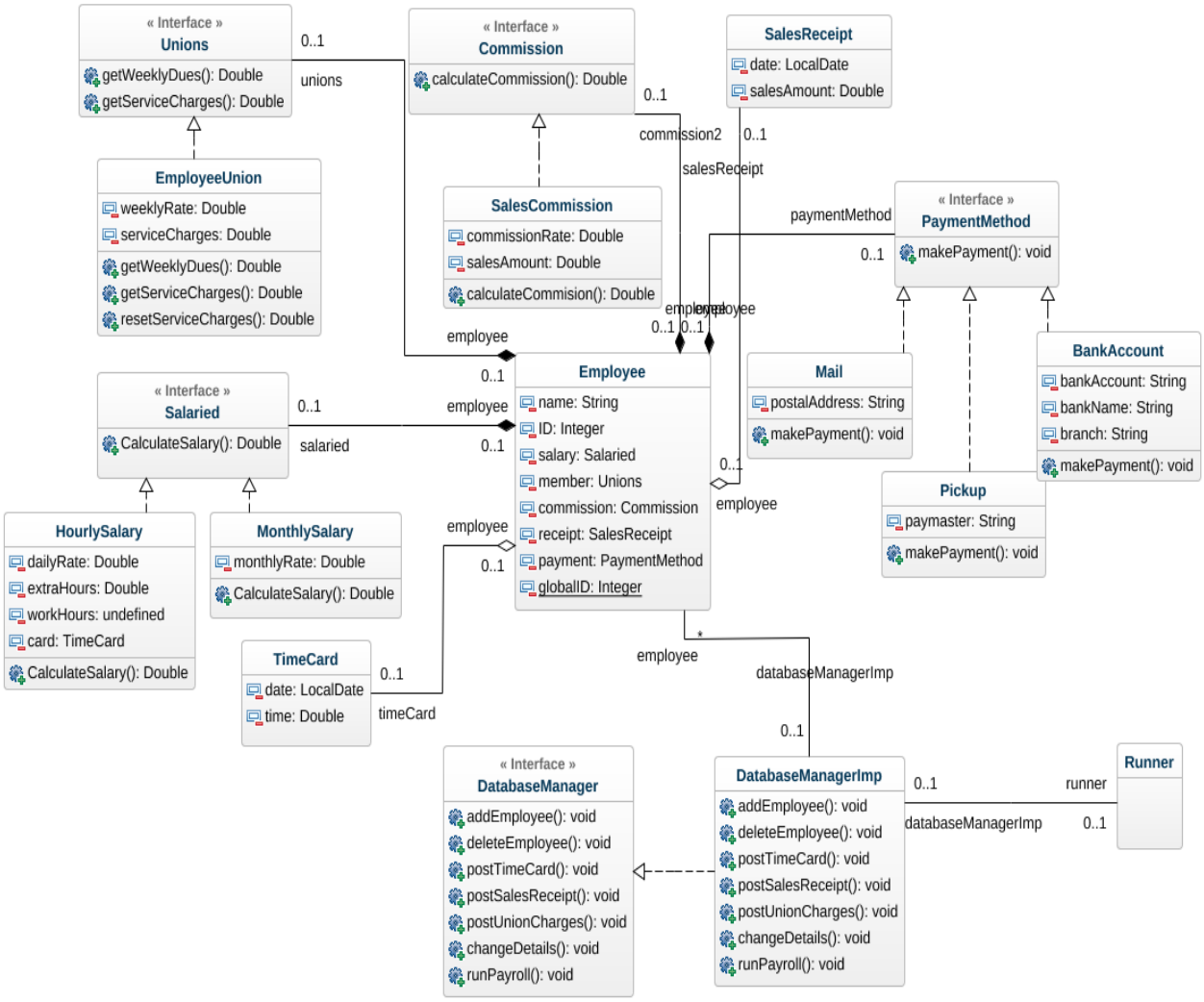
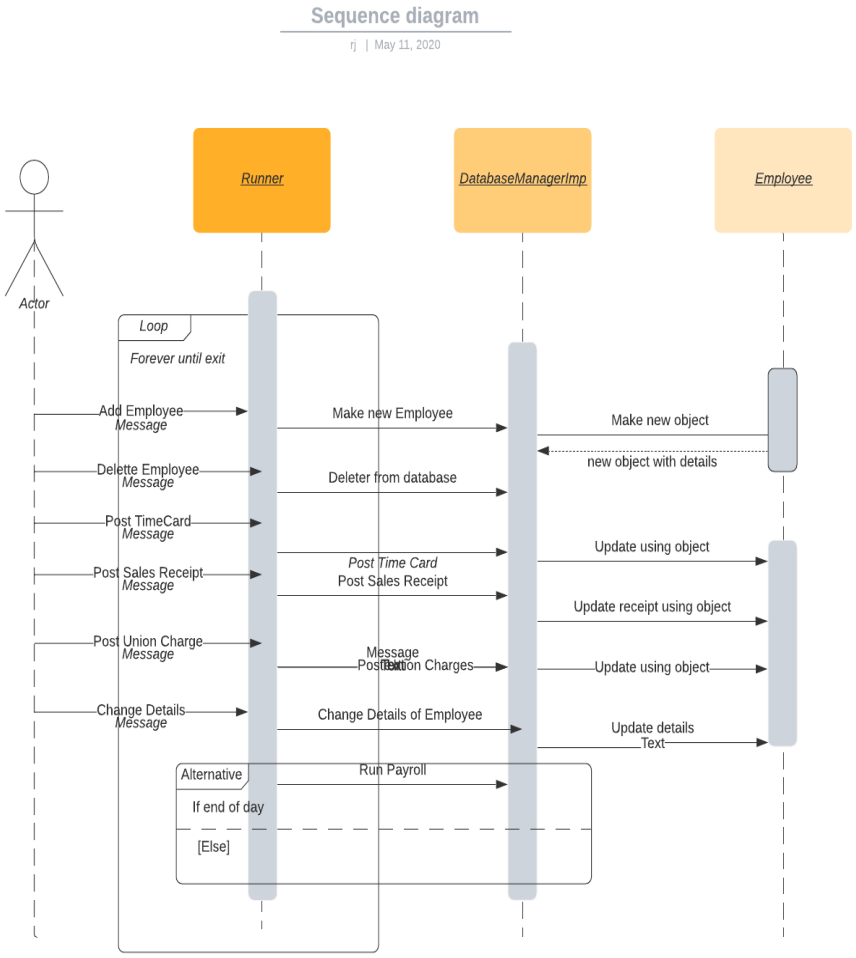
Documentation for Payroll System Design

