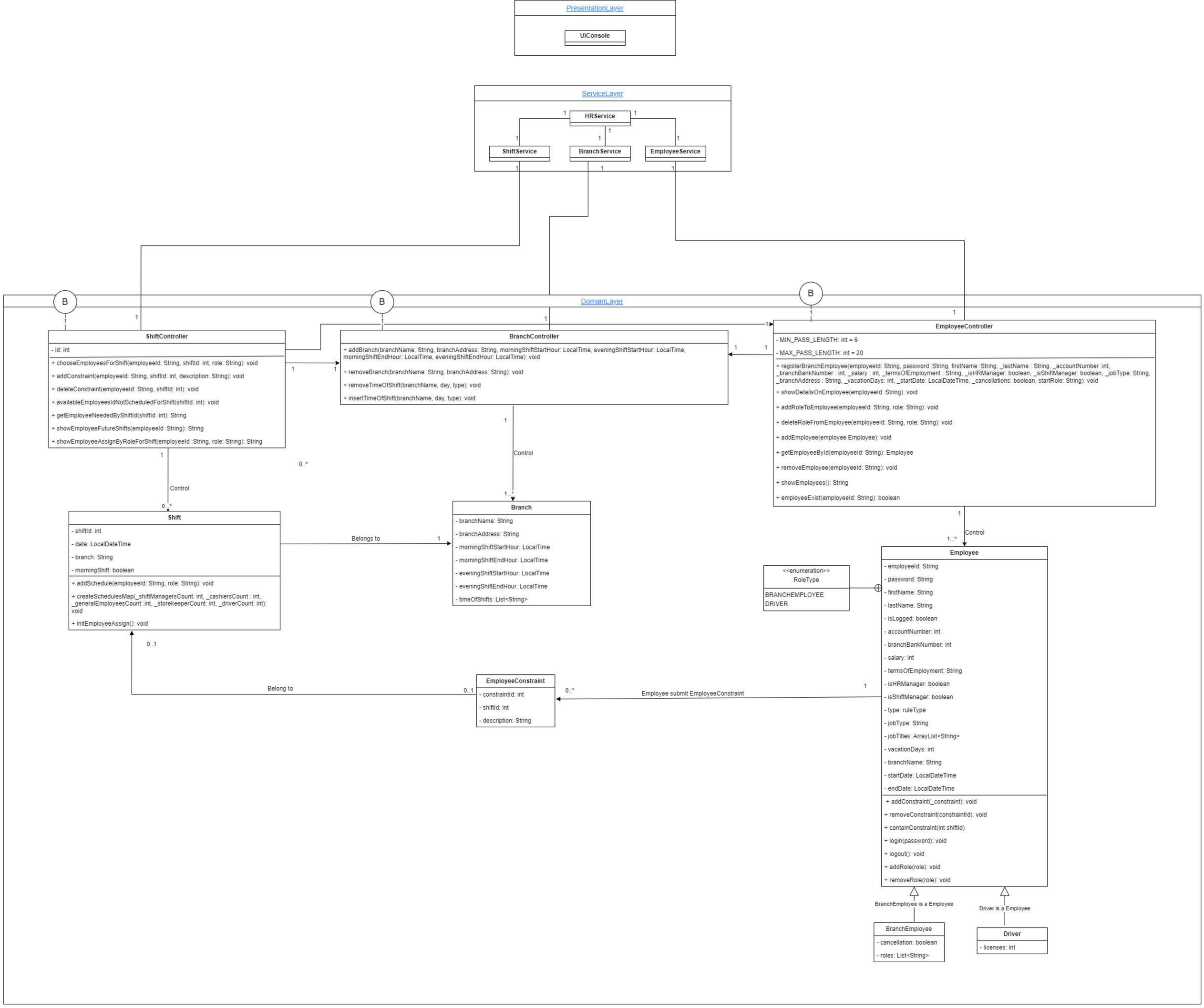
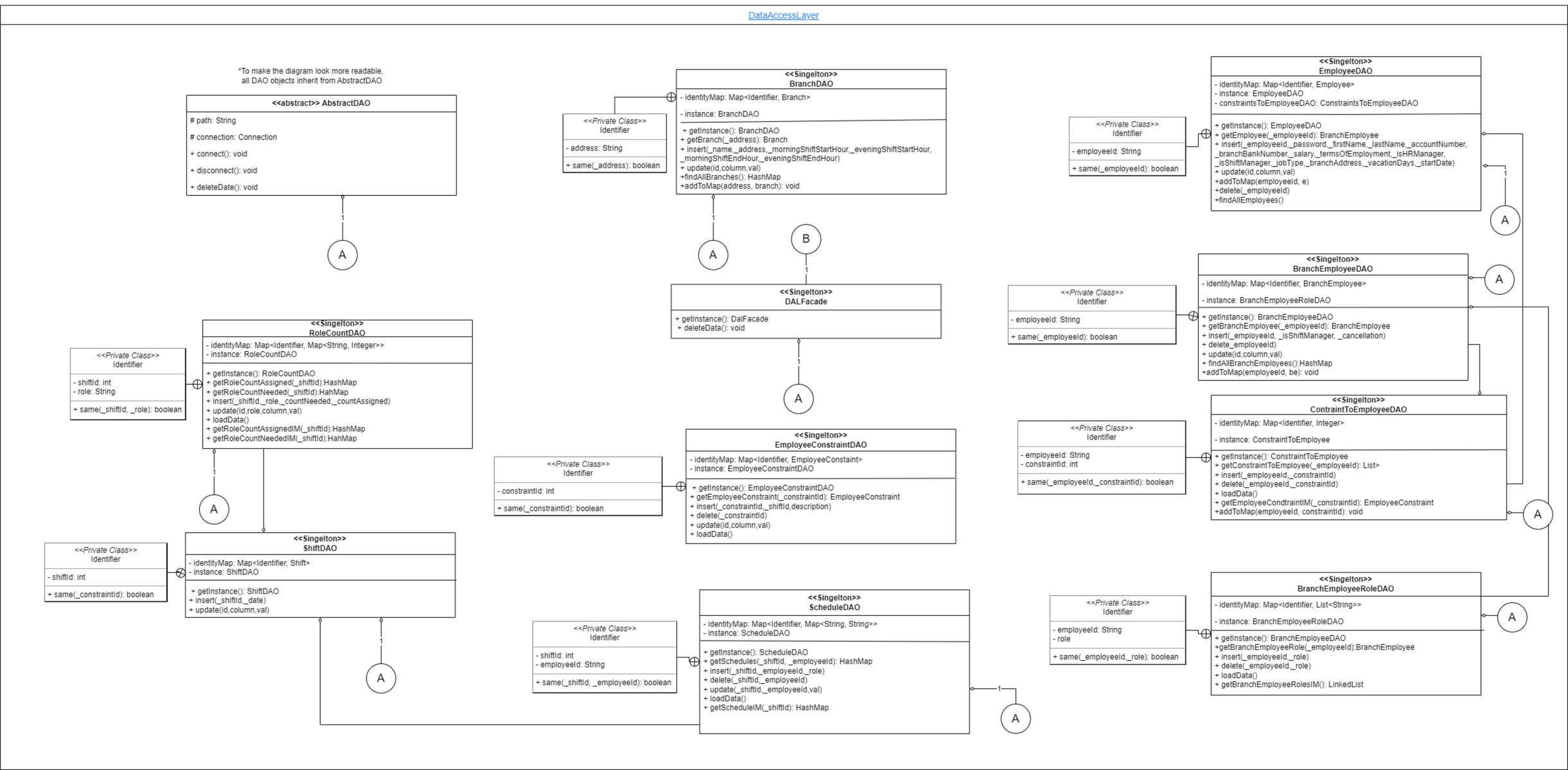

PresentationLayer
UIConsole
ServiceLayer
HRService
ShiftService
BranchService
EmployeeService
DomainLayer
ShiftController
- id: int
+ chooseEmployeesForShift(employeeId: String, shiftId: int, role: String): void
+ addConstraint(employeeId: String, shiftId: int, description: String): void
+ deleteConstraint(employeeId: String, shiftId: int): void
+ availableEmployeesIdNotScheduledForShift(shiftId: int): void
+ getEmployeeNeededByShiftId(shiftId :int): String
+ showEmployeeFutureShifts(employeeId :String): String
+ showEmployeeAssignByRoleForShift(employeeId :String, role: String): String
BranchController
+ addBranch(branchName: String, branchAddress: String, morningShiftStartHour: LocalTime, eveningShiftStartHour: LocalTime, morningShiftEndHour: LocalTime, eveningShiftEndHour: LocalTime): void
+ removeBranch(branchName: String, branchAddress: String): void
+ removeTimeOfShift(branchName, day, type): void
+ insertTimeOfShift(branchName, day, type): void
EmployeeController
- MIN_PASS_LENGTH: int = 6
- MAX_PASS_LENGTH: int = 20
+ registerBranchEmployee(employeeId: String, password :String, firstName :String, _lastName : String, _accountNumber :int, _branchBankNumber : int, _salary : int, _termsOfEmployment : String, _isHRManager: boolean, _isShiftManager: boolean, _jobType: String, _branchAddress : String, _vacationDays: int, _startDate: LocalDateTime, _cancellations: boolean, startRole: String): void
+ showDetailsOnEmployee(employeeId: String): void
+ addRoleToEmployee(employeeId: String, role: String): void
+ deleteRoleFromEmployee(employeeId: String, role: String): void
+ addEmployee(employee Employee): void
+ getEmployeeById(employeeId: String): Employee
+ removeEmployee(employeeId: String): void
+ showEmployees(): String
+ employeeExist(employeeId: String): boolean
