אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

ההנדסה למדעי הפקולטה
מידע מערכות להנדסת המחלקה

Faculty of Engineering Sciences
Dept. of Information Systems Engineering

Ben-Gurion University of the Negev

Grocery store

Grocery Store Management System

UML and additional diagrams

Submitters:

Nofar Carmel 315875542
Ido Rom 312239858
Lior Hassan 207227372
Shauli Genish 311242150
Noy Ezra 208347971
Shenhav Carmel 208421651
Itay Bouganim 305278384
Sahar Vaya 205583453

פירוט שינויים בתרשימים :

תרשימי רצף ושיתוף פעולה:

A&B – הוספת actors בתחילת התרשים, הוספת שלב הLogin בתחילת התרשים, תיקון המעברים והעברת מידע בין הactor לprinter.

D&C – הוספת שלב הLogin בתחילת התרשים, חיבור התחברות לממשק הספקים והמלאי דרך התפריט הראשי של הstore manager, תיקון מעברים.

- הוספנו תרשים רצף ושיתוף פעולה עבור פעולת הLogin שרלוונטית לכל התרשימים הנ"ל.

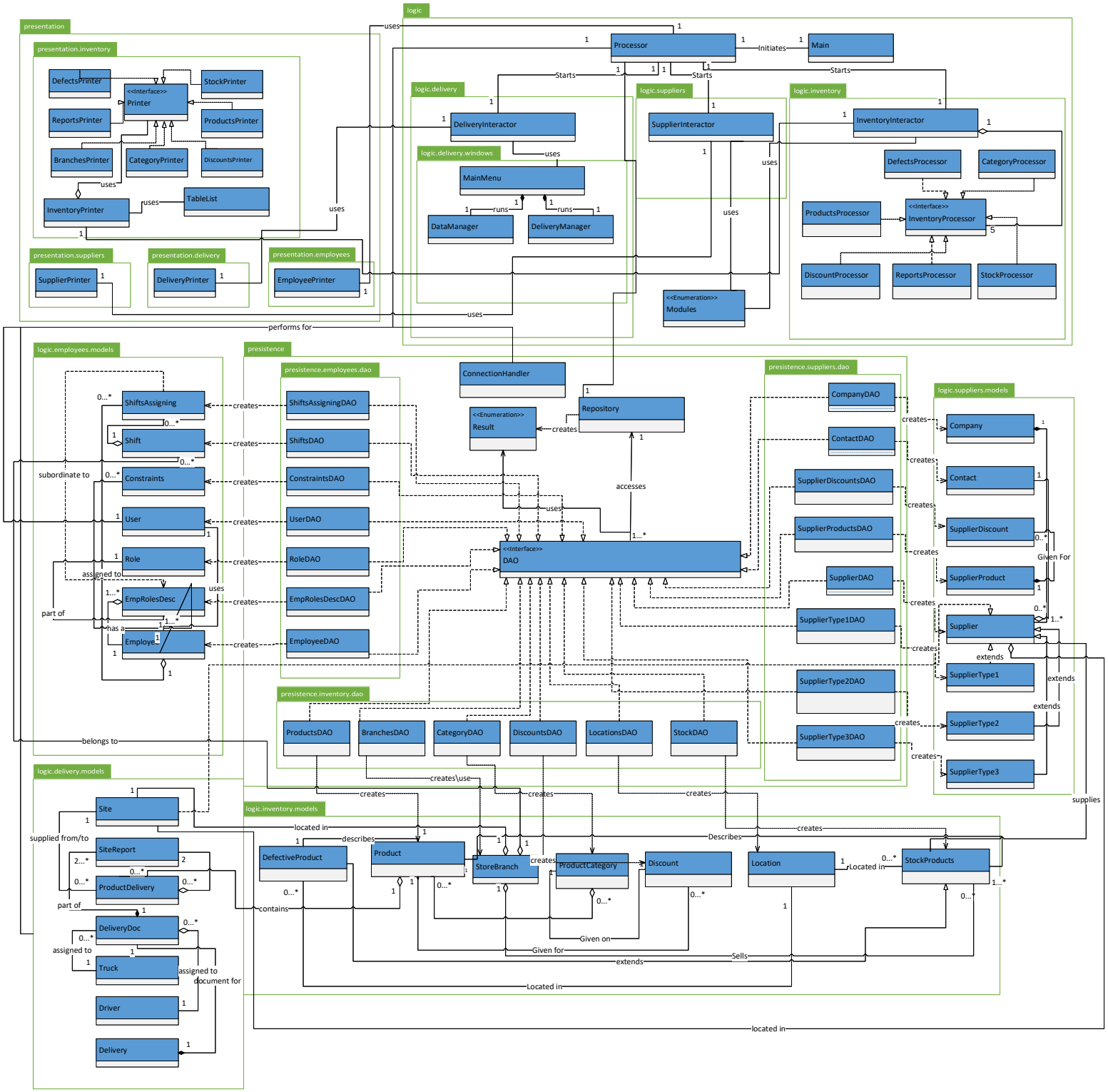
-ERD

ייצוג מסד הנתונים המשולב של ארבעת המודולים . האיחוד נעשה דרך טבלאות הספקים , הסניפים והמוצרים. הגדרת הקשרים בין היישויות הרלוונטיות .

– UML

עדכון המחלקות וחיבור הממשקים של כל המודולים.

UML Class Diagram



UML - Class specification

Deliveries.presentation::Printer

```
+instance – null: Printer
-reader: ConsoleReader
-scan: Scanner
-Printer(): ctor
+getInstance(): Printer
+closePrinter(): void
-flushScreen(String message, boolean error): void
+mainMenu(): int
+tablesManager(): int[]
+getID(): int
+getDriverShift(): String[]
+getSiteAddress(): String
+getLicensePlate(): String
+getProductDelivery(): String[]
+getSiteReport(): String[]
+printMessage(boolean error, String message): void
+getBranchData(): String[]
+getDeliveryData(): String[]
+getDeliveryDocData(): String[]
+getDriverData(): String[]
+getProductData(): String[]
+getProductDeliveryData(): int
+getSiteReportData(): String[]
+getSiteData(): String[]
+getSupplierData(): String[]
+getTruckData(): String[]
+PrintDelivery(): String[]
+waitForEnter(): void
-tryParseBool(String s): boolean
-tryParseDate(String s): boolean
-tryParseInt(String s): boolean
-tryParseTime(String s): boolean
-tryParseDouble(String s): boolean
```

Deliveries.logic.Windows::DeliveryManager

```
-driverShiftDAO – new DriverShiftDAO(): DriverShiftDAO
-supplierDAO – new SupplierDAO(): SupplierDAO
-deliveryDAO – new DeliveryDAO(): DeliveryDAO
-productDAO – new ProductDAO(): ProductDAO
-branchDAO – new BranchDAO(): BranchDAO
-productDeliveryDAO – new ProductDeliveryDAO(): ProductDeliveryDAO
-siteReportDAO – new SiteReportDAO(): SiteReportDAO
-truckDAO – new TruckDAO(): TruckDAO
-product: Product
-branch: Branch
-amount: int
-deliveries: List<Delivery>
-supplier: Supplier
+manageDelivery(): void
-addProduct(Delivery d, int option): boolean
-updatePrd(Delivery d): void
-upgradeTruck(Delivery d, double v): Delivery
-canDrive(Truck t, Driver driver): boolean
-updateWeight(int fromOrder, int toOrder, List<SiteReport> siteReports): void
-getTruckWeight(Delivery d): double
-createDelivery(): Delivery
-contains(Delivery d, Site site): boolean
```

Deliveries.presistence::DeliveryDAO

```
-connectionHandler – new ConnectionHandler(): ConnectionHandler
-deliveryDocDAO – new DeliveryDocDAO(): DeliveryDocDAO
-tablename – "deliveries": String
-deliveryDocsTable – "deliveryDocs": String
+findID(int ID): Delivery
+insert(Delivery d): void
+update(Delivery d): void
+delete(Delivery d): void
+findAvailableDeliveries(): List<Delivery>
+getNewID(): int
+isAvailable(int driverId, String shiftType, Date date): boolean
```

Deliveries.presistence::SupplierDAO

```
-connectionHandler – new ConnectionHandler(): ConnectionHandler
-prodDAO – new ProductDAO(): ProductDAO
-tablename – "suppliers": String
-siteTablename – "sites": String
+findID(int ID): Supplier
+findAddrss(String addrss): Supplier
+insert(Supplier s): void
+update(Supplier s): void
+delete(Supplier s): void
+findByAreaAndProduct(String area, int id): Supplier
+getIndependent(String area, int id): boolean
```

Deliveries.presistence::DeliveryDocDAO

```
-connectionHandler – new ConnectionHandler(): ConnectionHandler
-driverDAO – new DriverDAO(): DriverDAO
-truckDAO – new TruckDAO(): TruckDAO
-siteReportDAO – new SiteReportDAO(): SiteReportDAO
-tablename – "deliveryDocs": String
+findID(int ID): DeliveryDoc
+insert(DeliveryDoc dd): void
+update(DeliveryDoc dd): void
+delete(DeliveryDoc dd): void
```

Deliveries.presistence::TruckDAO

```
-connectionHandler – new ConnectionHandler(): ConnectionHandler
-tablename – "trucks": String
-deliveriesTable – "deliveries": String
-deliverieDocTable – "deliveryDocs": String
+findlicense(String lp): Truck
+insert(Truck t): void
+update(Truck t): void
+delete(Truck t): void
+findAvailable(Date startDate, String shift, double v): Truck
```

Deliveries.presistence::ProductDeliveryDAO

```
-connectionHandler – new ConnectionHandler(): ConnectionHandler
-siteDAO – new SiteDAO(): SiteDAO
-branchDAO – new BranchDAO(): BranchDAO
-supplierDAO – new SupplierDAO(): SupplierDAO
-productDAO – new ProductDAO(): ProductDAO
-tablename – "productsDeliveries": String
+findID(int deliveryID, String suppAddrs, String brenchAddrs, int prodID): ProductDelivery
+insert(ProductDelivery pd): void
+update(ProductDelivery pd): void
+delete(ProductDelivery pd): void
+findSite(Site s): List<ProductDelivery>
```

Deliveries.presistence::BranchDAO

```
-connectionHandler – new ConnectionHandler(): ConnectionHandler
-tablename – "branches": String
-siteTablename – "sites": String
+findID(int ID): Branch
+findAddrss(String addrss): Branch
+insert(Branch branch): void
+update(Branch branch): void
+delete(Branch branch): void
```

Deliveries.presistence::SiteDAO

```
-branchDAO – new BranchDAO(): BranchDAO
-supplierDAO – new SupplierDAO(): SupplierDAO
+findAddrs(String addrss): Site
+insert(Site s): void
+update(Site s): void
+delete(Site s): void
```

Deliveries.logic.Windows::DataManager

```
-show – true: boolean
+ManagerWIndow(): void
-manageBranch(int mode): void
-manageDelivery(int mode): void
-manageDeliveryDoc(int mode): void
-manageDriver(int mode): void
-manageDriverShift(int mode): void
-manageProduct(int mode): void
-manageProductDelivery(int mode): void
-manageSiteReport(int mode): void
-manageSite(int mode): void
-manageSupplier(int mode): void
-manageTruck(int mode): void
```

Deliveries.presistence::DriverShiftDAO

```
-connectionHandler – new ConnectionHandler(): ConnectionHandler
-driverDAO – new DriverDAO(): DriverDAO
-deliveryDAO – new DeliveryDAO(): DeliveryDAO
-tablename – "drivers_shifts": String
+findID(int driverID, Date shiftDate, String shiftType): DriverShift
+insert(DriverShift ds): void
+delete(DriverShift ds): void
+getAvailable(): List<DriverShift>
```

Deliveries.presistence::SiteReportDAO

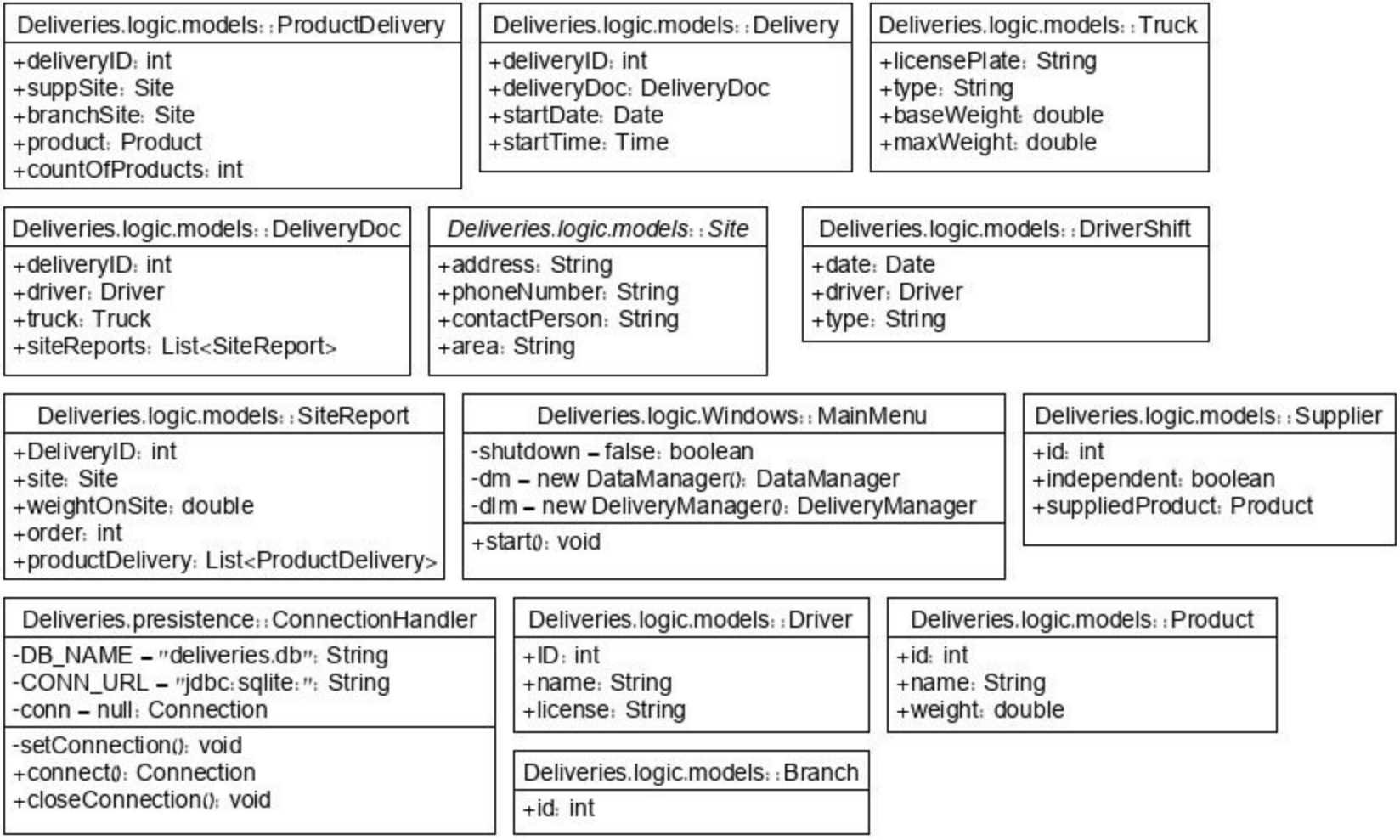
```
-connectionHandler – new ConnectionHandler(): ConnectionHandler
-siteDAO – new SiteDAO(): SiteDAO
-productDeliveryDAO – new ProductDeliveryDAO(): ProductDeliveryDAO
-tablename – "site_report": String
+findID(int deliveryID, String siteAddrss): SiteReport
+insert(SiteReport s): void
+update(SiteReport s): void
+delete(SiteReport s): void
+findByDelivery(int deliveryID): List<SiteReport>
```

Deliveries.presistence::ProductDAO

```
-connectionHandler – new ConnectionHandler(): ConnectionHandler
-tablename – "products": String
+findID(int ID): Product
+insert(Product p): void
+update(Product p): void
+delete(Product p): void
```

Deliveries.presistence::DriverDAO

```
-connectionHandler – new ConnectionHandler(): ConnectionHandler
-tablename – "drivers": String
+findID(int ID): Driver
+insert(Driver d): void
+update(Driver d): void
+delete(Driver d): void
```


