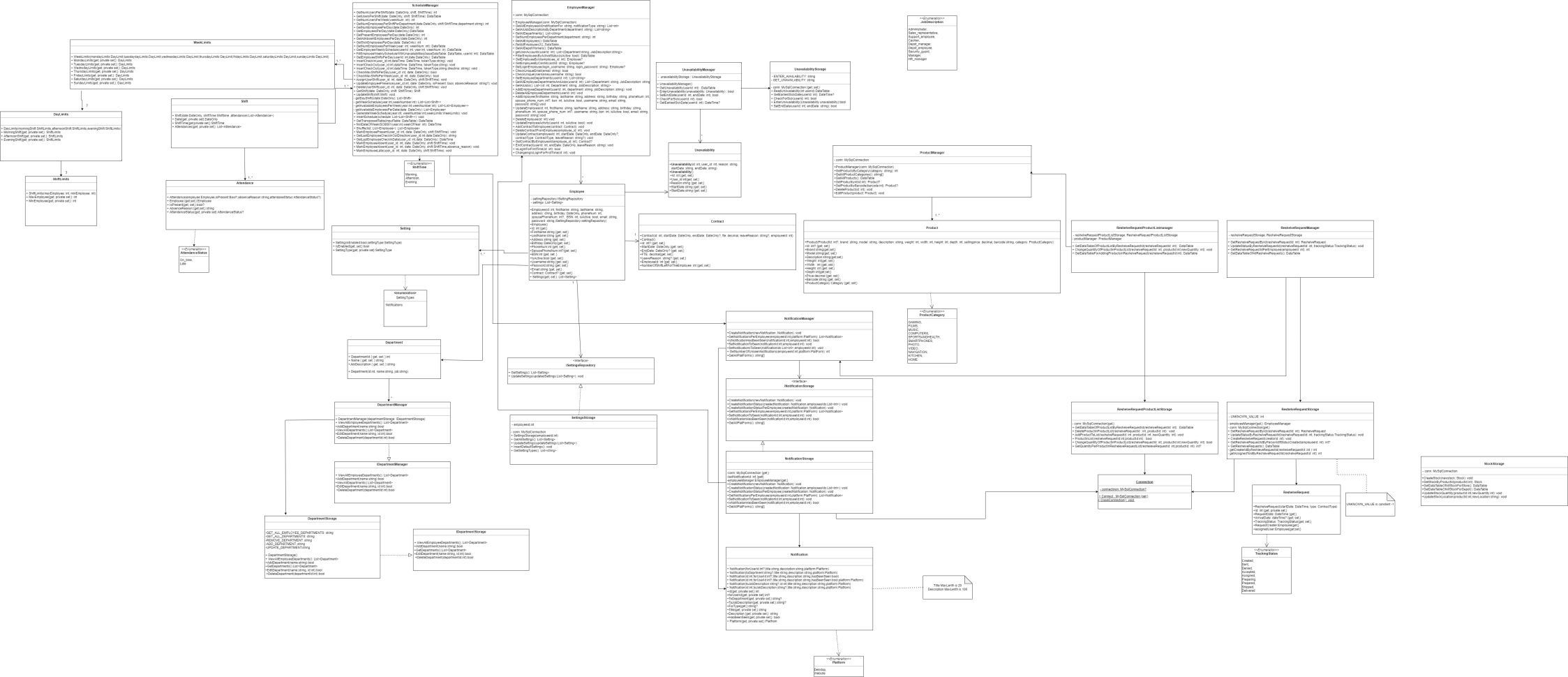
WeekLimits
+ WeekLimits(monday:Limit; DayLimit tuesday:DayLimit; DayLimit wednesday:Limit; DayLimit thursday:Limit; DayLimit friday:Limit; DayLimit saturday:DayLimit; DayLimit sunday:Limits DayLimit)
+ Monday:Limit(get; private set): DayLimits
+ Tuesday:Limit(get; private set): DayLimits
+ Wednesday:Limit(get; private set): DayLimits
+ Thursday:Limit(get; private set): DayLimits
+ Friday:Limit(get; private set): DayLimits
+ Saturday:Limit(get; private set): DayLimits
+ Sunday:Limit(get; private set): DayLimits
DayLimits
+ DayLimits(morningShift:ShiftLimits; afternoonShift:ShiftLimits; eveningShift:ShiftLimits)
+ MorningShift(get; private set): ShiftLimits
+ AfternoonShift(get; private set): ShiftLimits
+ EveningShift(get; private set): ShiftLimits
ShiftLimits
+ ShiftLimits(minEmployee: int, minEmployee: int)
+ MinEmployee(get; private set): int
+ MinEmployee(get; private set): int
Shift
+ Shift(date:DateOnly, shiftTime:ShiftTime, attendances:List<Attendance>)
+ Date(get; private set):DateOnly
+ ShiftTime(get;private set): ShiftTime
+ Attendances(get; private set): List<Attendance>
Attendance
+ Attendance(employee:Employee,isPresent:Bool?,absenceReason:string,attendanceStatus:AttendanceStatus?)
+ Employee(get; private set):Employee
+ IsPresent(get; set):bool?
+ AbsenceReason:(get;set):string
+ AttendanceStatus(get; private set):AttendanceStatus?
<<Enumeration>>
AttendanceStatus
On_time
Late
ScheduleManager
+ GetNumUsersPerShift(date: DateOnly, shift: ShiftTime): int
+ GetUsersPerShift(date: DateOnly, shift: ShiftTime): DataTable
+ GetNumUsersPerWeek(week: int): int
+ GetNumEmployeesPerDay(date:DateOnly, shiftTime:ShiftTime,department:string): int
+ GetNumEmployeesPerDay(date:DateOnly): int
+ GetEmployeesPerDay(date:DateOnly):DataTable
+ GetEmployeesPerDay(date:DateOnly): DataTable
+ GetNumEmployeesPerDay(departmentDepartment: string): int
+ GetShiftEmployeesPerDay(date: DateOnly): int
+ GetNumEmployeesPerWeek(week: int): int
+ GetNumEmployeesPerWeek(year: int, weekNum: int): DataTable
+ GetEmployeesPerWeek(scheduleusuario: int, year:int, weekNum: int):DataTable