Control
Control
Control
Shift
- shiftId: int
- date: LocalDateTime
- branch: String
- morningShift: boolean
+ addSchedule(employeeId: String, role: String): void
+ createSchedulesMap(_shiftManagersCount: int, _cashiersCount : int, _generalEmployeesCount :int, _storekeeperCount: int, _driverCount: int): void
+ initEmployeeAssign(): void
Belongs to
Branch
- branchName: String
- branchAddress: String
- morningShiftStartHour: LocalTime
- morningShiftEndHour: LocalTime
- eveningShiftStartHour: LocalTime
- eveningShiftEndHour: LocalTime
- timeOfShifts: List<String>
<<enumeration>>
RoleType
BRANCHEMPLOYEE
DRIVER
Employee
- employeeId: String
- password: String
- firstName: String
- lastName: String
- isLogged: boolean
- accountNumber: int
- branchBankNumber: int
- salary: int
- termsOfEmployment: String
- isHRManager: boolean
- isShiftManager: boolean
- type: ruleType
- jobType: String
- jobTitles: ArrayList<String>
- vacationDays: int
- branchName: String
- startDate: LocalDateTime
- endDate: LocalDateTime
+ addConstraint(_constraint): void
+ removeConstraint(constraintId): void
+ containConstraint(int shiftId)
+ login(password): void
+ logout(): void
+ addRole(role): void
+ removeRole(role): void
Belong to
EmployeeConstraint
- constraintId: int
- shiftId: int
- description: String
Employee submit EmployeeConstraint
BranchEmployee is a Employee
Driver is a Employee
BranchEmployee
- cancellation: boolean
- roles: List<String>
Driver
- licenses: int


DataAccessLayer
*To make the diagram look more readable, all DAO objects inherit from AbstractDAO
<<abstract>> AbstractDAO
path: String
connection: Connection
+ connect(): void
+ disconnect(): void
+ deleteDate(): void
<<Private Class>> Identifier
- address: String
+ same(_address): boolean
<<Singelton>> BranchDAO
- identityMap: Map<Identifier, Branch>
- instance: BranchDAO
+ getInstance(): BranchDAO