Deliveries.logic.models::ProductDelivery
+deliveryID: int
+suppSite: Site
+branchSite: Site
+product: Product
+countOfProducts: int
Deliveries.logic.models::Delivery
+deliveryID: int
+deliveryDoc: DeliveryDoc
+startDate: Date
+startTime: Time
Deliveries.logic.models::Truck
+licensePlate: String
+type: String
+baseWeight: double
+maxWeight: double
Deliveries.logic.models::DeliveryDoc
+deliveryID: int
+driver: Driver
+truck: Truck
+siteReports: List<SiteReport>
Deliveries.logic.models::Site
+address: String
+phoneNumber: String
+contactPerson: String
+area: String
Deliveries.logic.models::DriverShift
+date: Date
+driver: Driver
+type: String
Deliveries.logic.models::SiteReport
+DeliveryID: int
+site: Site
+weightOnSite: double
+order: int
+productDelivery: List<ProductDelivery>
Deliveries.logic.Windows::MainMenu
-shutdown = false: boolean
-dm = new DataManager(): DataManager
-dlm = new DeliveryManager(): DeliveryManager
+start(): void
Deliveries.logic.models::Supplier
+id: int
+independent: boolean
+suppliedProduct: Product
Deliveries.presistence::ConnectionHandler
-DB_NAME = "deliveries.db": String
-CONN_URL = "jdbc:sqlite:": String
-conn = null: Connection
-setConnection(): void
+connect(): Connection
+closeConnection(): void
Deliveries.logic.models::Driver
+ID: int
+name: String
+license: String
Deliveries.logic.models::Product
+id: int
+name: String
+weight: double
Deliveries.logic.models::Branch
+id: int

```
Employees.presentation::Printer
-scan: Scanner
#reader: ConsoleReader
+Printer(): ctor
+closePrinter(): void
-flushScreen(String message, boolean error): void
+loginMenu(): String[]
+mainMenu(boolean master, boolean HR, boolean TM): int
+masterManage(): String
+addUserPrompt(): String[]
+printDeleteUser(): String
+printAddUser(): String[]
+editMenu(): String[]
+printProfile(String[] details): void
+printMessage(boolean error, String message): void
+printMyScheduleConstraints(String[] constraints): void
+printEditScheduleConstraints(boolean b): String[]
+printShifts(String[] shifts): void
+printScheduleConstraints(String[] constraints): void
+printSucDel(Result res): void
+printSucInsert(Result res): void
+printUpdateShifts(boolean b): String
+printDeleteShift(): String
+printInsShift(): String[]
+printSucDelShift(Result res): void
+printSucInsertShift(Result res): void
+printMyShiftAss(String[] myShiftsAssigns): void
+manageShiftsAssignPrompt(String[] myShiftsAssigns): int
+editShiftsAssign(): int[]
+castIntErr(String s): int
+printSucEditShiftAssign(Result res): boolean
+manageEmplyeesPrompt(String[] employees): int
+deleteEmployee(int[] ids): int
-getEmpValid(int[] ids, boolean newEmp): int
+updateEmployee(int[] ids, boolean isNew): String[]
+noRole(): void
+noCons(): void
+mangNoCons(): void
+printERR(): void
+addJobEmp(): String[]
+noShiftNum(): void
```

```
Employees.presistence.dao::EmployeeDAO
~df = new SimpleDateFormat("dd/MM/yyyy"): SimpleDateFormat
-conn = null: Connection
+update(Employee emp, Integer key): Result
+delete(Employee emp): Result
+insert(Employee emp): Result
+findByKey(Employee emp): Employee
+setConnection(Connection conn): void
+findByVal(Object id): LinkedList<Employee>
```

```
Employees.logic.models::EmpRolesDesc
+key: Pair<Integer, Integer>
+roleDescription: String
```

```
Employees.logic.models::Employee
+key: int
+HR: boolean
+firstName: String
+lastName: String
+salary: int
+firstEmployed: Date
+employmentCond: String
```

```
presistence::Repository
+DB_MASTER = "master": String
-daoMap: Map<DAO_TYPE, DAO>
-DB_NAME = "users.db": String
-CONN_URL = "jdbc:sqlite:": String
-conn = null: Connection
+Repository(): ctor
-setConnection(): void
+connect(): void
+closeConnection(): void
-createDB(): void
+getDAO(DAO_TYPE daoType): DAO
```

```
Employees.presistence.dao::UserDAO
-conn = null: Connection
+update(User user, String key): Result
+delete(User user): Result
+insert(User user): Result
+findByKey(User user): User
+setConnection(Connection conn): void
+findByVal(Object id): LinkedList<User>
```

```
Employees.logic.models::User
+master: boolean
+HR: boolean
+TM: boolean
+username: String
+password: String
+id: int
+firstName: String
+lastName: String
```

```
Employees.presistence.dao::EmpRolesDescDAO
-conn = null: Connection
+update(EmpRolesDesc empRolesDesc, Pair<Integer, Integer> key): Result
+delete(EmpRolesDesc empRolesDesc): Result
+insert(EmpRolesDesc empRolesDesc): Result
+findByKey(EmpRolesDesc empRolesDesc): EmpRolesDesc
+setConnection(Connection conn): void
+findByVal(Object id): LinkedList<EmpRolesDesc>
+findRules(int empID): LinkedList<Integer>
```

```
Employees.logic.models::Constraints
+empID: int
+shiftTime: String
+date: Date
```

```
Employees.logic.models::Shift
+managerID: int
+dateTime: Date
+shiftID: int
+shiftTime: String
```

```
Employees.logic::Processor
-MAX_ID_LENGTH = 9: int
~df = new SimpleDateFormat("dd/MM/yyyy"): SimpleDateFormat
-shutdown: boolean
-connected = null: User
-repo: Repository
-printer: Printer
+Processor(Printer printer): ctor
+start(): void
+checkUserValidity(String username, String pwd): User
-validateID(String id): boolean
-processLogin(): boolean
-setIfHR(EmpRolesDesc empTested): void
-setIfTM(EmpRolesDesc empTested): void
-processLogout(): void
-processMainMenu(): boolean
-processDeleteUser(): void
-processViewProfile(boolean master): void
-processEditMenu(boolean master): void
-processAddUser(): void
-getConstraints(int id): void
-getAllConstraints(): void
-getShifts(): void
-editConstraints(): void
-updateShifts(): void
-processManageShiftsAssign(): void
-processEmployees(): void
+hasRole(ShiftAssigning assign): boolean
+hasConstraint(ShiftAssigning assign): boolean
+managerHasConstraint(Shift shift): boolean
+viewShiftsAss(int id): void
+AddJobEmp(): void
-manageDeliveries(): void
+getFutureDriversShifts(): LinkedList<ShiftAssigning>
```

```
Employees.presistence.dao::ConstraintsDAO
~df = new SimpleDateFormat("dd/MM/yyyy"): SimpleDateFormat
~dfH = new SimpleDateFormat("hh:mm:ss"): SimpleDateFormat
-conn = null: Connection
+update(Constraints constraints, String key[]): Result
+delete(Constraints constraints): Result
+insert(Constraints constraints): Result
+findByKey(Constraints constraints): Constraints
+setConnection(Connection conn): void
+findByVal(Object id): LinkedList<Constraints>
```

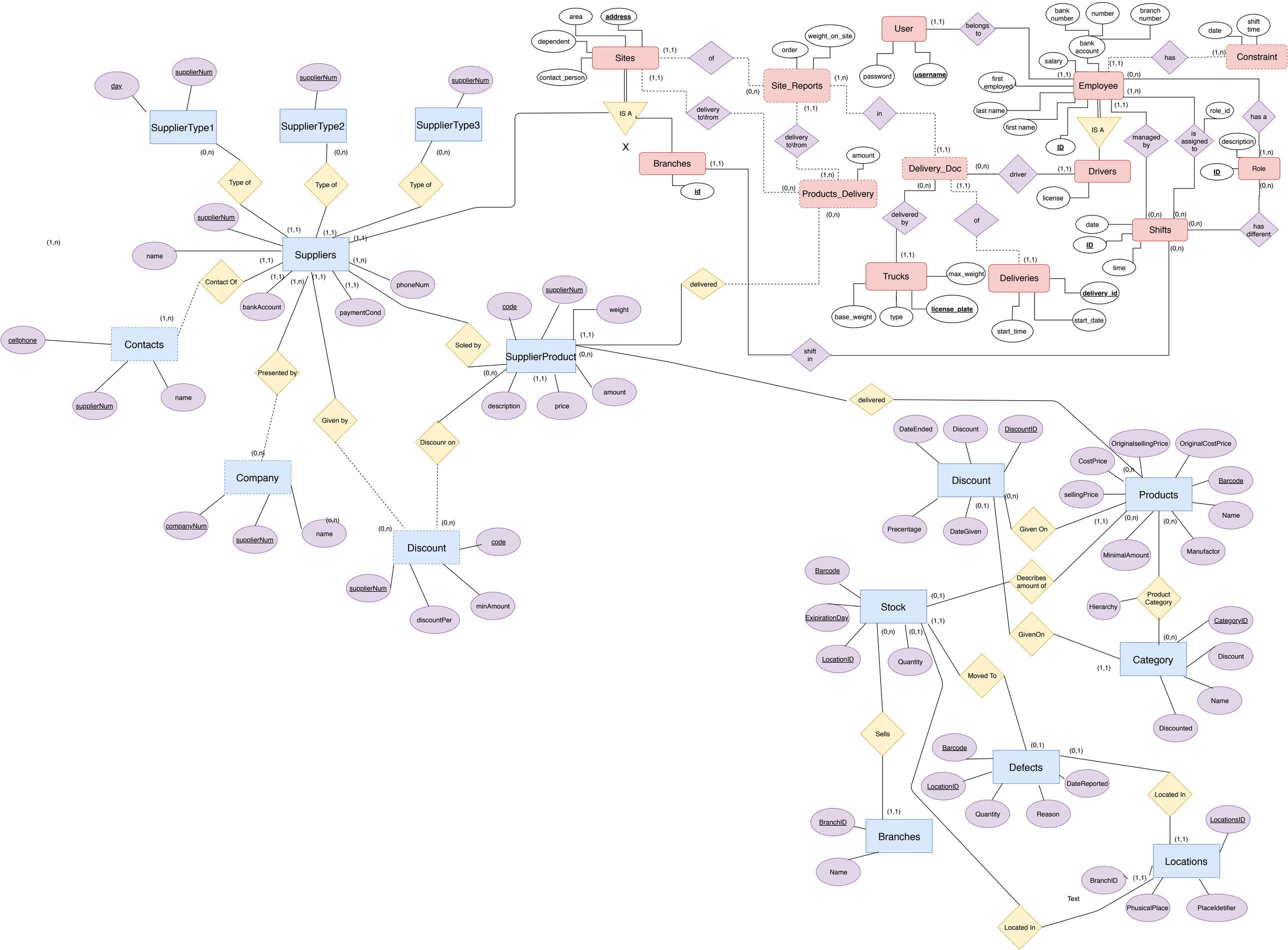
```
Employees.logic.models::ShiftAssigning
+shiftID: int
+roleID: int
+empID: int
+roleDesc: String
```

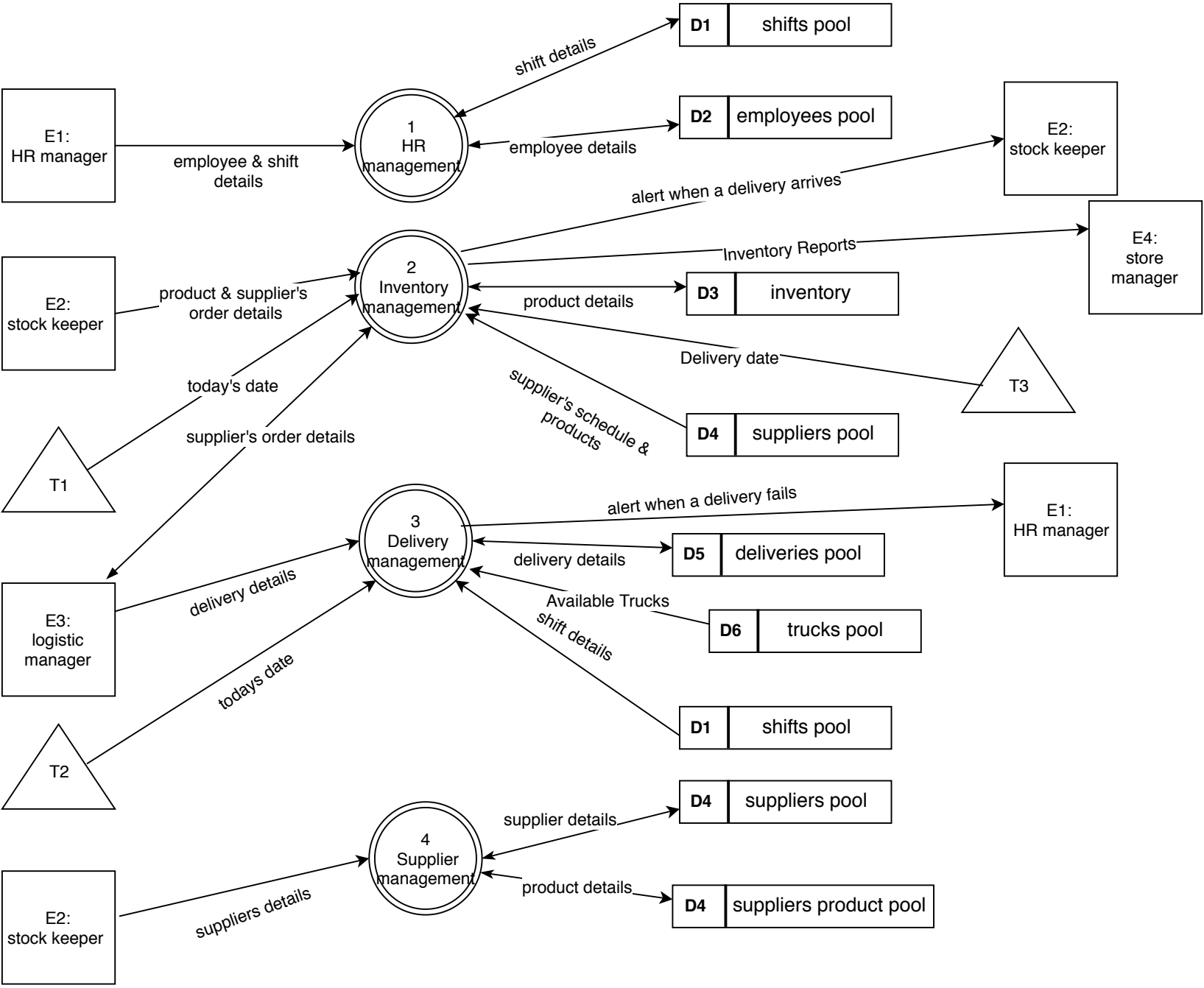
```
Employees.presistence.dao::ShiftsDAO
~df = new SimpleDateFormat("dd/MM/yyyy"): SimpleDateFormat
-conn = null: Connection
+update(Shift shift, Integer key): Result
+delete(Shift shift): Result
+insert(Shift shift): Result
+findByKey(Shift shift): Shift
+setConnection(Connection conn): void
+findByVal(Object id): LinkedList<Shift>
```

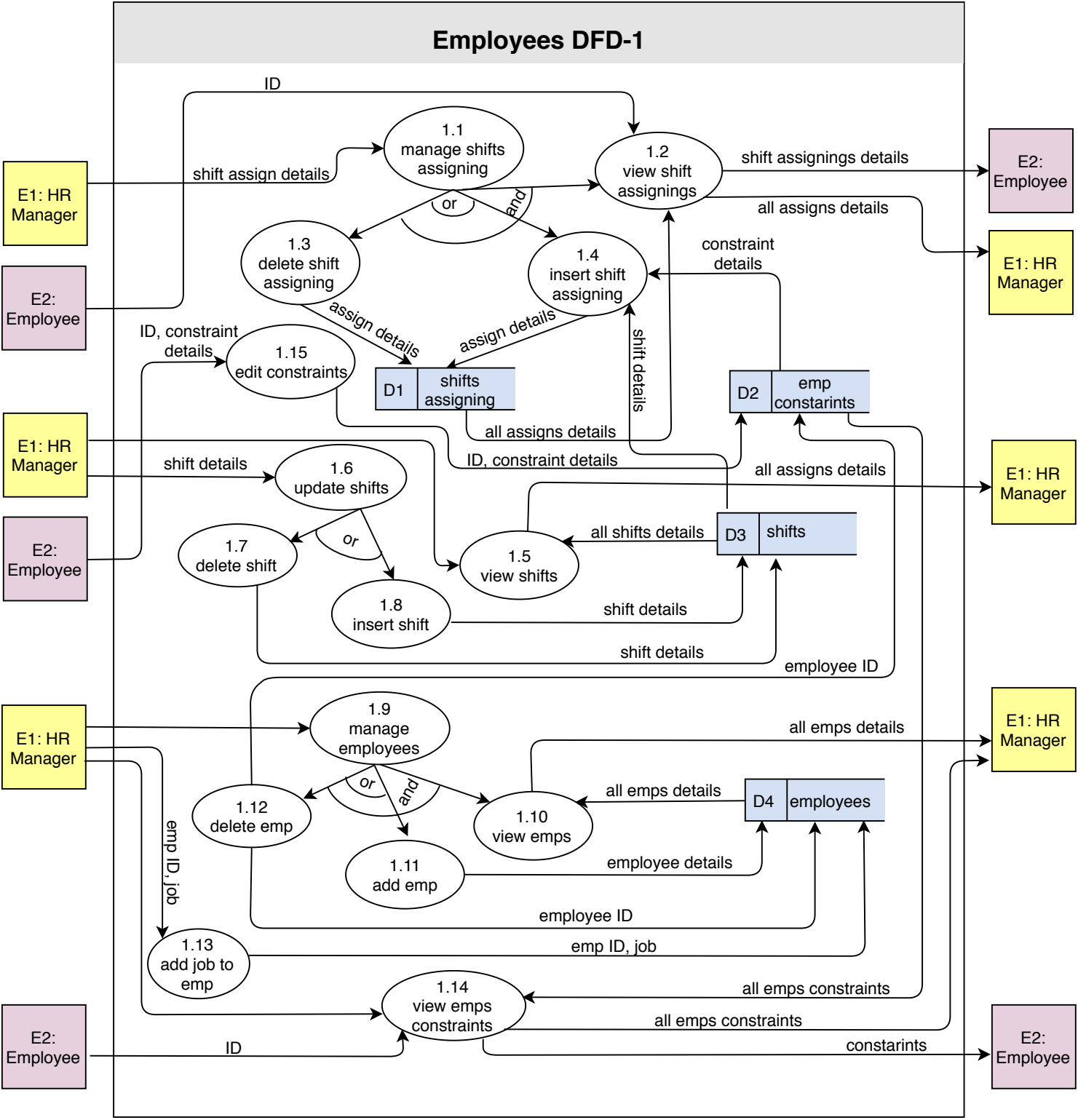
```
Employees.presistence.dao::ShiftsAssigningDAO
-conn = null: Connection
+update(ShiftAssigning shiftAssigning, Integer empID): Result
+delete(ShiftAssigning shiftAssigning): Result
+insert(ShiftAssigning shiftAssigning): Result
+findByKey(ShiftAssigning shiftAssigning): ShiftAssigning
+setConnection(Connection conn): void
+findByVal(Object id): LinkedList<ShiftAssigning>
```