+ GetEmployeesPerWeek(year:int,weekNum:int,department:string):DataTable, user:int): DataTable
+ GetEmployeesShiftPerDay(date:DateOnly): DataTable
+ GetEmployeesByIdSemana(week: int): Employee[*]
+ InsertCheckIn(user_id:int,date:DateTime, tokenType:string): void
+ InsertCheckOut(user_id:int,date:DateTime, tokenType:string): void
+ InsertCheckIn(user_id:int,date:DateTime, tokenType:string,direction: string): void
+ InsertCheckOut(user_id:int,date:DateTime, tokenType:string): void
+ CheckIsNoShiftPerWeek(user_id: int, date: DateOnly): bool
+ AssignUserShift(user_id: int, date: DateOnly, shift:ShiftTime): void
+ ChangeUserShift(user_id: int, date: DateOnly, shift:ShiftTime): void
+ DeleteUserShift(user_id: int, date: DateOnly, shift:ShiftTime): void
+ GetShift(date: DateOnly, shift: ShiftTime): Shift
+ UpdateShift(shift:Shift): void
+ getDayShift(date:DateOnly): List<Shift>
+ getAvailableEmployeesPerWeek(week:int): List<List<Shift>>
+ getAvailableEmployeesPerWeek(year:int,weekNum:int): List<List<Employee>>
+ getAvailableEmployeesPerDate(date: DateOnly): List<Employee>
+ getAvailableEmployeesPerDate(date:DateOnly,shiftTime:ShiftTime): List<Employee>
+ InsertSchedule(schedule:List<List<Shift>>): void
+ DeTransposeTable(inputTable:DataTable): DataTable
+ fnGetDaysOfWeek(DG0001(year:int,weekOfYear: int): DateTime
+ fnTheBest: List<Employee>; List<Employee>
+ GetLastEmployeeCheckIn(date:DateOnly, shiftTime: ShiftTime): void
+ GetLastEmployeeCheckOut(date:DateOnly, shiftTime: ShiftTime): void
+ GetLastEmployeeCheckIn(date:user_id:int,date:DateOnly): DateTime
+ MarkEmployeeAbsent(user_id: int, date: DateOnly, shiftTime: void
+ MarkEmployeeAbsent(user_id: int, date: DateOnly, shiftTime:absence_reason: void
+ MarkEmployeeLate(user_id: int, date: DateOnly, shiftTime): void
<<Enumeration>>
ShiftTime
Morning
Afternoon
Evening
EmployeeManager
- conn: MySqlConnection
+ EmployeeManager(conn: MySqlConnection)
+ GetAllEmployees(notificationFor: string, notificationType: string): List<int>
+ GetAllJobDescriptions(p:DepartmentDepartment: string): List<string>
+ GetAllJobDescriptions(): List<string>
+ GetNumEmployeesPerDepartment(department: string): int