+ getBranch(_address): Branch
+ insert(_name,_address,_morningShiftStartHour,_eveningShiftStartHour, _morningShiftEndHour,_eveningShiftEndHour)
+ update(id,column,val)
+findAllBranches(): HashMap
+addToMap(address, branch): void
<<Private Class>> Identifier
- employeeId: String
+ same(_employeeId): boolean
<<Singelton>> EmployeeDAO
- identityMap: Map<Identifier, Employee>
- instance: EmployeeDAO
- constraintsToEmployeeDAO: ConstraintsToEmployeeDAO
+ getInstance(): EmployeeDAO
+ getEmployee(_employeeId): BranchEmployee
+ insert(_employeeId,_password,_firstName,_lastName,_accountNumber, _branchBankNumber,_salary,_termsOfEmployment,_isHRManager, _isShiftManager,_jobType,_branchAddress,_vacationDays,_startDate)
+ update(id,column,val)
+addToMap(employeeId, e)
+delete(_employeeId)
+findAllEmployees()
<<Singelton>> DALFacade
+ getInstance(): DalFacade
+ deleteData(): void
<<Singelton>> BranchEmployeeDAO
- identityMap: Map<Identifier, BranchEmployee>
- instance: BranchEmployeeRoleDAO
+ getInstance(): BranchEmployeeDAO
+ getBranchEmployee(_employeeId): BranchEmployee
+ insert(_employeeId, _isShiftManager, _cancellation)
+ delete_employeeId)
+ update(id,column,val)
+ findAllBranchEmployees():HashMap
+addToMap(employeeId, be): void
<<Private Class>> Identifier
- employeeId: String
+ same(_employeeId): boolean
<<Singelton>> RoleCountDAO
- identityMap: Map<Identifier, Map<String, Integer>>
- instance: RoleCountDAO
+ getInstance(): RoleCountDAO
+ getRoleCountAssigned(_shiftId):HashMap
+ getRoleCountNeeded(_shiftId):HahMap
+ insert(_shiftId,_role,_countNeeded,_countAssigned)
+ update(id,role,column,val)
+ loadData()
+ getRoleCountAssignedIM(_shiftId):HashMap
+ getRoleCountNeededIM(_shiftId):HahMap
<<Private Class>> Identifier
- shiftId: int
- role: String
+ same(_shiftId, _role): boolean
<<Singelton>> ContraintToEmployeeDAO
- identityMap: Map<Identifier, Integer>