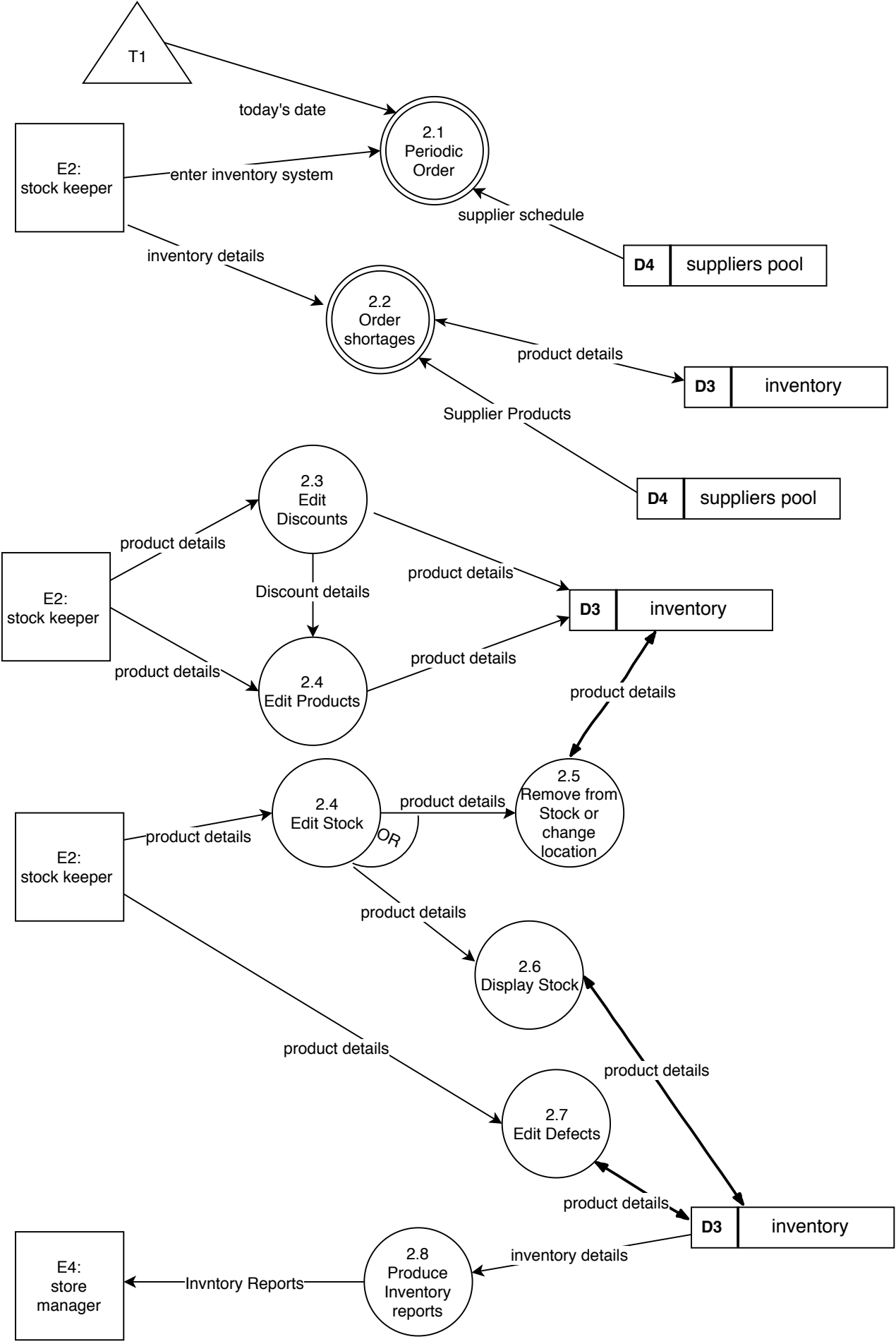
```
Contact
+name: String
+phoneNum: String
+supplierNum: int
+Contact(int supplierNum, String phoneNum, String name): ctor
+Contact(int supplierNum): ctor

Company
-companyNum: int
+supplierNum: int
+name: String
+Company(int companyNum, int supplierNum, String name): ctor

Discount
+supplierNum: int
+code: String
+minAmount: int
+discount: int
+Discount(int supplierNum, String code, int minAmount, int discount): ctor
+Discount(int supplierNum): ctor

Product
+supplierNum: int
+price: int
+amount: int
+desc: String
+code: String
+Product(String desc, String code, int supplierNum, int price, int amount): ctor
+Product(int supplierNum): ctor

Supplier
+supplierNum: int
+name: String
+bankAccount: String
+paymentCond: String
+phoneNum: String
+Supplier(int supplierNum, String name, String bankAccount, String paymentCond, String phoneNum): ctor
+Supplier(int supplierNum): ctor

SupplierType1
+days: LinkedList<String>
+SupplierType1(int supplierNum, String name, String bankAccount, String paymentCond, String phoneNum, LinkedList<String> days): ctor
+SupplierType1(int supplierNum): ctor

SupplierType2
+SupplierType2(int supplierNum, String name, String bankAccount, String paymentCond, String phoneNum): ctor
+SupplierType2(int supplierNum): ctor

SupplierType3
+SupplierType3(int supplierNum, String name, String bankAccount, String paymentCond, String phoneNum): ctor
+SupplierType3(int supplierNum): ctor

SupplierInteractor
-shutdown: boolean
-connected = null: Supplier
-repo: Repository
-printer: SupplierPrinter
+SupplierInteractor(SupplierPrinter printer, Repository repo): ctor
+start(): void
+checkSupplierValidity(int supplierNum): Supplier
-processLogin(): boolean
-processMainMenu(): boolean
-processEditMenu(): void
-processViewProfile(): void
-processViewProduct(): void
-processAddProduct(): void
-processViewDiscount(): void
-processAddDiscount(): void
-processViewContact(): void
-processAddContact(): void
-processDeleteProduct(): void
-processDeleteDiscount(): void
-processDeleteContact(): void
-processViewPresentingCompany(): void
-processAddPresentingCompany(): void
-processDeleteCompany(): void

CategoryProcessor
-repo: Repository
-prompter: Prompter
+CategoryProcessor(Prompter prompter, Repository repo): ctor
+process(): void
-processPrintAllCategories(): void
-processDisplayCategoryDetails(): void
-processDeleteCategory(): void
-processRenameCategory(): void
-processAddNewCategory(): void

DiscountProcessor
-repo: Repository
-prompter: Prompter
+DiscountProcessor(Prompter prompter, Repository repo): ctor
+process(): void
-processDisplayDiscountedCategories(String discounter): void
-processViewDiscounts(String discounter): void
-processCancelDiscount(String discounter): void
-reDiscount(Discount discount): void
-processAddNewDiscount(String discounter): boolean
-processPrintAllDiscounts(String discounter): void

DefectsProcessor
-repo: Repository
-prompter: Prompter
+activeBranch: StoreBranch
+currentDate: Date
+DefectsProcessor(Prompter prompter, Repository repo): ctor
+process(): void
-processReportDefectiveProdcut(): void
-processRemoveFromDefective(): void
-processViewAllDefective(): void

InventoryInteractor
-instance = null: InventoryInteractor
-processors: Map<Modules, Processor>
-shutdown: boolean
-activeStoreBranch = null: StoreBranch
-currentDate: Date
-repo: Repository
-prompter: Prompter
-InventoryInteractor(Prompter prompt, Repository repo): ctor
+getInstance(Repository repo): InventoryInteractor
+start(): void
-moveExpiredToDefects(Map.Entry<List<StockProducts>, List<StockProducts>> expired): void
+logToBranch(): boolean
-processOpenNewBranch(): void
-processCloseBranch(): void
-processViewChain(): void
-processMainMenu(): boolean

«interface» Processor
~process(): void

ProductsProcessor
-repo: Repository
-prompter: Prompter
+ProductsProcessor(Prompter prompter, Repository repo): ctor
+process(): void
-processViewProductByCategory(): void
-processDisplayDetails(): void
-processDeleteProduct(): boolean
-processAddNewProduct(): boolean
-processUpdateProduct(): boolean
-processEditProductCategories(String mainCategory, List<Map.Entry<Integer, String>> subCategories): void
-processGetProductDetails(boolean wait, String message): Product
-reDiscount(Product p): void
-processPrintAllProducts(): void

ReportsProcessor
-repo: Repository
-prompter: Prompter
-activeBranch: StoreBranch
+currentDate: Date
+ReportsProcessor(Prompter prompter, Repository repo): ctor
+process(): void
-processInventoryReport(): void
-processDefectedReport(): void
-processNotifications(): void

StockProcessor
+activeBranch: StoreBranch
-repo: Repository
-prompter: Prompter
+StockProcessor(Prompter prompter, Repository repo): ctor
+process(): void
-processMoveStock(): void
-processRemoveFromStock(): void
-processAddToStock(): void
-addToStock(Product p): void
-moveStock(int barcode): void
-processDisplayStockDetails(): void
-processDisplayAllStock(): void
-processDisplayAllStockByCategory(): void
-checkLegalLocationString(StockProducts p, Location location): boolean

BranchesPrinter
+printMenu(): int
+chooseBranch(boolean warn): int
+promptNewBranchName(): String
+printChainBranches(List<StoreBranch> branches): void

CategoryPrinter
+printMenu(): int
+promptViewByCategory(): int
+categoryIDPrompt(): int
+multipleCategoryIDPrompt(): List<Integer>
+categoryPrompt(String additional): String
+printCategoryDetails(String cat_name, int cat_id, String additional): void
+promptCategoryAdded(String catName): void
+printProductCategories(List<ProductCategory> categories, int barcode, boolean wait, String message, boolean error, boolean newScreen): void
+printCategories(List<ProductCategory> categories, boolean wait, String message, boolean error, boolean newScreen): void

DefectsPrinter
+printMenu(): int
+printAllStockByLocationAndID(List<Map.Entry<Integer, StockProducts>> inv, boolean wait): void
+printAllDefectiveByLocationAndID(List<Map.Entry<Integer, DefectiveProduct>> inv, boolean wait): void
+printAskIdentifier(int limit): int

DiscountsPrinter
+printMenu(): int
+printMenu(String discounter): int
+printManageDiscountsSelect(): int
+byProductCategoryDiscountPrompt(String prod_descriptor, String title): int
+addDiscountDetailsPrompt(): Double
+promptDiscountToRemove(List<Discount> discounts): int
+activeDiscountPrompt(): boolean
+printDiscounts(List<Discount> discounts, boolean wait, String discounter, String message, boolean error): void
+printDiscountsWithNames(List<Map.Entry<String, Discount>> discounts, String discounter, boolean wait, String message, boolean error): void
+printDiscountedCategories(List<ProductCategory> categories, boolean wait, String message, boolean error, boolean newScreen): void

«interface» Printer
~printMenu(): int

ProductsPrinter
+printMenu(): int
+addProductDetailsPrompt(): String[]
+addProductMainCategory(): String
+checkAddProductsSubCategory(): boolean
+addProductSubCategories(int hierarchy): String
+promptCategoryNotExist(String catName): boolean
+printUpdateSelectProduct(): Map.Entry<Integer, String>
+printProducts(List<Product> products, boolean wait, String message, boolean error, boolean newScreen): void

ReportsPrinter
+printMenu(): int
+chooseCategoryAll(): int
+reportDefects(List<DefectiveProduct> defects, List<DefectiveProduct> defectsLocations): void
+printInventory(ProductCategory category, List<StockProducts> inv, boolean wait): void
+printCategoryNotExist(int id, boolean wait): void

TableList
-BLINE = { "-", "" }: String[]
-CROSSING = { "+", "" }: String[]
-VERTICAL_TSEP = { "|", "" }: String[]
-VERTICAL_BSEP = { "|", "" }: String[]
-TOP_BOTTOM = "=": String
-TLINE = "": String
-CORNER_TL = "": String
-CORNER_TR = "": String
-CORNER_BL = "": String
-CORNER_BR = "": String
-CROSSING_L = "": String
-CROSSING_R = "": String
-CROSSING_T = "": String
-CROSSING_B = "": String
-descriptions: String[]
-table: ArrayList<String[]>
-tableSizes: int[]
-rows: int
-finder: int
-filter: String
-ucode: boolean
-comparator: Comparator<String[]>
-spacing: int
-aligns[]: EnumAlignment
+TableList(String... descriptions): ctor
+TableList(int columns, String... descriptions): ctor
-updateSizes(String[] elements): void
+compareWithC(Comparator<String[]> c): TableList
+sortBy(int column): TableList
+align(int column, EnumAlignment align): TableList
+withSpacing(int spacing): TableList
+addRow(String... elements): TableList
+filterBy(int pari0, String pattern): TableList
+withUnicode(boolean ucodeEnabled): TableList
+print(): void
-gc(String[] src): String

StockPrinter
+printMenu(): int
+promptStockDisplayByLocation(): boolean
+promptExpirationDate(): Date
+promptQuantity(String adj): int
+promptDeletion(): boolean
+promptLocation(StoreBranch activeBranch): Location
+movePrompt(StoreBranch activeBranch, String adj): Location
+printStockAmounts(List<StockProducts> products): void
+printAllStock(List<StockProducts> inv, boolean wait): void
+printAllStockByLocation(List<StockProducts> inv, boolean wait): void

Prompter
+ILLEGAL = -1: int
+RETURN = 0: int
-initiated = false: boolean
-MAX_BARCODE_LENGTH = 9: int
#reader: ConsoleReader
#scan = new Scanner(System.in): Scanner
+printers = new HashMap() {
    {
        put(Modules.BRANCHES, new BranchesPrinter());
        put(Modules.PRODUCTS, new ProductsPrinter());
        put(Modules.DISCOUNTS, new DiscountsPrinter());
        put(Modules.CATEGORIES, new CategoryPrinter());
        put(Modules.STOCK, new StockPrinter());
        put(Modules.REPORTS, new ReportsPrinter());
        put(Modules.DEFECTS, new DefectsPrinter());
    }
}: Map<Modules, Printer>
+Prompter(): ctor
+closePrompter(): void
-flushScreen(String message, boolean error): void
#menuSelection(int lowerBound, int upperBound): int
+mainMenu(StoreBranch branch): int
+printMessage(Boolean error, String message): void
+barcodePrompt(): int
+nameSearchPrompt(): String
+byBarcodeNamePrompt(String manegement, String title): int
+deletePrompt(): Boolean
+promptNotifications(List<StockProducts> expired, List<StockProducts> critical_amount, Date currentDate): void

SupplierPrinter
-reader: Scanner
+SupplierPrinter(): ctor
+closePrinter(): void
-flushScreen(String message, boolean error): void
+printMessage(boolean error, String message): void
+loginMenu(): int
+loginDetails(): String
+Deleter(): String
+RegisterChoice(): int
+Register(): String[]
+mainMenu(): int
+editMenu(): String[]
+printProfile(String[] details): void
+printProfileType1(String[] details, LinkedList<String> days): void
+printProduct(List<Product> products): void
+addProduct(): String[]
+printDiscountList(List<Discount> discounts): void
+addDiscount(): String[]
+printContacts(List<Contact> contacts): void
+addContact(): String[]
+DeleteProduct(): String
+DeleteDiscount(): String
+DeleteContact(): String
+addCompany(): String[]
+printCompanies(List<Company> companies): void
+DeleteCompany(): String
+RegisterDays(): LinkedList<String>
-checkDayValidity(String day): boolean

BranchesDAO
+conn = null: Connection
+update(StoreBranch branch, Integer key, Integer dummy, Integer dummy2): Result
+delete(StoreBranch branch): Result
+insert(StoreBranch branch): Result
+findByKey(StoreBranch branch): StoreBranch
+findByName(StoreBranch branch): StoreBranch
+findAll(): List<StoreBranch>

CategoryDAO
+conn = null: Connection
+update(ProductCategory category, Integer key, String newName, Integer key3): Result
+delete(ProductCategory productCategory): Result
+deleteAllByProduct(ProductCategory productCategory): Result
+insert(ProductCategory productCategory): Result
+findByKey(ProductCategory productCategory): ProductCategory
+findCategoryByKey(Integer key): ProductCategory
+findByName(ProductCategory productCategory): ProductCategory
+insertCategory(ProductCategory category): Result
+findAllByProduct(Product prod): List<ProductCategory>
+findAllDiscounted(String discounter): List<ProductCategory>
+findAll(): List<ProductCategory>

DefectiveDAO
+conn = null: Connection
+update(DefectiveProduct defectiveProduct, Integer key1, Integer key2, Integer key3): Result
+delete(DefectiveProduct defect): Result
+insert(DefectiveProduct defect): Result
+findByKey(DefectiveProduct defectiveProduct): DefectiveProduct
+findAll(StoreBranch activeBranch, Integer barcode): List<DefectiveProduct>
+findAllTotal(StoreBranch activeBranch): List<DefectiveProduct>

LocationsDAO
+conn = null: Connection
+update(Location location, Integer key1, Integer key2, Integer key3): Result
+delete(Location location): Result
+insert(Location location): Result
+findByKey(Location location): Location
+findLocation(Location location): Location

DiscountsDAO
+conn = null: Connection
+update(Discount d, Integer key, Integer dummy, Integer dummy2): Result
+delete(Discount d): Result
+insert(Discount d): Result
+findByKey(Discount d): Discount
+findByBarcode(Discount d, boolean checkEnded, boolean checkDiscounter): List<Discount>
+findAllByDiscounter(String discounter, boolean onlyActive): List<Discount>
+findAllLike(String like, String discounter, boolean checkEnded): List<Map.Entry<String, Discount>>
-createDiscountFromResult(ResultSet rs): Discount

ProductsDAO
+conn = null: Connection
+update(Product prod, Integer key, Integer dummy, Integer dummy2): Result
+delete(Product prod): Result
+insert(Product prod): Result
+findByKey(Product prod): Product
+findAll(): List<Product>
+findAllLike(String like): List<Product>
+findAllByCategory(String name, int categoryID): List<Product>
-createProductFromResult(ResultSet rs): Product

StockDAO
+conn = null: Connection
+update(StockProducts stock, Integer key1, Date key2, Integer key3): Result
+delete(StockProducts stock): Result
+insert(StockProducts stock): Result
+findByKey(StockProducts stockProducts): StockProducts
+findTotalAmount(StockProducts stock, Boolean warehouse): Result
+findAmountByLocations(StoreBranch activeBranch, int barcode): List<StockProducts>
+findExpired(StoreBranch activeBranch): Map.Entry<List<StockProducts>, List<StockProducts>>
+findCriticalAmount(StoreBranch activeBranch): List<StockProducts>
+findAllInventory(StoreBranch activeBranch, ProductCategory category): List<StockProducts>
+findAllInventoryByBarcode(StoreBranch activeBranch, int barcode): List<StockProducts>
+findAllByLocation(StoreBranch activeBranch): List<StockProducts>
+findByBarcodeAndLocation(StoreBranch activeBranch, int barcode): List<StockProducts>
-createStockFromResult(ResultSet rs, boolean location): StockProducts

«interface» DAO
~update(T t, K1 key1, K2 key2, K3 key3): Result
~delete(T t): Result
~insert(T t): Result
~findByKey(T t): T
~setConnection(Connection conn): void

Repository
+inventory_daoMap: Map<Modules, DAO>
+supplier_daoMap: Map<Modules, DAO>
-INV_DB_NAME = "inventory.db": String
-SUP_DB_NAME = "supplier.db": String
+CONN_URL = "jdbc:sqlite:": String
-DRIVER = "org.sqlite.JDBC": String
-inv_conn = null: Connection
-sup_conn = null: Connection
+Repository(): ctor
+connect(): void
+closeConnection(): void
-createInventoryDB(): void
-createSupplierDB(): void
-createTriggers(): void
```

