRoleCountDAO: scheduleToRoleDAO
- scheduleDAO: ScheduleDAO
- instance: BranchShiftDAO
+ getInstance(): BranchShiftDAO
+ getBranchShift(_shiftId): BranchShift
+ insert(_shiftId,_branch,_type,_shiftManagerN,_storeKeeperN,_cashierN,_generalEmployeeN, _guardN,cleanerN,_merchandiserN)
+ update(id,column,val)
+ findAllBranchShifts(): HashMap<Integer, BranchShift>
<<Singelton>>
BranchEmployeeDAO
- identityMap: Map<Identifier, List<String>>
- instance: BranchEmployeeRoleDAO
+ getInstance(): BranchEmployeeDAO
+ getBranchEmployee(_employeeId): BranchEmployee
+ insert(_employeeId,_cancellation,_management)
+ delete_employeeId)
+ findAllBranchEmployees():HashMap<String,BranchEmployee>
<<Private Class>>
Identifier
- shiftId: int
+ same(_constraintId): boolean
<<Private Class>>
Identifier
- itemDeliveryId: int
+ same(itemDeliveryId): boolean
<<Singelton>>
StopUnloadItemsDAO
- identityMap: Map<Identifier, ItemDelivery>
+ getInstance(): StopUnloadItemsDAO
+ insert(stopId, itemDeliveryId): void
+ delete(stopId, itemDeliveryId): void
+ findAllStopUnloadItems(): void
<<Private Class>>
Identifier
- itemDeliveryId:int
+ same(itemDeliveryId): boolean
<<Singelton>>
ScheduleDAO
- identityMap: Map<Identifier, Map<String, String>>
- instance: ScheduleDAO
+ getInstance(): ScheduleDAO
+ getSchedules(_shiftId, _employeeId): HashMap<String,String>
+ insert(_shiftId,_employeeId,_role)
+ delete(_shiftId,_employeeId)
+ update(_shiftId,_employeeId,val)
+ loadData()
+ getScheduleIM(_shiftId): HashMap<String, String>
<<Singelton>>
RoleCountDAO
- identityMap: Map<Identifier, Map<String, Integer>>
- instance: RoleCountDAO
+ getInstance(): RoleCountDAO
+ getRoleCountAssigned(_shiftId):HashMap<String,Integer>
+ getRoleCountNeeded(_shiftId):HahMap<String,Integer>
+ insert(_shiftId,_role,_countNeeded,_countAssigned)
+ update(id,role,column,val)
+ loadData()
+ getRoleCountAssignedIM(_shiftId):HashMap<String,Integer>
+ getRoleCountNeededIM(_shiftId):HahMap<String,Integer>
<<Singelton>>
ProductCancellationToShiftDAO
- identityMap: Map<Identifier, ProductCancellation>
- instance: ProductCancellationToShiftDAO
+ getInstance(): ProductCancellationToShiftDAO
+ getPCToShift(shiftId): List<ProductCancellation>
+ insert(_productId,_employeeId,_shiftId)
+ delete(_productId,_employeeId,_shiftId)
+ loadData(): void
+ getPCToShiftIM(shiftId): List<ProductCancellation>
<<Singelton>>
BranchEmployeeRoleDAO
- identityMap: Map<Identifier, List<String>>
- instance: BranchEmployeeRoleDAO
+ getInstance(): BranchEmployeeDAO
+getBranchEmployeeRole(_employeeId):BranchEmployee
+ insert(_employeeId,_role)
+ delete_employeeId,_role)
+ loadData()
+ getBranchEmployeeRolesIM(): LinkedList<String>
<<Private Class>>
Identifier
- productId: String
- employeeId: String
- shiftId: int
+ same(productId,_employeeId,_shiftId): boolean
<<Private Class>>
Identifier
- employeeId: String
- role
+ same(_employeeId,_role): boolean
<<Private Class>>
Identifier
- constraintId: int
+ same(_constraintId): boolean
<<Singelton>>
EmployeeConstraintDAO
- identityMap: Map<Identifier, EmployeeConstraint>
- instance: EmployeeConstraintDAO
+ getInstance(): EmployeeConstraintDAO
+ getEmployeeConstraint(_constraintId): EmployeeConstraint
+ insert(_constraintId,_shiftId,description)
+ delete(_constraintId)
+ update(id,column,val)
+ loadData()
<<Singelton>>
StopLoadItemsDAO
- identityMap: Map<Identifier, ItemDelivery>
+ getInstance(): StopLoadItemsDAO
+ insert(stopId, itemDeliveryId): void
+ delete(stopId, itemDeliveryId): void
+ findAllStopLoadItems(stopId): void
<<Singelton>>
TruckDeliveryDatesDAO
- identityMap: Map<Identifier,LocalDate>
+ getInstance(): TruckDeliveryDatesDAO
+ getDeliveryDates(truckId): LinkedList<LocalDate>
+ insert(id, date): void
+ delete(truckId,d): void
+ findAllTruckDeliveryDates(): void
<<Private Class>>
Identifier
- truckId: int
+ same(truckId): boolean
<<Singelton>>
TruckDAO
- identityMap: Map<Identifier,Truck>
+ getInstance(): TruckDAO
+ getTruck(truckId): Truck
+ Insert(id, type, model, netWeight, maxWeight, status): void
+ update(id,column,val): void
+ delete(truckId): void
+ findAllTrucks(): HashMap<Integer, Truck>