- instance: ConstraintToEmployee
+ getInstance(): ConstraintToEmployee
+ getConstraintToEmployee(_employeeId): List>
+ insert(_employeeId,_constraintId)
+ delete(_employeeId,_constraintId)
+ loadData()
+ getEmployeeCondtraintIM(_constraintId): EmployeeConstraint
+addToMap(employeeId, constraintId): void
<<Private Class>> Identifier
- employeeId: String
- constraintId: int
+ same(_employeeId,_constraintId): boolean
<<Private Class>> Identifier
- constraintId: int
+ same(_constraintId): boolean
<<Singelton>> EmployeeConstraintDAO
- identityMap: Map<Identifier, EmployeeConstaint>
- instance: EmployeeConstraintDAO
+ getInstance(): EmployeeConstraintDAO
+ getEmployeeConstraint(_constraintId): EmployeeConstraint
+ insert(_constraintId,_shiftId,description)
+ delete(_constraintId)
+ update(id,column,val)
+ loadData()
<<Private Class>> Identifier
- shiftId: int
+ same(_constraintId): boolean
<<Singelton>> ShiftDAO
- identityMap: Map<Identifier, Shift>
- instance: ShiftDAO
+ getInstance(): ShiftDAO
+ insert(_shiftId,_date)
+ update(id,column,val)
<<Singelton>> ScheduleDAO
- identityMap: Map<Identifier, Map<String, String>>
- instance: ScheduleDAO
+ getInstance(): ScheduleDAO
+ getSchedules(_shiftId, _employeeId): HashMap
+ insert(_shiftId,_employeeId,_role)
+ delete(_shiftId,_employeeId)
+ update(_shiftId,_employeeId,val)
+ loadData()
+ getScheduleIM(_shiftId): HashMap
<<Private Class>> Identifier
- shiftId: int
- employeeId: String
+ same(_shiftId, _employeeId): boolean
<<Private Class>> Identifier
- employeeId: String
- role
+ same(_employeeId,_role): boolean
<<Singelton>> BranchEmployeeRoleDAO
- identityMap: Map<Identifier, List<String>>
- instance: BranchEmployeeRoleDAO
+ getInstance(): BranchEmployeeDAO
+getBranchEmployeeRole(_employeeId):BranchEmployee
+ insert(_employeeId,_role)
+ delete(_employeeId,_role)
+ loadData()
+ getBranchEmployeeRolesIM(): LinkedList
A
B