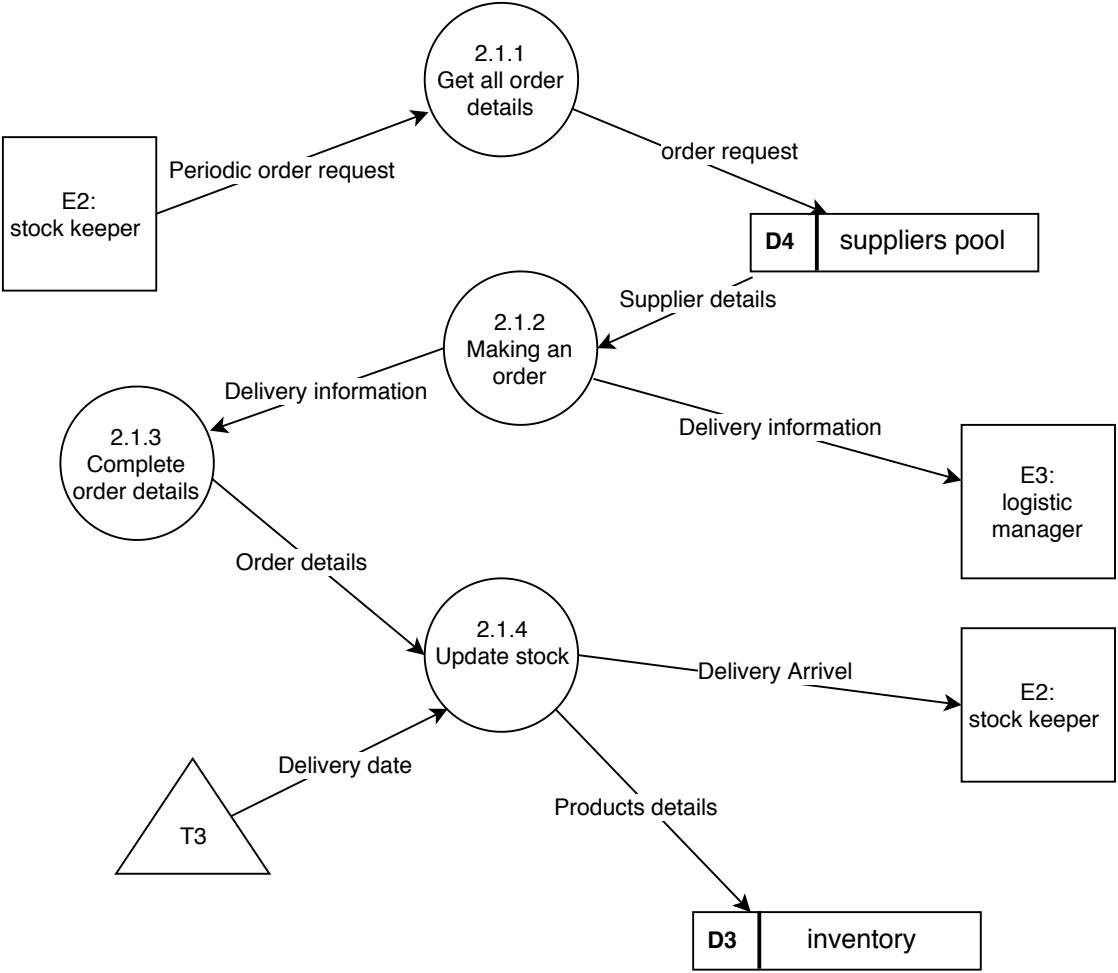
ERD Diagram:

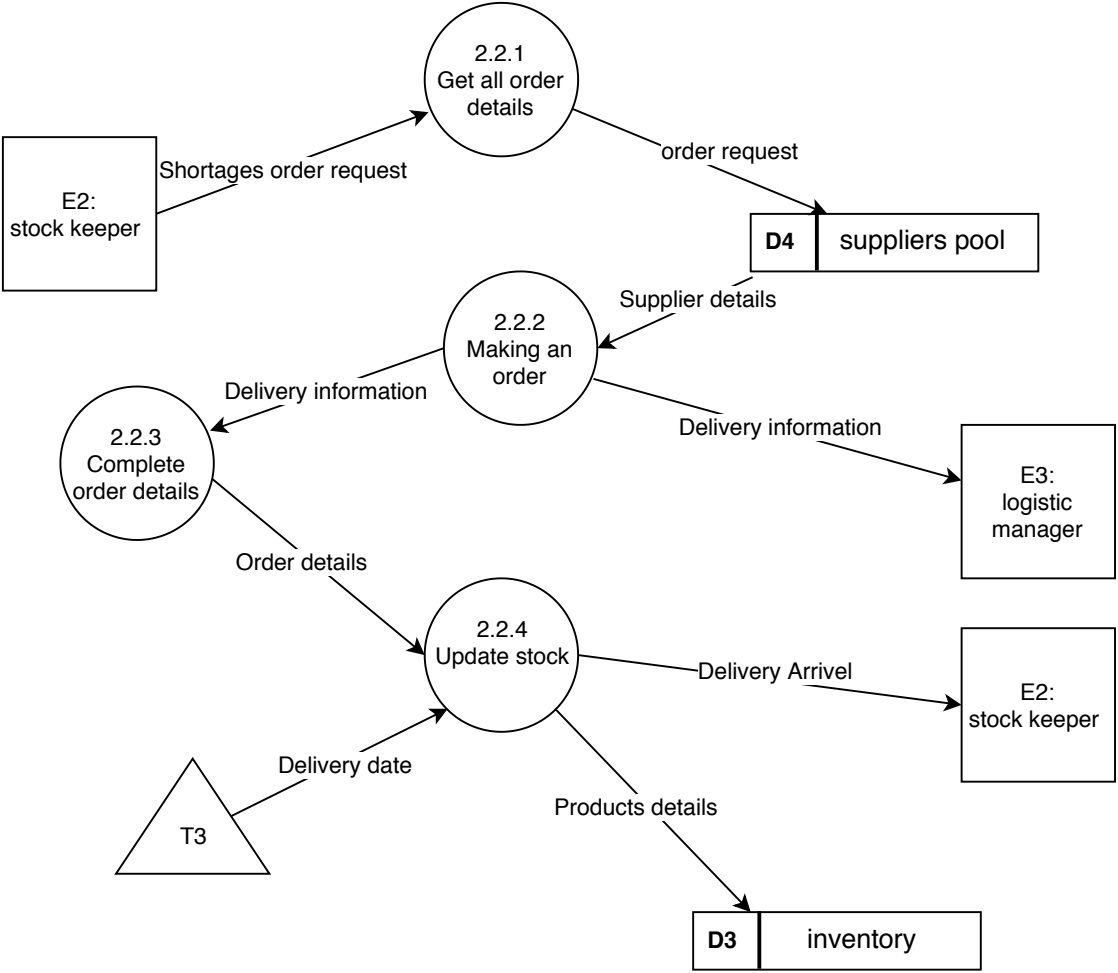

Sites
area
address
dependent
contact_person
of
IS A
X
Branches
id
delivery to\from
Site_Reports
order
weight_on_site
in
Products_Delivery
amount
delivery to\from
User
password
username
belongs to
bank number
number
branch number
bank account
salary
first employed
Employee
last name
first name
ID
has
date
shift time
Constraint
IS A
Drivers
license
driver
Delivery_Doc
managed by
is assigned to
role_id
has a
description
Role
ID
Shifts
date
ID
time
has different
delivered by
of
Trucks
max_weight
base_weight
type
license_plate
Deliveries
delivery_id
start_date
start_time
delivered
shift in
SupplierType1
SupplierType2
SupplierType3
supplierNum
day
Type of
Suppliers
name
Contact Of
bankAccount
paymentCond
phoneNum
Contacts
cellphone
supplierNum
name
Presented by
Given by
Company
companyNum
supplierNum
name
Discount
code
supplierNum
minAmount
discountPer
Discounr on
Soled by
SupplierProduct
code
supplierNum
weight
amount
description
price
delivered
DateEnded
Discount
DiscountID
Precentage
DateGiven
Given On
Products
OriginalsellingPrice
OriginalCostPrice
CostPrice
Barcode
sellingPrice
Name
MinimalAmount
Manufactor
Describes amount of
Product Category
Hierarchy
Stock
Barcode
ExipirationDay
LocationID
Quantity
GivenOn
Category
CategoryID
Discount
Name
Discounted
Moved To
Sells
Defects
Barcode
LocationID
DateReported
Quantity
Reason
Located In
Branches
BranchID
Name
Locations
LocationsID
BranchID
Text
PhusicalPlace
PlaceIdetifier
Located In


E1:
HR manager
employee & shift
details
1
HR
management
shift details
D1
shifts pool
D2
employees pool
employee details
E2:
stock keeper
alert when a delivery arrives
E4:
store
manager
Inventory Reports
E2:
stock keeper
product & supplier's
order details
2
Inventory
management
product details
D3
inventory
Delivery date
T3
today's date
supplier's schedule &
products
D4
suppliers pool
supplier's order details
T1
alert when a delivery fails
E1:
HR manager
3
Delivery
management
delivery details
D5
deliveries pool
E3:
logistic
manager
delivery details
Available Trucks
D6
trucks pool
shift details
todays date
D1
shifts pool
T2
D4
suppliers pool
supplier details
4
Supplier
management
product details
D4
suppliers product pool
E2:
stock keeper
suppliers details

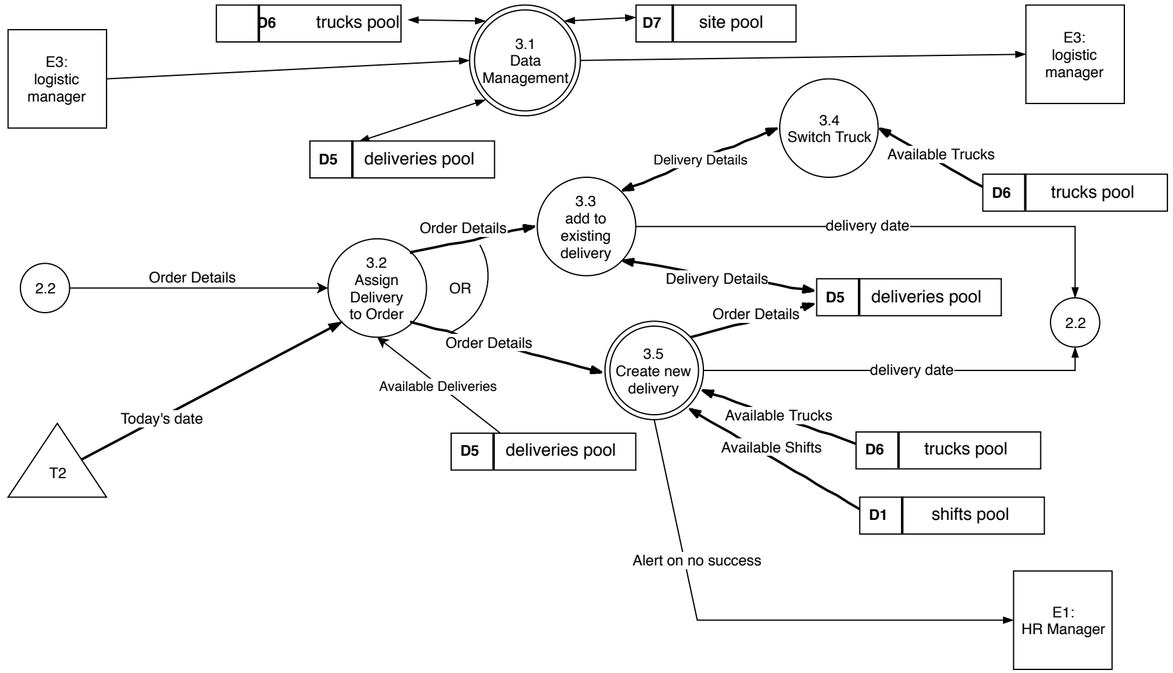

Employees DFD-1
ID
1.1
manage shifts
assigning
1.2
view shift
assignings
shift assignings details
E2:
Employee
E1: HR
Manager
shift assign details
or
and
all assigns details
1.3
delete shift
assigning
1.4
insert shift
assigning
constraint
details
E1: HR
Manager
E2:
Employee
assign details
assign details
shift details
ID, constraint
details
1.15
edit constraints
D1
shifts
assigning
D2
emp
constarints
all assigns details
E1: HR
Manager
shift details
1.6
update shifts
ID, constraint details
all assigns details
E1: HR
Manager
E2:
Employee
1.7
delete shift
or
1.5
view shifts
all shifts details
D3
shifts
1.8
insert shift
shift details
shift details
employee ID
E1: HR
Manager
1.9
manage
employees
all emps details
E1: HR
Manager
or
and
1.12
delete emp
1.10
view emps
all emps details
D4
employees
emp ID, job
1.11
add emp
employee details
employee ID
1.13
add job to
emp
emp ID, job
1.14
view emps
constraints
all emps constraints
all emps constraints
E2:
Employee
ID
constarints
E2:
Employee


T1
today's date
2.1
Periodic
Order
E2:
stock keeper
enter inventory system
supplier schedule
D4
suppliers pool
inventory details
2.2
Order
shortages
product details
D3
inventory
Supplier Products
D4
suppliers pool
2.3
Edit
Discounts
product details
E2:
stock keeper
Discount details
product details
D3
inventory
product details
2.4
Edit Products
product details
product details
2.5
Remove from
Stock or
change
location
2.4
Edit Stock
product details
OR
E2:
stock keeper
product details
product details
2.6
Display Stock
product details
product details
2.7
Edit Defects
product details
D3
inventory
E4:
store
manager
Invntory Reports
2.8
Produce
Inventory
reports
inventory details

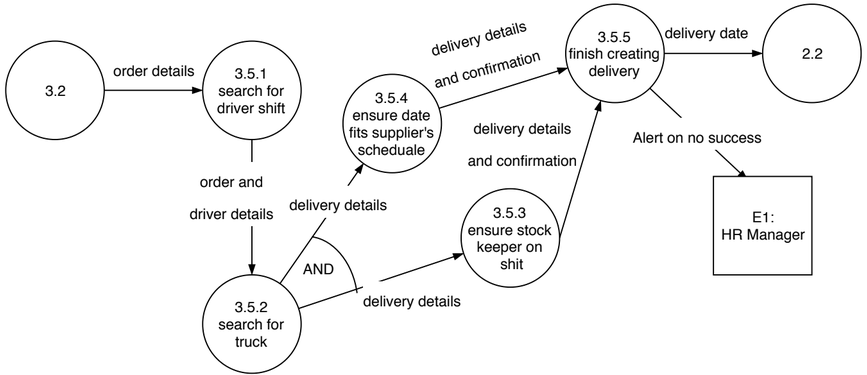

2.1.1
Get all order details
E2:
stock keeper
Periodic order request
order request
D4
suppliers pool
Supplier details
2.1.2
Making an order
Delivery information
2.1.3
Complete order details
Delivery information
E3:
logistic manager
Order details
2.1.4
Update stock
Delivery Arrivel
E2:
stock keeper
Delivery date
T3
Products details
D3
inventory


