VAIBHAV SHARMA

FKPayroll App

Objectives

To create a payroll system which will run every day take input from user and generate salary of Employees.
The system has one admin which will be the only user.
The functions which can be performed are

1. Add an employee.
2. Delete an employee.
3. Update Employee Details.
4. Post time card
5. Post sales receipt
6. Generate salary upto the input date.
7. Update union details.

Assumptions

- **Input given by admin is in correct format.**
- **Date entered by admin are in increasing order.**
- **Some Salaried Employee can also get commission.**
- **Payment is made only at friday for HourEmployee.**
- **Any employee can be a union memeber.**

My Design Approach

An Employee can be of two type. Type 1 is hour employee, employee that is paid every friday based on how much hours he has worked in a day.
And type 2 are salaried employee. These employee are paid two types of income f

- Basic salary at the end of month.
- Commission, based on the sales made , every alternate friday.

Type 1 employee class is HourEmployee
Type 2 employee class is SalariedEmployee

Both these classes implements Employee interface which has a function caculatesalary() (Arguments and return type can be seen in code)

To store all employees data and other information mysql databse Fkpayroll is created.
This database has 3 tables:

1. **Employee table:**
    - firstName
    - lastName
    - modeOfSalary this can take values 1 , 2 or 3. 1 means Employee wil pick his salary from pay master. 2 means salary will be posted to address given in postalAdd field. 3 means salary will be credited to account immediately.
    - lastPaidDate stores last time employee was paid.
    - Rate has different meaning depending on the type of employee.If type is 1 then rate signifies how much employee is paid per hour. If type is 2 rate is what percentage of sales amount employee get as commission.
    - isUnionMember tells whether employee is a union member or not.
    - weeklyDue is the union weekly cost which is deducted from employee salary.
    - Type
    - Salary is not defined in case of type 1 (Hour Employee)
    - Id is the primary key auto increment.
    - lastComm stores the date last time Salaried Employee was paid his commission.
    - postalAdd stores postal address for employee with mode of payment as 2.
    - accountNo stores account number of employee to credit salary into.