+ GetAllEmployees(): DataTable
+ GetEmployee(id:int): Employee
+ GetAllDepartments(): DataTable
+ getUsersAccounts(user: int): List<DepartmentDrive, jobDescription:string>
+ FilterEmployeesByAuthorizationLevel(level: int): DataTable
+ GetEmployeeByIdSemana(week: int): Employee[*]
+ GetEmployeeByIdSemana(week: int, year: int): Employee[*]
+ GetLoginEmployee(login_username: string, login_password: string): Employee?
+ CheckUniqueEmail(email: string): bool
+ CheckUniqueUsername(username: string): bool
+ GetEmployeeDepartmentAndJobs(user: int): List<int>
+ GetAllEmployeesDepartmentAndJobs(user: int): List<DepartmentDrive, jobDescription: string>
+ GetAllJobs(): List<Job id, Department: string, jobDescription: string>
+ AddEmployeeDepartment(user: int, department: string, jobDescription: string): void
+ UpdateEmployeeDepartment(user: int): void
+ AddEmployee(firstName: string, lastName: string, address: string, birthday: string, pinarkum: int, spouse_phone_num: int?, ssn: int, isActive: bool, username: string, email: string, password: string): void
+ UpdateEmployee(id: int, firstName: string, lastName: string, address: string, birthday: string, pinarkum: int, spouse_phone_num: int?, username: string, ssn: int, isActive: bool, email: string, password: string): void
+ DeleteEmployee(id: int): void
+ ChangeEmployeeStatus(id: int, isActive: bool): void
+ AddContractToEmployee(contract: Contract): void
+ DeleteContractFromEmployee(employee_id: int): void
+ UpdateContractEmployee(id: int, startDate: DateOnly, endDate: DateOnly?, contractType: ContractType, leaveRetain: string?): void
+ EmployeeCheckIn(id: int, endDate: DateOnly, leaveRetain: string): void
+ EmployeeCheckOut(id: int, endDate: DateOnly, leaveRetain: string): void
+ isLeftForFirstTime(id: int): bool
+ ChangeLeftLeftForFirstTime(id: int): void
UnavailabilityManager
- unavailability:Storage , UnavailabilityStorage
+ UnavailabilityManager()
+ GetUnavailability(user:int): DataTable
+ EnterUnavailability(unavailability: Unavailability): bool
+ GetAllUnDates(user:int): int, endDate: int): bool
+ CheckForDuplicate(id: int): bool
+ GetEarliestStartDate(user:int): int: DateTime?