2.2.1
Get all order details
E2:
stock keeper
Shortages order request
order request
D4
suppliers pool
Supplier details
2.2.2
Making an order
Delivery information
2.2.3
Complete order details
Delivery information
E3:
logistic manager
Order details
2.2.4
Update stock
Delivery Arrivel
E2:
stock keeper
Delivery date
T3
Products details
D3
inventory

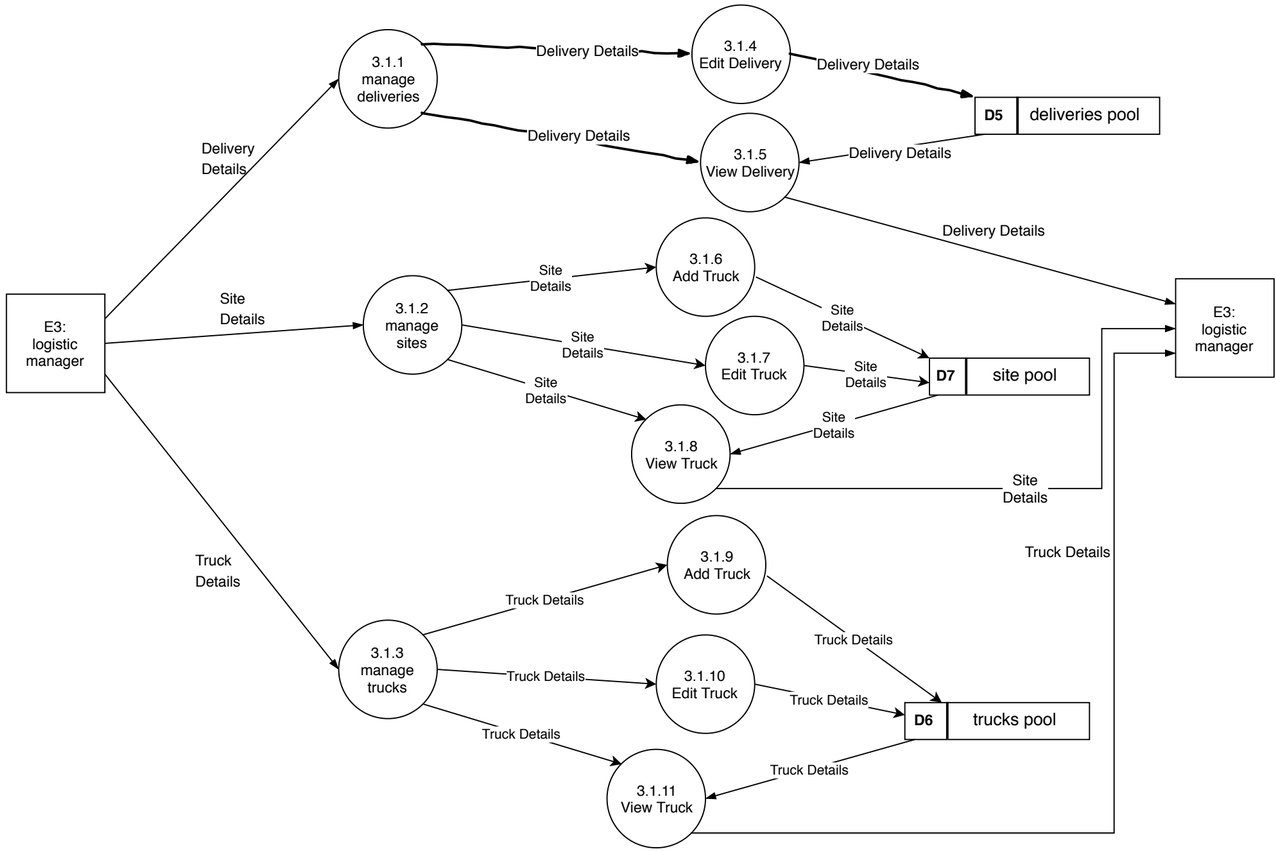
DFD-3



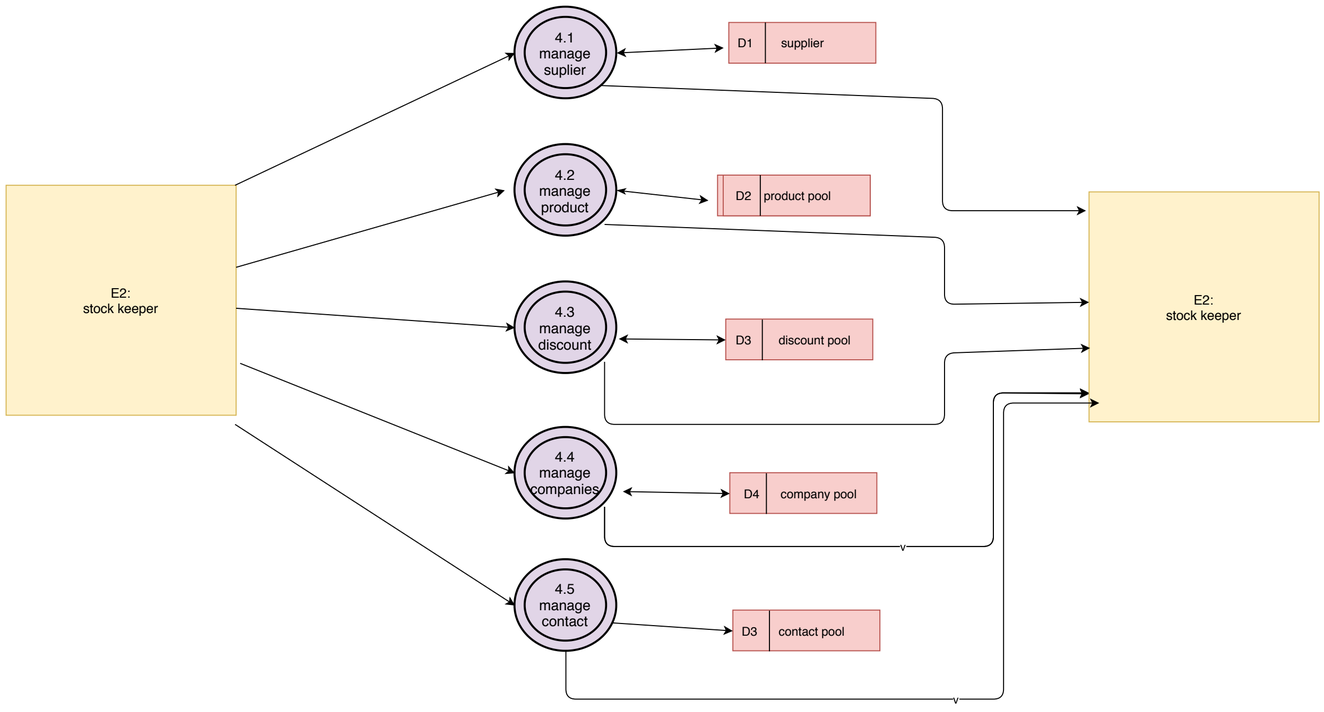
DFD-3.5



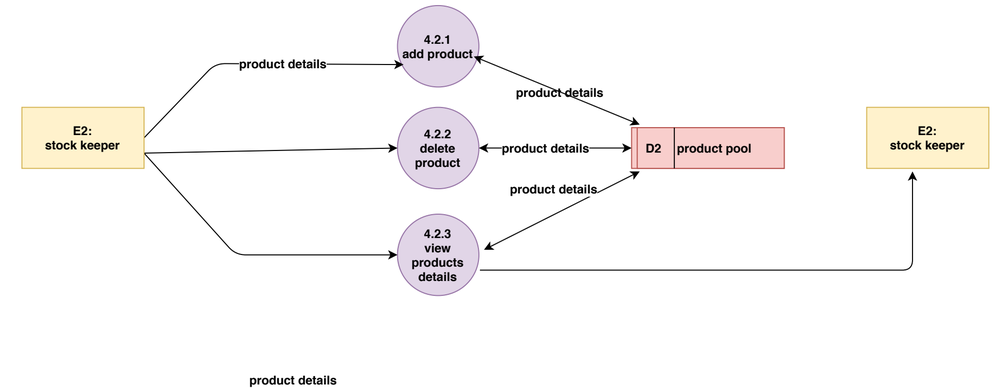
DFD-3.1



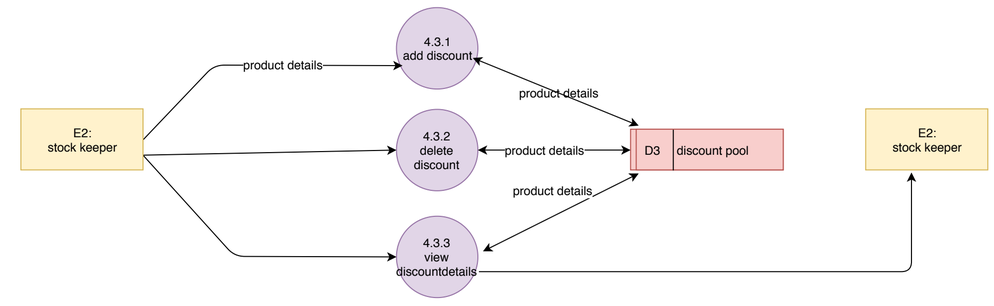
DFD 4



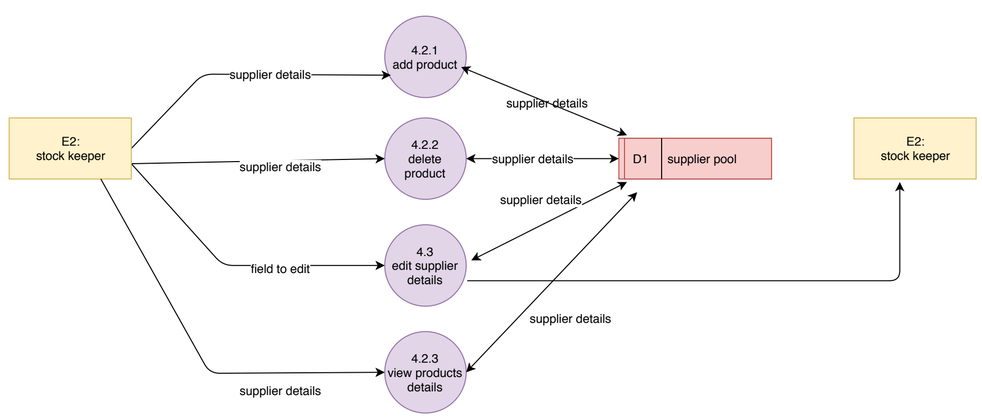