By Rahul Jha

UML Diagram


« Interface »
Unions
getWeeklyDues(): Double
getServiceCharges(): Double
0..1
unions
« Interface »
Commission
calculateCommission(): Double
SalesReceipt
date: LocalDate
salesAmount: Double
0..1
commission2
0..1
salesReceipt
EmployeeUnion
weeklyRate: Double
serviceCharges: Double
getWeeklyDues(): Double
getServiceCharges(): Double
resetServiceCharges(): Double
SalesCommission
commissionRate: Double
salesAmount: Double
calculateCommision(): Double
paymentMethod
« Interface »
PaymentMethod
makePayment(): void
0..1
employee
0..1
employee
0..1
0..1
« Interface »
Salaried
CalculateSalary(): Double
0..1
salaried
employee
0..1
Employee
name: String
ID: Integer
salary: Salaried
member: Unions
commission: Commission
receipt: SalesReceipt
payment: PaymentMethod
globalID: Integer
Mail
postalAddress: String
makePayment(): void
BankAccount
bankAccount: String
bankName: String
branch: String
makePayment(): void
0..1
employee
Pickup
paymaster: String
makePayment(): void
HourlySalary
dailyRate: Double
extraHours: Double
workHours: undefined
card: TimeCard
CalculateSalary(): Double
MonthlySalary
monthlyRate: Double
CalculateSalary(): Double
employee
0..1
TimeCard
date: LocalDate
time: Double
0..1
timeCard
*
employee
databaseManagerImp
0..1
« Interface »
DatabaseManager
addEmployee(): void
deleteEmployee(): void
postTimeCard(): void
postSalesReceipt(): void
postUnionCharges(): void
changeDetails(): void
runPayroll(): void
DatabaseManagerImp
addEmployee(): void
deleteEmployee(): void
postTimeCard(): void
postSalesReceipt(): void
postUnionCharges(): void
changeDetails(): void
runPayroll(): void
0..1
runner
databaseManagerImp
0..1
Runner

Sequence Diagram



Assumptions:

1. I have assumed that any employee may not be part of more than one union or be given commission from more than one source, if in the future, any more unions or commissions are added.
2. The details entered in the CLI are appropriate because no checks for incorrect input have been added.
3. Service charges for the union are specific to each employee.

Design Choices:

1. I have made an interface for salary so that if any other method of salary payment is involved in the future it may be incorporated as well.
2. I have made an interface for Commission and Union as well in case of further additions to either.
3. Payment Method is an interface that is implemented by three classes: Mail, Pickup and BankAccount.
4. TimeCard and SalesReceipt are separate classes. TimeCard has been incorporated in the HourlySalary implementation of the salary interface because it had no use outside of salary calculation. However, I have kept SalesReceipt as a member variable of Employee type, although it could be included in the SalesCommission class which is the implementation of Commission interface because the SalesReceipt class may be used in calculation of other commissions added in the future, which would mandate the inclusion of a separate SalesReceipt member variable in every such implementation of Commission interface.
5. I have included a Union variable as a member of Employee class so that the Union information can be stored for each employee. This is because the weekly dues and service charges for different employees may differ as they some may be able to pay out the dues immediately while some may not. If the Union is null, then the employee is not a part of any Union.
6. I have created an interface Database Manager to implement the use cases. DatabaseManagerImp realises this interface.
7. The Runner class runs the application by providing the options in the DatabaseManger interface.