UnavailabilityStorage
- ENTER_AVAILABILITY: string
- GET_UNAVAILABILITY: string
- conn: MySqlConnection (get; set)
+ ReadUnAvailability(int uSerID):DataTable
+ GetEarliestStartDate(user:int): DateTime?
+ CheckForDuplicate(id: int): bool
+ EnterUnavailability(unavailability: unavailability):void
+ GetAllUnDates(user:int: endDate: int): bool
Unavailability
+ Unavailability(id: int, user_id: int, reason: string, startDate: string, endDate: string)
+ id: int (get; set)
+ uid: int (get; set)
+ User_id:int(get; set)
+ Reason:string (get; set)
+ StartDate:string (get; set)
+ EndDate:string (get; set)
<<Enumeration>>
JobDescription
Administrator
Sales_representative
Support_employee
Cashier
Store_manager
Detail_employee
Security_guard
Manager
HR_manager
ProductManager
- conn: MySqlConnection
+ ProductManager(conn: MySqlConnection)
+ GetAllProductsByCategory(category: int)
+ GetAllProductCategories(): string[]
+ GetAllProducts(): DataTable
+ GetProductBy(id:int): Product?
+ GetProductByBarcode(barcode:int): Product?
+ GetProductBy(id: int)
+ EditProduct(product: Product): void
Product
+ Product(productid: int?, brand: string, model: string, description: string, weight: int, width: int, height: int, depth: int, sellingprice: decimal, barcode:string, category: ProductCategory)
+ id: int (get; set)
+ Brand:string(get; set)
+ Model:string(get; set)
+ Description:string(get;set)
+ Height: int(get; set)
+ Width: int (get; set)
+ Height:int(get; set)
+ Depth:int(get; set)
+ Price:decimal (get; set)
+ Barcode:string (get; set)
+ ProductCategory: Category (get; set)
<<Enumeration>>
ProductCategory
DIAMOND,
FILMS,
MUSIC,
COMPUTERS,
HOME_APPLIANCE,
BAKATTACHES,
PHOTO,
VIDEO,
INFORMATION,
GARDEN,
HOME,
RechieveRequestProductListManager
- rechieveRequestProductListStorage: RechieveRequestProductListStorage
- productManager: ProductManager
+ GetDataTableOfProductList(RechieveRequestid:rechieveRequestid: int): DataTable
+ ChangeQuantityOfProductInList(rechieveRequestid: int productid:int newQuantity: int): void
+ GetDataTableFromStringProducts(RechieveRequestrechieveRequestid:int): DataTable
RestoreRequestManager
- rechieveRequestStorage: RechieveRequestStorage
+ GetRechieveRequestBy(id:rechieveRequestid: int): RechieveRequest
+ UpdateStatus(rechieveRequestrechieveRequestid: int, trackingStatus:TrackingStatus): void
+ GetRechieveRequestsOfPeriod(employeeuserid: int): int
+ GetDataTableOfAllRechieveRequests(): DataTable
Connection
- connection: MySqlConnection?
+ Connect_MySqlConnection: net()
+ CloseConnection(): void
RestoreRequestProductListStorage
- conn: MySqlConnection(get;)
+ GetDataTableOfProductList(RestoreRequestrechieveRequestid: int): DataTable
+ DeleteProductOfProductList(rechieveRequestid: int productid: int): void
+ AddProductToProductList(rechieveRequestid: int productid: int, newQuantity: int): void
+ ProductExistsInrechieveRequestid(int productid): bool
+ ChangeQuantityOfProductInProductList(rechieveRequestid: int, productid:int,newQuantity: int): void
+ GetQuantityPerProductInRechieveRequestList(rechieveRequestid: int,productid: int): int?
RestoreRequestStorage
- UNKNOWN_VALUE: int
- employeeManager(get;): EmployeeManager
- conn: MySqlConnection(get;)
+ GetRechieveRequestBy(rechieveRequestid: int): RechieveRequest
+ GetRechieveRequestsBy(rechieveRequestrechieveRequestid: int, trackingStatus:TrackingStatus): void
+ CreateRechieveRequest(creatorid: int): void
+ GetRechieveRequestsBy(PersonIdOfStatusCreatorsemployeeid: int): int?