DFD 4.2



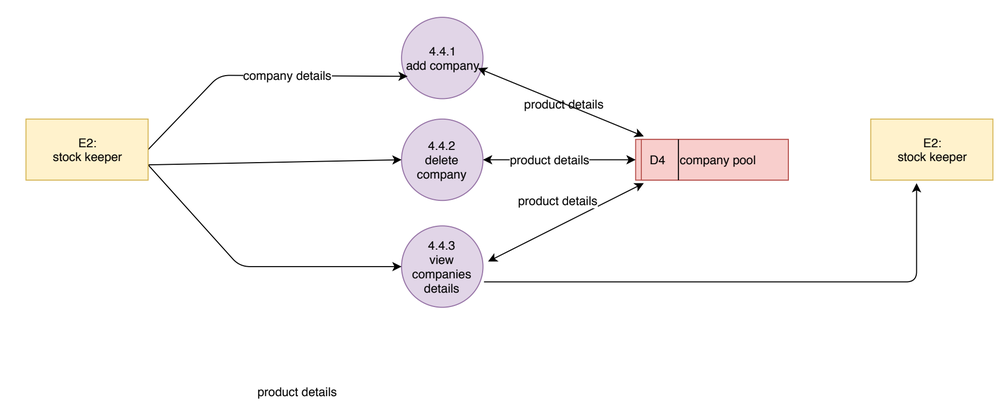
DFD 4.3



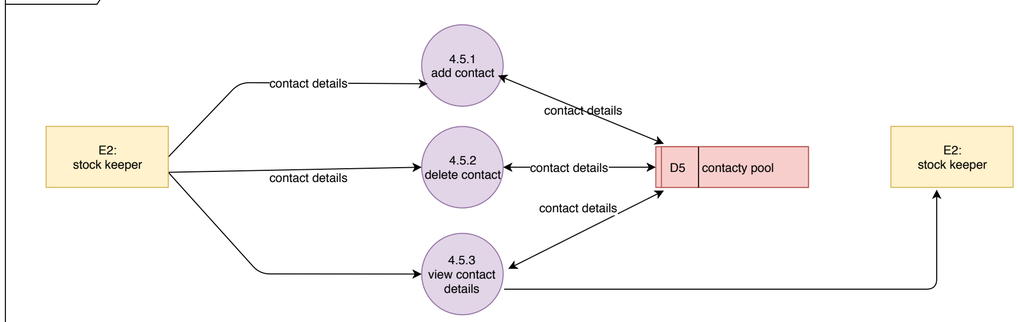
DFD 4.1



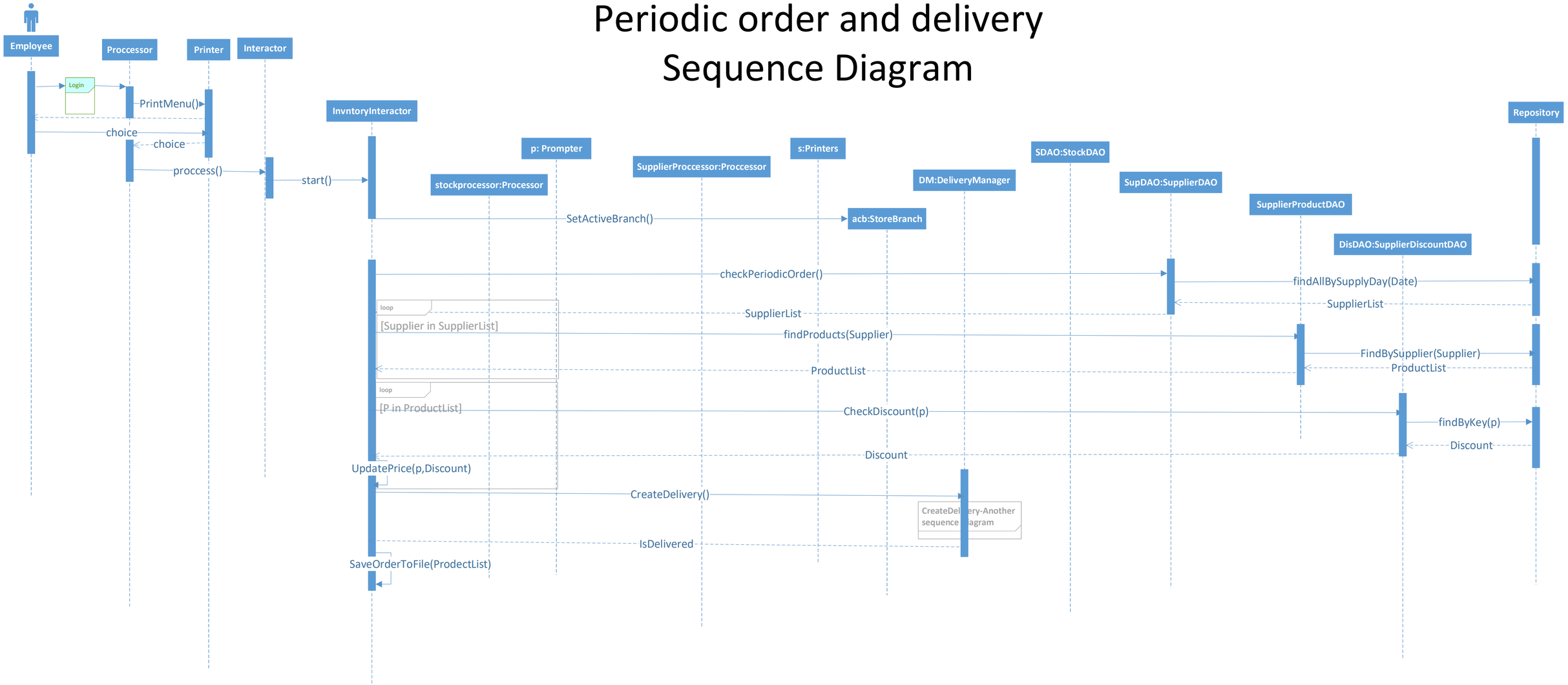
DFD 4.4



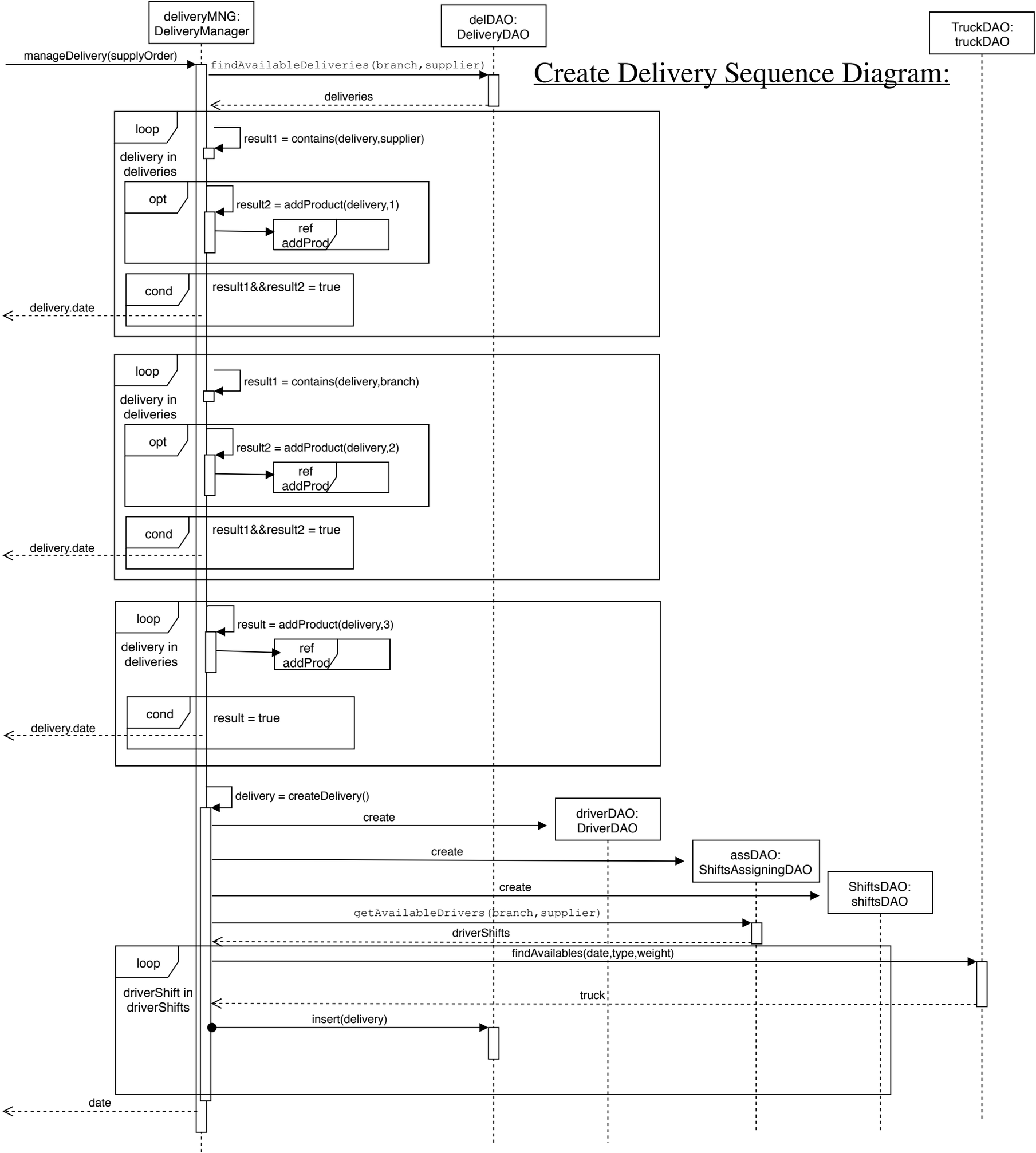
DFD 4.5



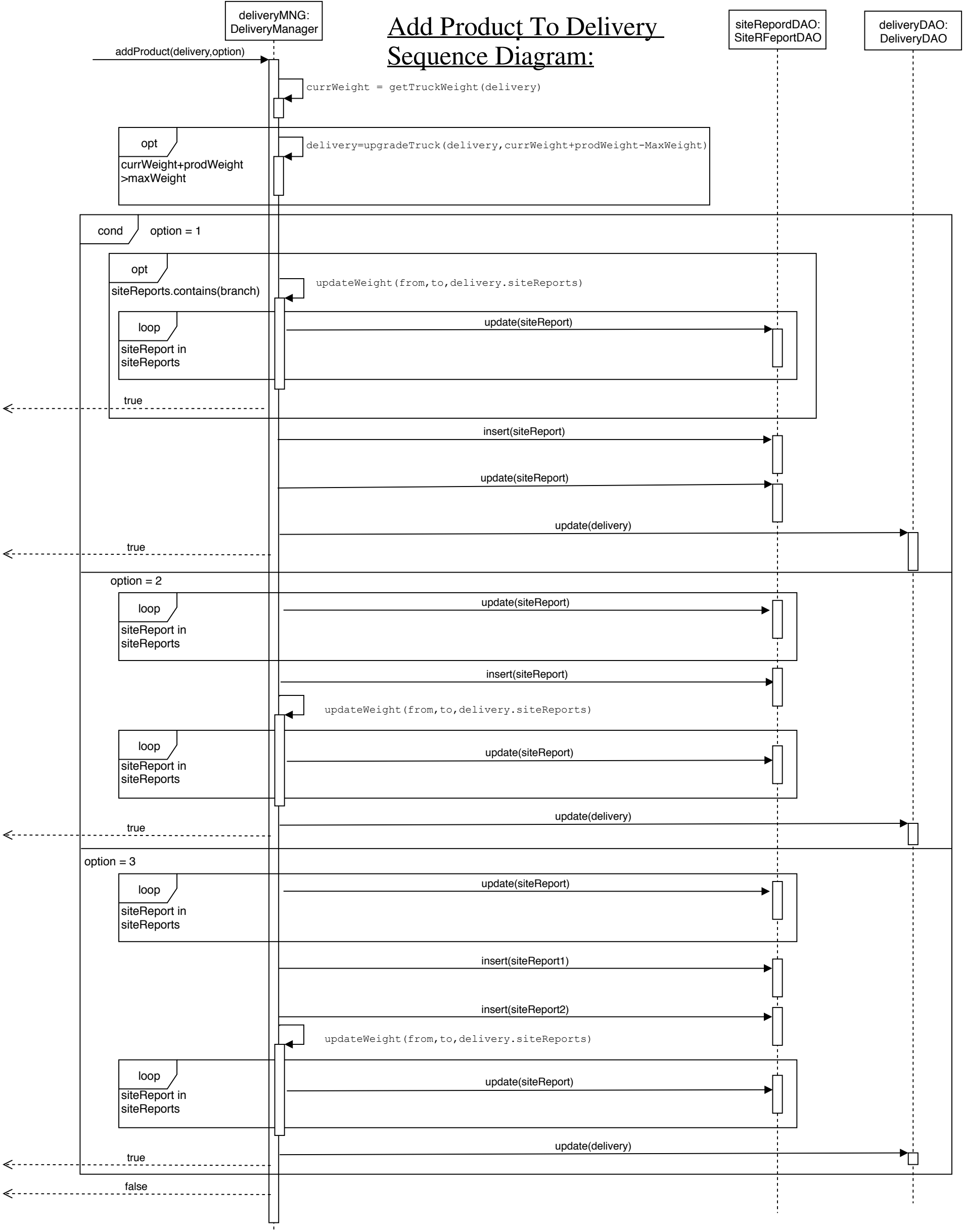
Periodic order and delivery Sequence Diagram



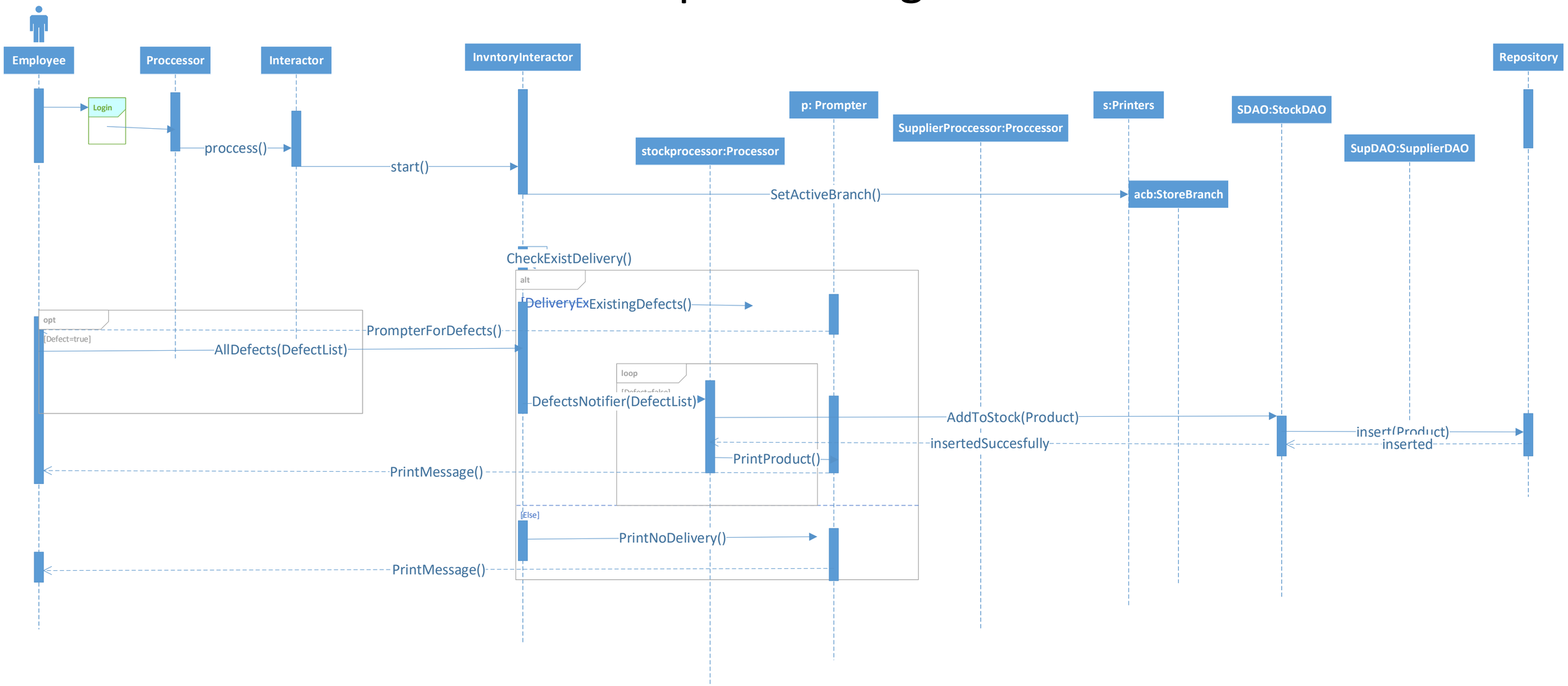
Create Delivery Sequence Diagram:



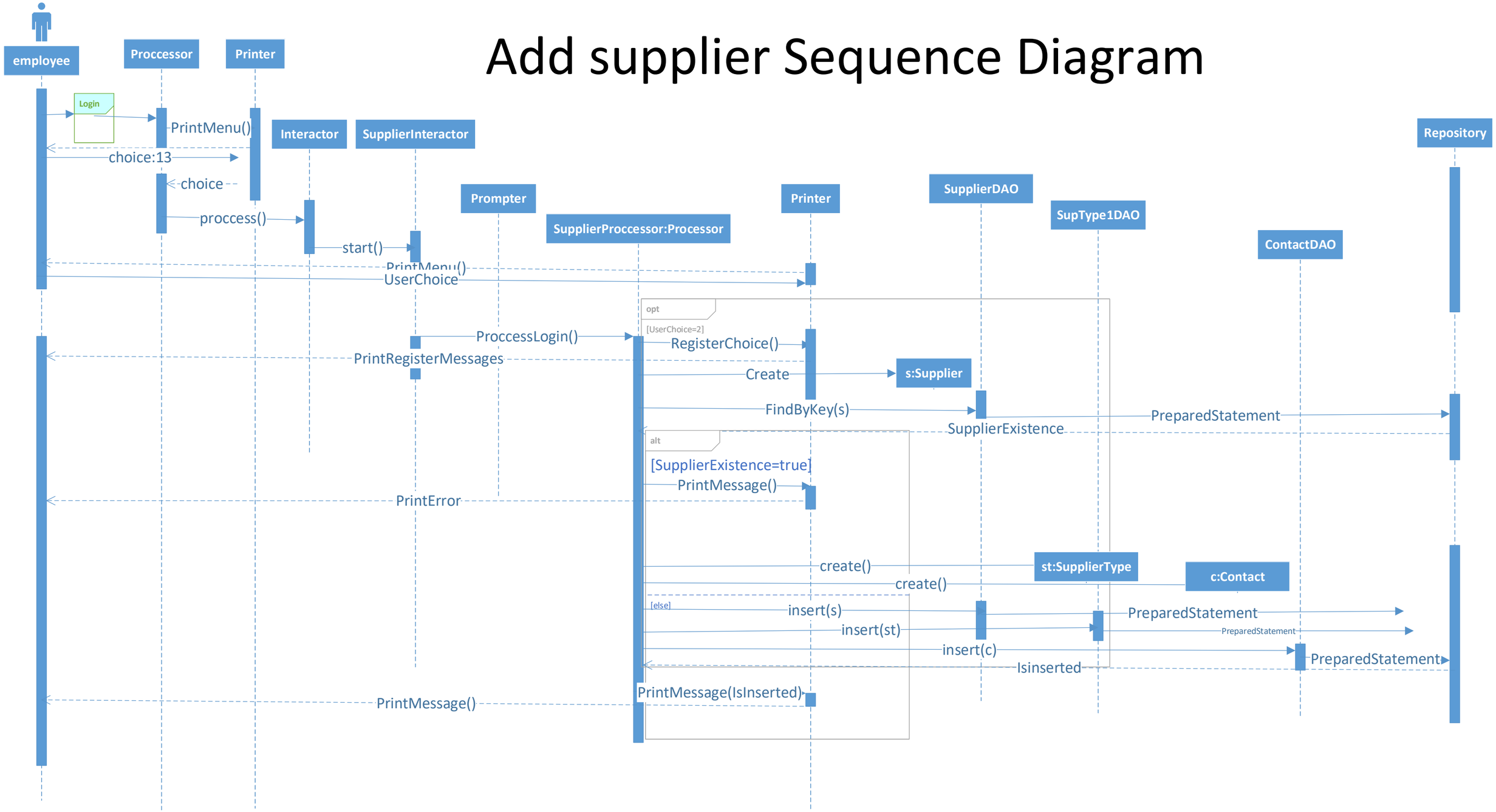
Add Product To Delivery Sequence Diagram:



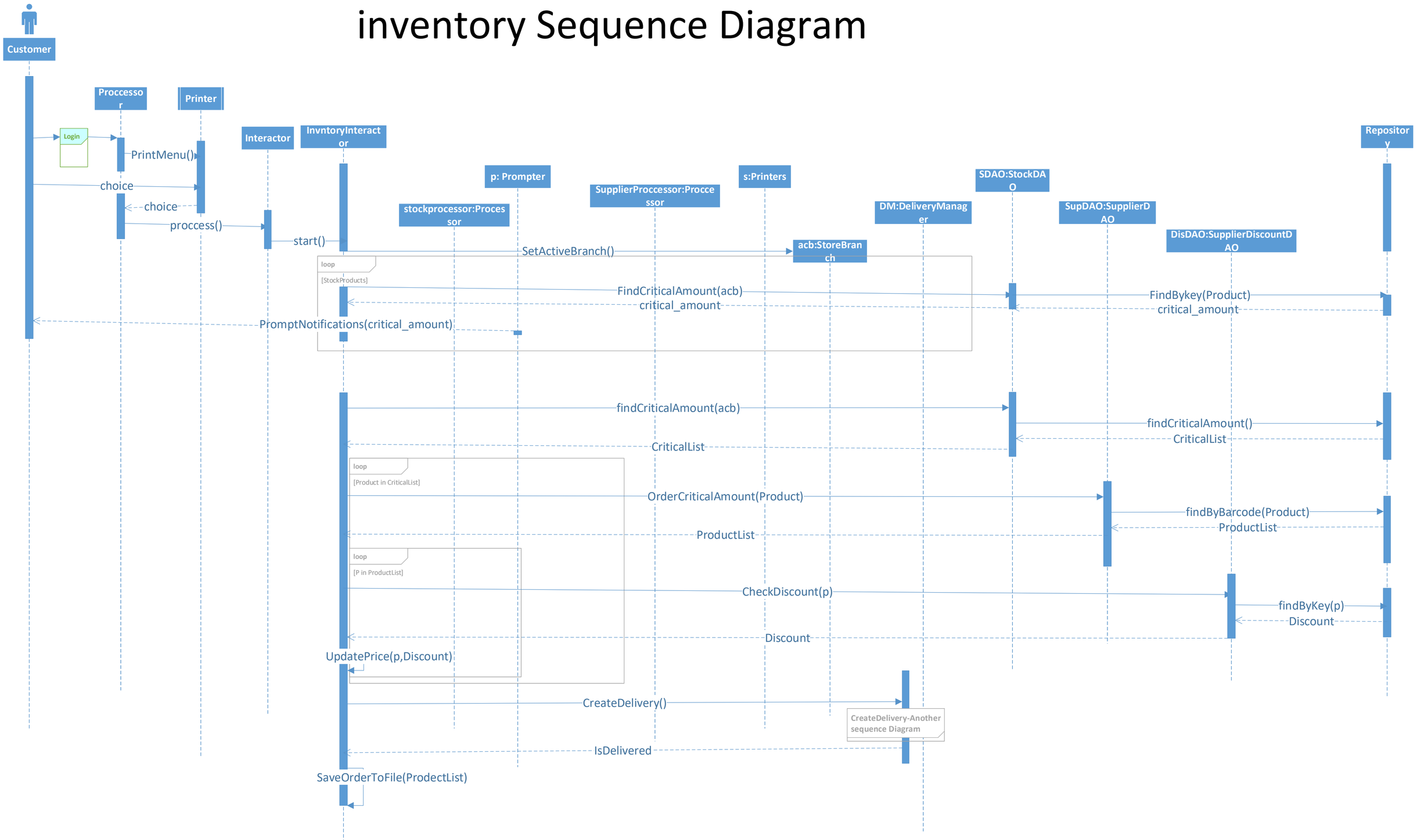
Receiving delivery and update inventory Sequence Diagram



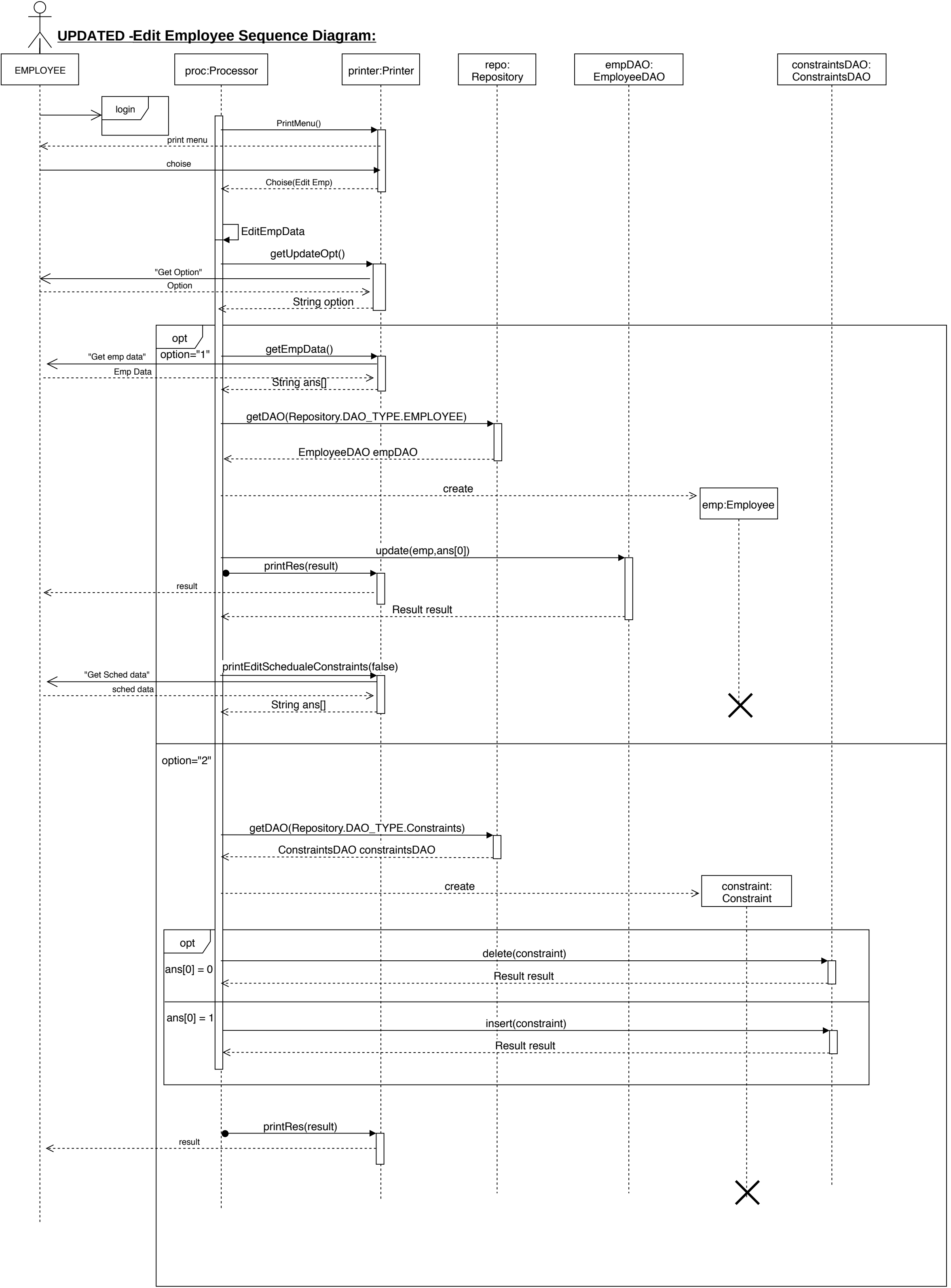
Add supplier Sequence Diagram



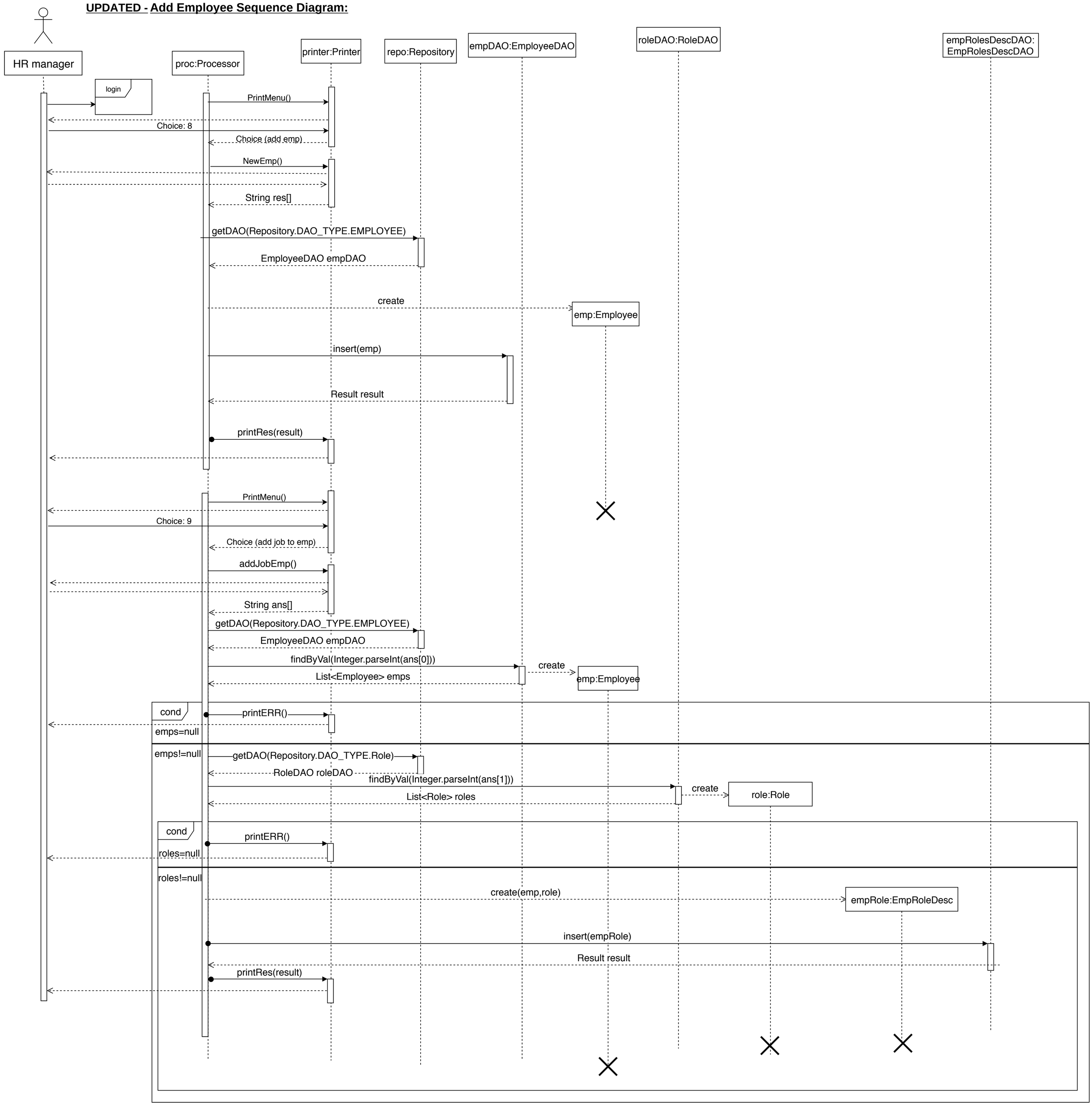
Order due to shortage and update inventory Sequence Diagram



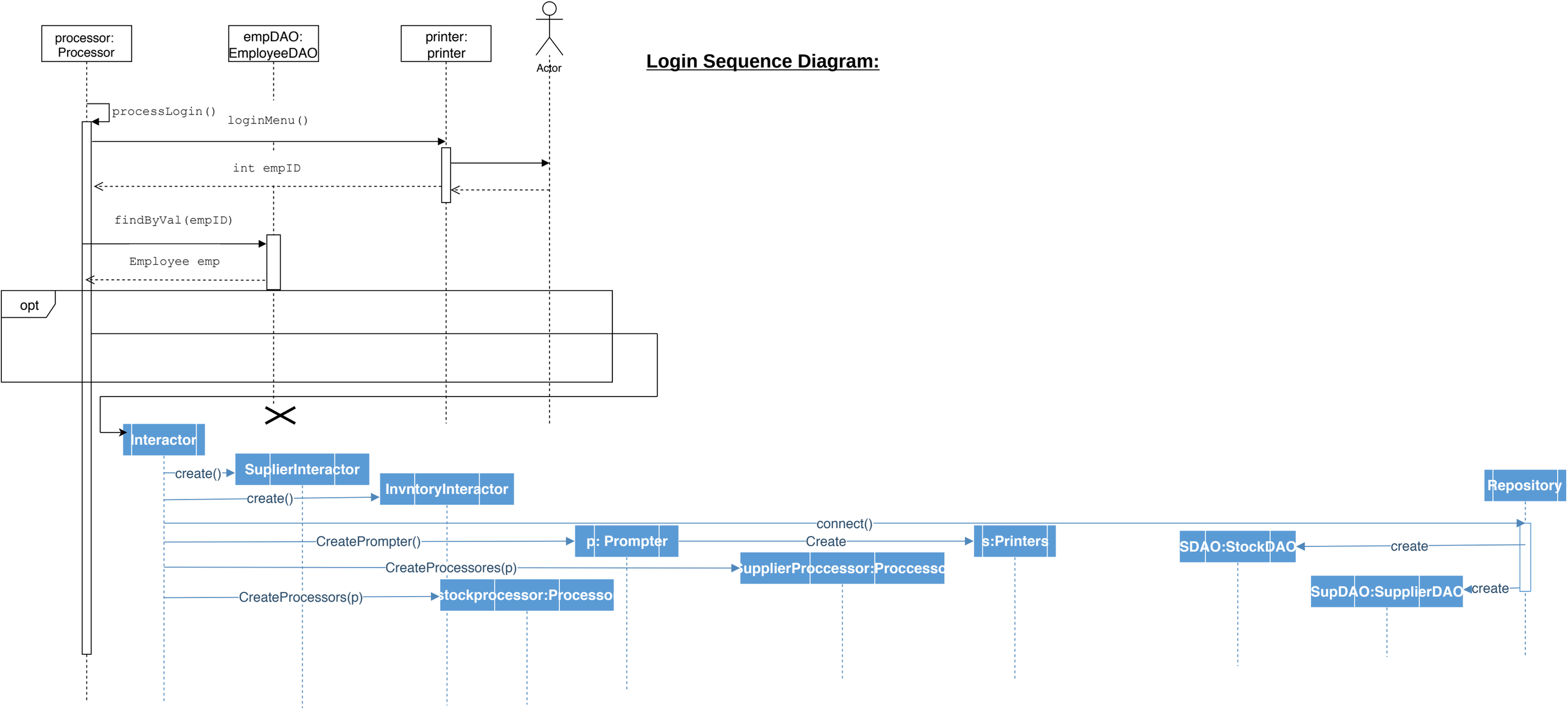
UPDATED -Edit Employee Sequence Diagram:



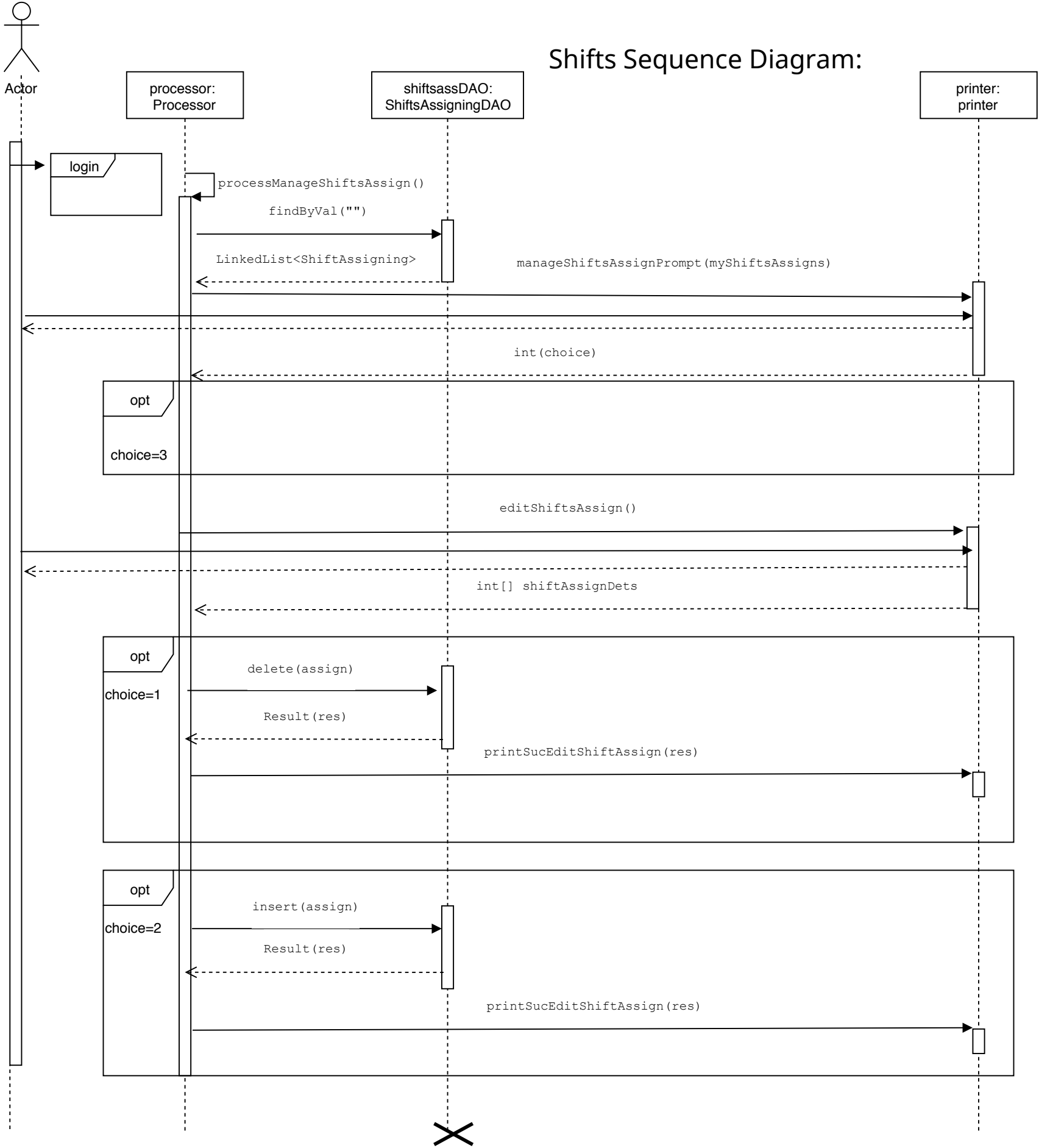
UPDATED - Add Employee Sequence Diagram:



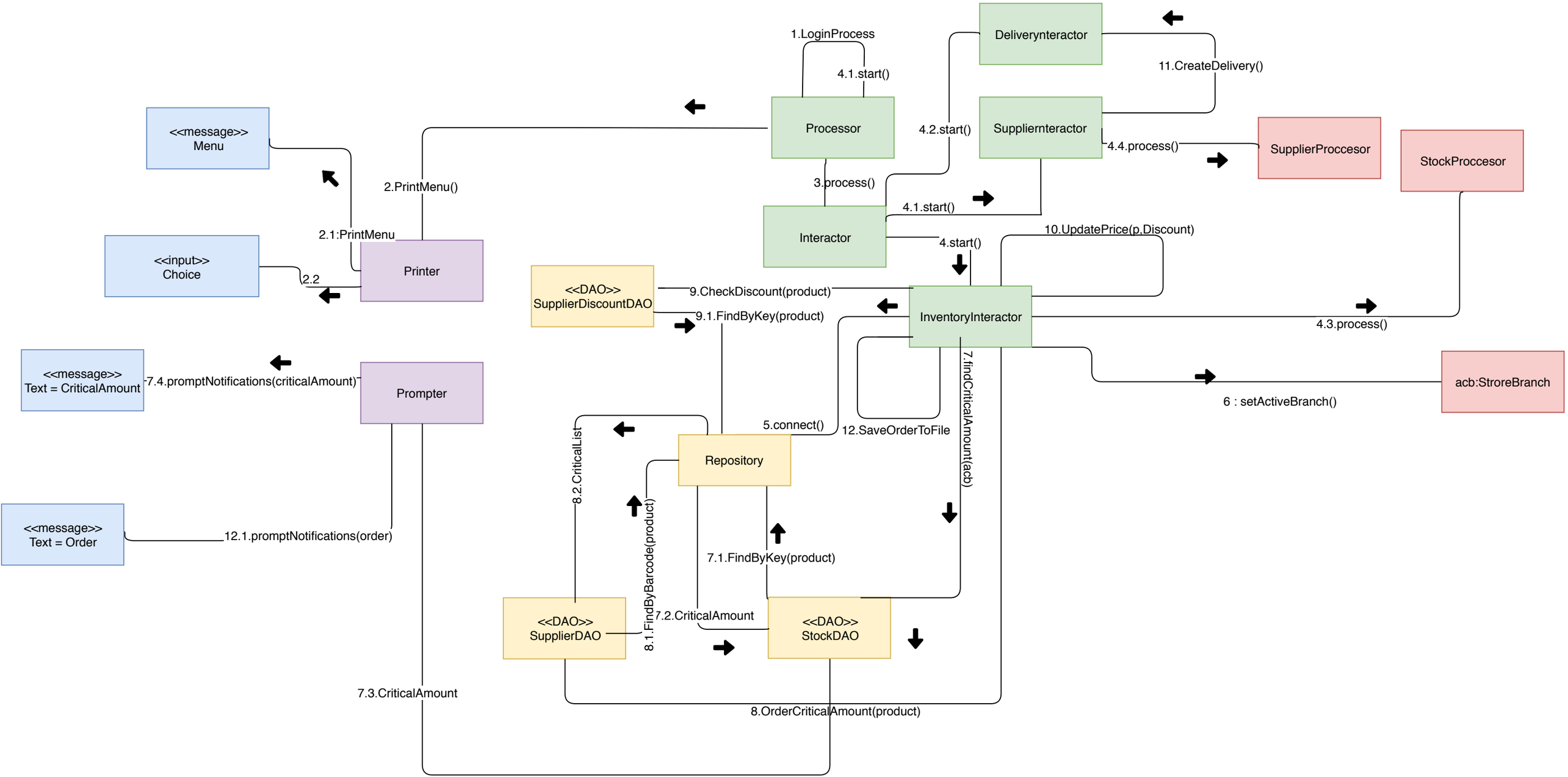
Login Sequence Diagram:



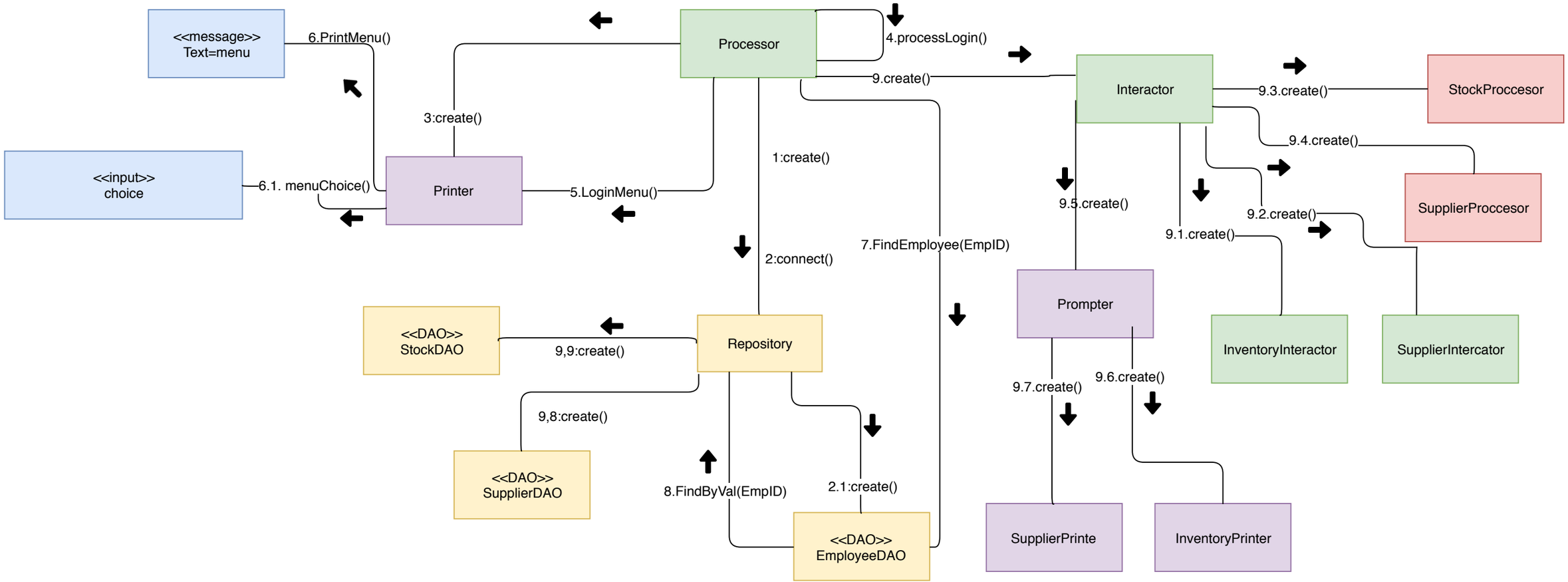
Shifts Sequence Diagram:



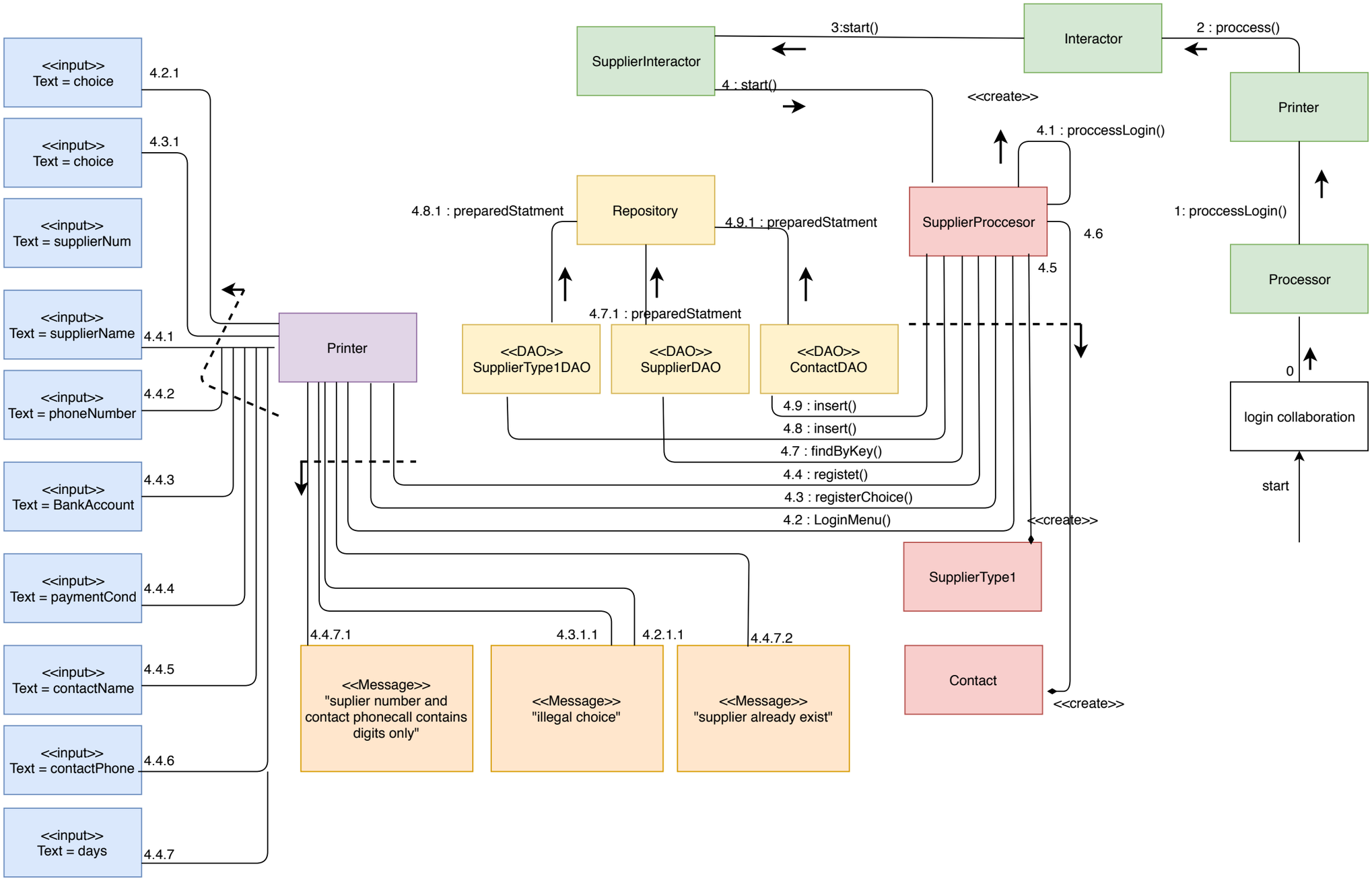
Collaboration Diagram - Update stock and shortage



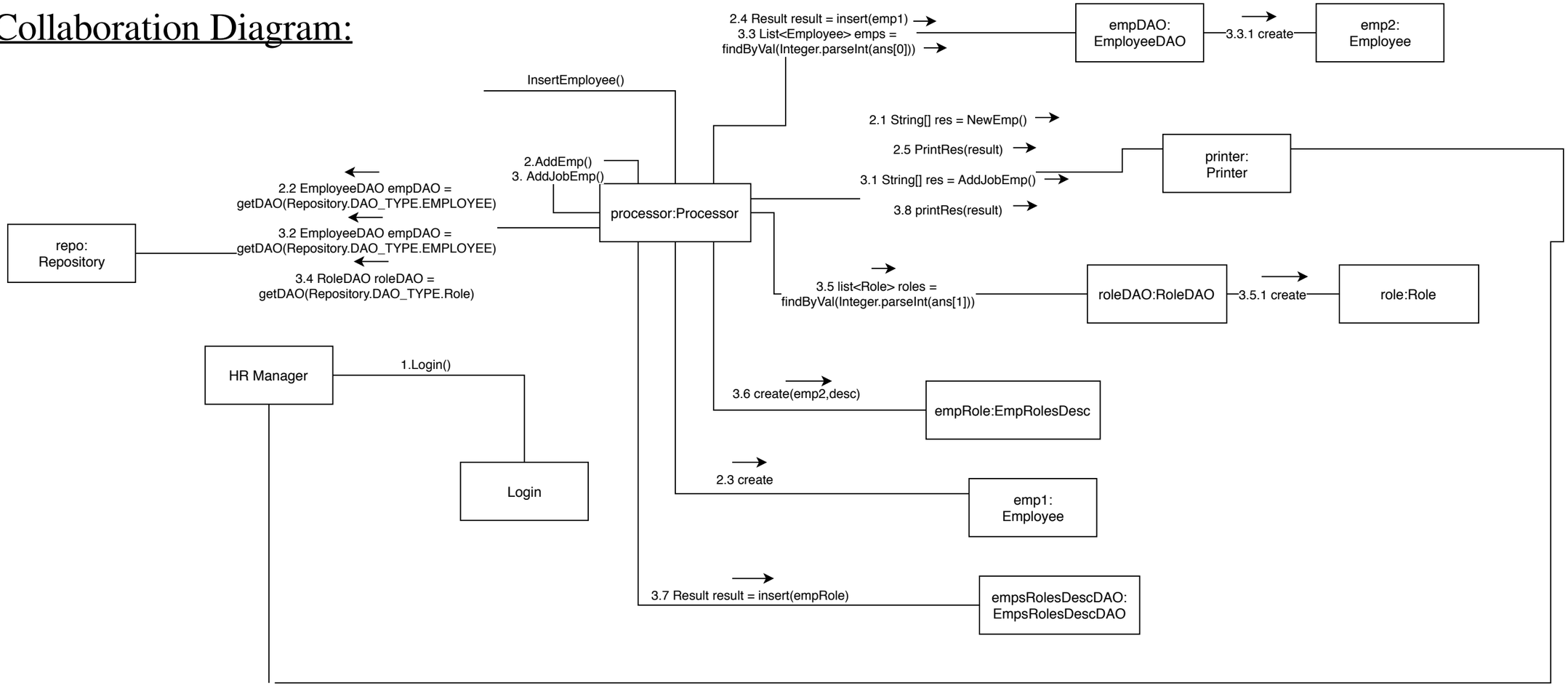
Collaboration Diagram - System Login



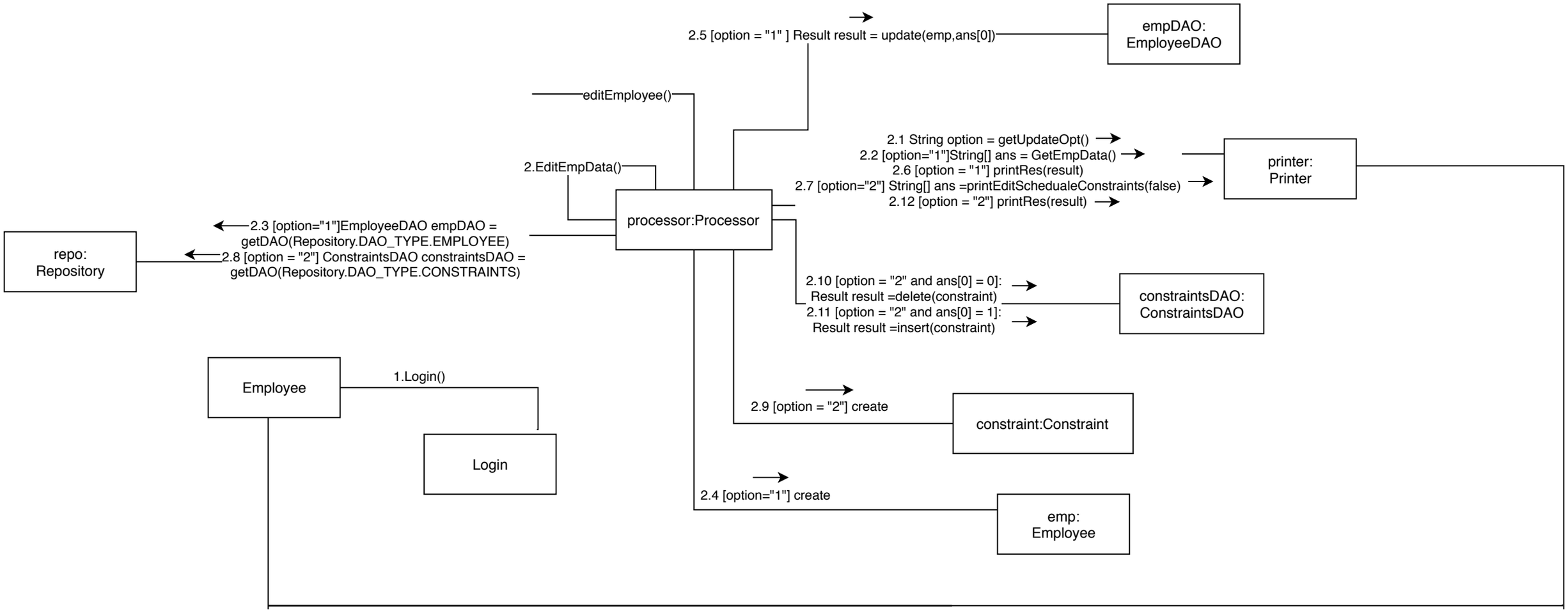
Collaboration Diagram - Adding new Supplier



Add Employee Collaboration Diagram:



Edit Employee - collaboration Diagram:



Menus Tree:

Menu 0:
S HR management.....(1)
S Inventory management .....(2)
S Delivery management.....(3)
S Supplier management.....(4)

Menu 1:
T manage shifts assigning.....1.1/1.2/1.3/1.4
T view shifts.....1.5
T update shifts 1.6/1.7/1.8
T manage employee 1.9/1.10/1.11/1.12
T add job to emp.....1.13
T view emps constraints.....1.14
T edit constraint constraints.....1.15

Menu 2:
S periodic order.....2.1
QT order shortage.....2.2/ 3.5.1/ 3.5.2 / 3.5.4/ 3.5.5/2
T edit discount and product.....2.3/2.4
T edit stock.....2.5/2.6/2.7
T edit defects.....2.8
T produce inventory reports.....2.9

Menu 2.1:
T periodic order.....2.1.1
T order shortage.....2.1.2/2.1.3/2.1.4

Menu 3:
S data managment 3.1
S assign delivery to order..... 3.3/ 3.4/ 3.5

Menu 3.1:
T manage deliveries.....3.1.1/ 3.1.4/ 3.1.5
T manage sites..... 3.1.2/ 3.1.6/ 3.1.7/ 3.1.8
T manage trucks.....3.1.3/ 3.1.9/ 3.1.10/ 3.1.11

Menu 4:
S manage supplier.....4.1
S manage product.....4.2
S manage discount.....4.3
S manage company.....4.4
S manage contact.....4.5

Menu 4.1:
T add new supplier card.....4.1.1
T delete supplier card.....4.1.2
T edit supplier card.....4.1.3
T view supplier details.....4.1.4

Menu 4.2:
T add product.....4.2.1
T delete product.....4.2.2
T view product.....4.2.3

Menu 4.3:
T add discount.....4.3.1
T delete discount.....4.3.2
T view discount.....4.3.3

Menu 4.4:
T add company.....4.4.1
T delete company.....4.4.2
T view company.....4.4.3

Menu 4.5:
T add contact.....4.5.1
T delete contact.....4.5.2
T view contact.....4.5.3