+ GetRechieveRequests(): DataTable
+ getCreatorBy(RechieveRequestrechieveRequestid: int): int
+ getAssignedToBy(RechieveRequestrechieveRequestid: int): int
UNKNOWN_VALUE is constant = 1
RestoreRequest
+ RechieveRequest(startDate: DateTime, type: ContractType): void
+ id: int (get; private set)
+ CreatorId:int (get; private set)
+ startDate: dateTime? (get; set;)
+ TrackingStatus: TrackingStatus(get; set;)
+ RequestCreator:Employee(get;)
+ assignedUser:Employee(get;set;)
<<Enumeration>>
TrackingStatus
Created,
Sent,
Denied,
Accepted,
Assigned,
Preparing,
Prepared,
Shipped,
Delivered
StockManager
- conn: MySqlConnection
+ CreateStock(newstock: Stock): void
+ GetStockBy(Productsproductid:int): Stock
+ GetDataTableOfAllStockFromStore(): DataTable
+ GetDataTableOfAllStockFromDepot(): DataTable
+ ChangeQuantityOfProductInStock(productid: int, newQuantity: int): void
+ UpdateStockLocation(productid:int,newLocation:string): void
Employee
- settingRepository:ISettingRepository
- settings: List<Setting>
+ Employee(id: int, firstName: string, lastName: string, address: string, birthday: DateOnly, pinarkum: int, spousePhoneNum: int?, SSN: int, isActive: bool, email: string, password: string, SettingRepository:settingRepository)
+ id: int (get;)
+ Id: int (get;)
+ FirstName:string (get; set;)
+ LastName:string (get; set;)
+ Address:string (get;)
+ Birthday:DateOnly(get; set;)
+ PinorKarum:int (get; set;)
+ SpousePhoneNum:int?(get; set;)
+ SSN:int (get; set;)
+ IsActive:bool (get; set;)
+ Username:string (get; set;)
+ Password:string (get; set;)
+ Email:string (get; set;)
+ Contract: Contract? (get; set;)
+ Settings(get; set;): List<Setting>
Contract
+ Contract(id: int, startDate:DateOnly, endDate:DateOnly?, fte: decimal, leaveRetain: string?, employeeId: int)
+ id: int (get;)
+ Id: int (get; set;)
+ StartDate: DateOnly (get; set;)
+ EndDate: DateOnly? (get; set;)
+ FTE: decimal(get; set;)
+ LeaveRetain:string (get; set;)
+ Employee:int (get; set;)
+ NumberOfSHIftLeftForTheEmployee: int (get; set;)
Setting
+ Setting(id:nullable:bool,settingType:SettingType)
+ IsEnable(get; set;): bool
+ SettingType(get; private set): SettingType
<<enumeration>>
SettingTypes
Notifications
Department
+ Department(id : (get; set;): int
+ Name: (get; set;): string
+ JobDescription : (get; set;): string
+ Department(id:int, name:string, job:string)
<<Interface>>
ISettingsRepository
+ GetSettings(): List<Setting>
+ UpdateSettings(updateSettings:List<Setting>): void
NotificationManager
+ CreateNotification(newNotification: Notification): void
+ CreateNotificationFor(employeeIds:List<int>, platformPlatform): List<Notification>
+ GetNotificationsPerEmployee(employeeId:int,employeeId: int): bool
+ SetNotificationToSeen(notificationId:int,employeeId: int): void
+ GetNotificationTOSeenNotificationId:int,List<int> employeeId: int): void
+ GetNumberOfUnseenNotifications(employeeId:int,platformPlatform): int
+ GetAllPlatforms(): string[]
<<Interface>>
INotificationStorage
+ CreateNotification(newNotification: Notification): void
+ CreateNotificationStatus(createdNotification: Notification,employeeIds:List<int>): void
+ CreateNotificationStatusFor(Employee,createdNotification: Notification): void
+ GetNotificationsPerEmployee(employeeId:int,platformPlatform): List<Notification>
+ GetNotificationsPerEmployee(employeeId:int,employeeId: int): bool
+ GetNotificationsSeenStatus(notificationId:int,employeeId: int): bool
+ GetAllPlatforms(): string[]
NotificationStorage
- conn: MySqlConnection (get;)
- isNotificationId:int (get;)
- employeeManager:EmployeeManager
+ CreateNotification(newNotification: Notification): void
+ CreateNotificationStatus(createdNotification: Notification,employeeIds:List<int>): void
+ CreateNotificationStatusFor(Employee,createdNotification: Notification): void
+ GetNotificationsPerEmployee(employeeId:int,platformPlatform): List<Notification>
+ GetNotificationTOSeenNotificationId:int,employeeId: int): bool
+ GetNotificationsSeenStatus(notificationId:int,employeeId: int): bool
+ GetAllPlatforms(): string[]
Notification
+ Notification(forUserId:int?,title:string,description:string,platform:Platform)
+ Notification(id:int,forUserId:int?,title:string,description:string,platform:Platform)
+ Notification(id:int,forUserId:int?,title:string,description:string,hasSeenSeen:bool)
+ Notification(id:int,forUserId:int?,title:string,description:string,hasSeenSeen:bool,platform:Platform)
+ Notification(uuid:int,isJobDescription:string?,title:string,description:string,platform:Platform)
+ Id:int (get;)
+ IsUserId:int(get; private set;int)
+ ToDepartment(get; private set;):string?
+ ToJobDescription(get; private set;):string?
+ IsUTypeId(get;):string?
+ Title:string (get; private set;):string
+ Description (get; private set): string
+ HasSeenSeen(get; private set): bool
+ Platform:(get; private set):Platform
Title MaxLenth is 25
Description MaxLenth is 100
<<Enumeration>>
Platform
Desktop,
Website
SettingsStorage
- employeeId:int
- conn: MySqlConnection
+ SettingsStorage(employeeId:int)
+ GetAllSettings(): List<Setting>
+ UpdateSettings(updateSettings:List<Setting>)
+ InsertDefaultSettings(): void
+ GetSettingTypes(): List<string>
<<Interface>>
IDepartmentManager
+ ViewAllEmployeeDepartments(): List<Department>
+ AddDepartment(name:string):bool
+ ViewAllDepartments(): List<Department>
+ EditDepartment(name:string, id:int):bool
+ DeleteDepartment(departmentId:int):bool
DepartmentManager
- DepartmentManager(departmentStorage: DepartmentStorage)
+ ViewAllEmployeeDepartments(): List<Department>
+ AddDepartment(name:string):bool
+ ViewAllDepartments(): List<Department>
+ EditDepartment(name:string, id:int):bool
+ DeleteDepartment(departmentId:int):bool
DepartmentStorage
-GET_ALL_EMPLOYEE_DEPARTMENTS: string
-GET_ALL_DEPARTMENTS: string
-REMOVE_DEPARTMENT: string
-ADD_DEPARTMENT: string
-UPDATE_DEPARTMENT:String
- DepartmentStorage()
+ ViewAllEmployeeDepartments(): List<Department>
+ AddDepartment(name:string):bool
+ ViewAllDepartments(): List<Department>
+ EditDepartment(name:string, id:int):bool
+ DeleteDepartment(departmentId:int):bool
IDepartmentStorage
+ ViewAllEmployeeDepartments(): List<Department>
+ AddDepartment(name:string):bool
+ GetDepartments(): List<Department>
+ EditDepartment(name:string, id:int):bool
+ DeleteDepartment(departmentId:int):bool