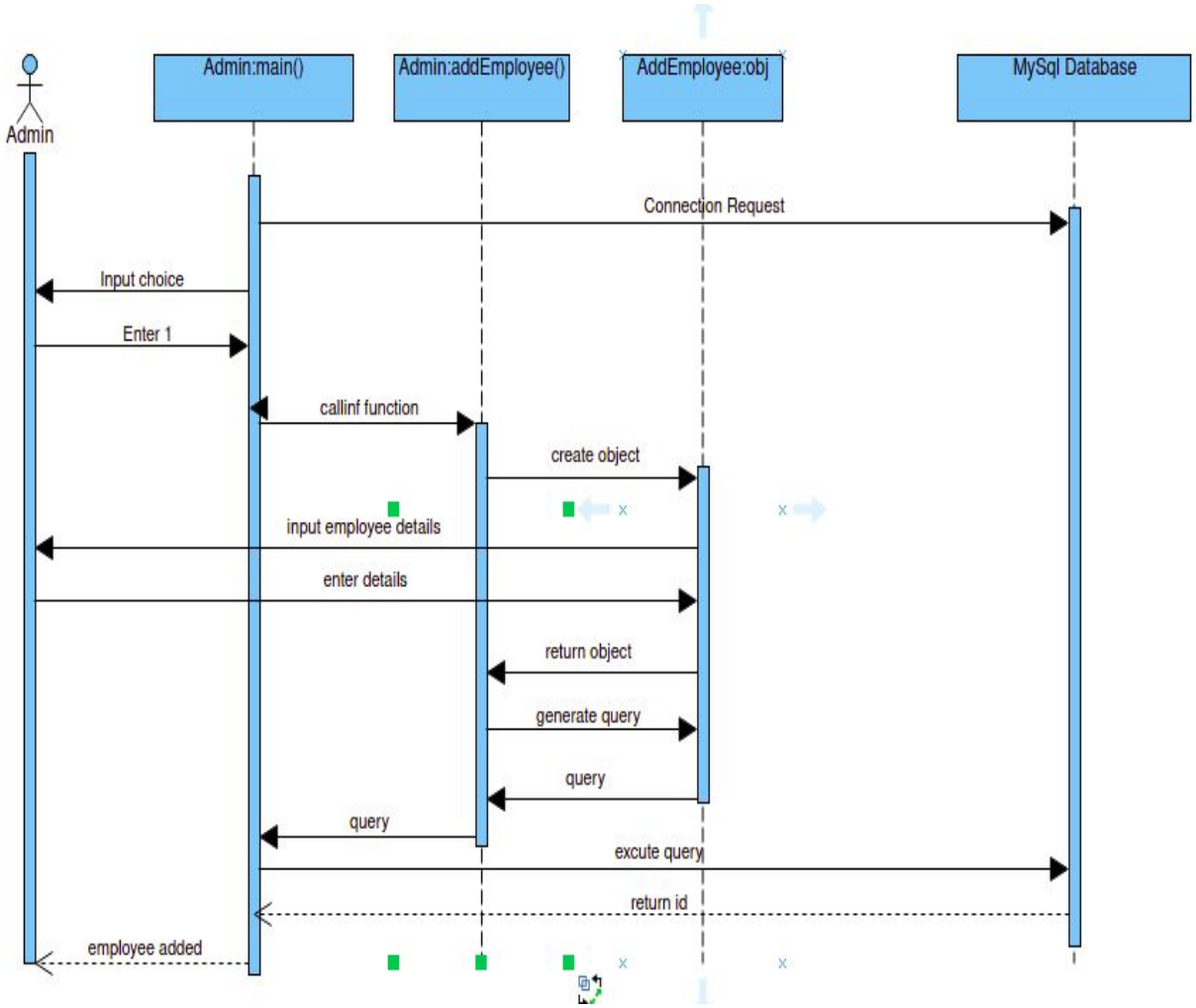
2. **DailyWork table:**

- This table stores information about type 1 employee (Hour Employee). How many hours they worked each day. A time card is stored in this table.
- Id primary key of table.
- empID foreign key which reference id of Employee
- Time: store the date
- Hours: store number of hours worked on date Time.

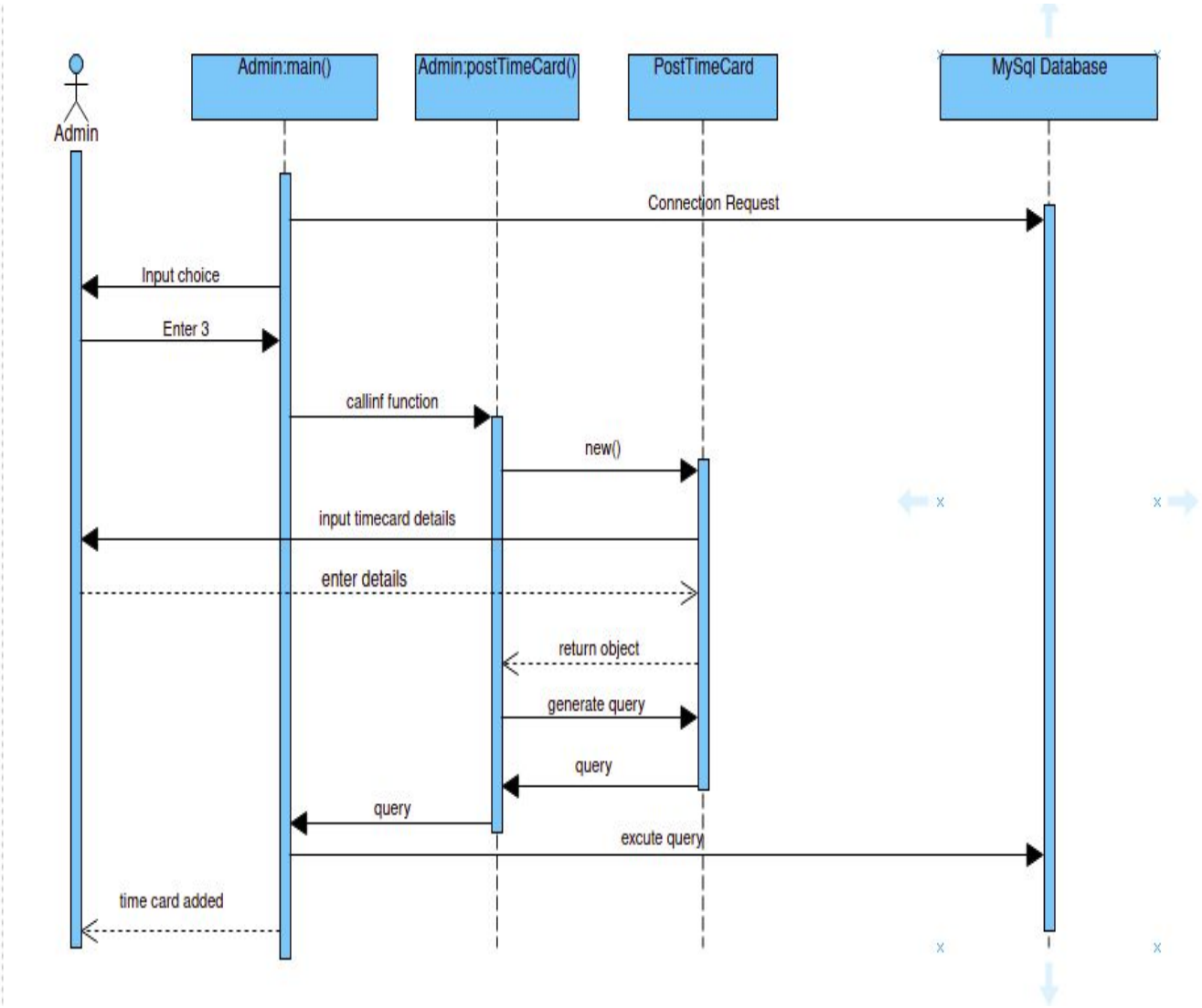
3. **Commission**
   - This table stores information about sales made by type 2 Employee (Salaried Employee). A sales receipt is stored in this table.
   - Id : primary key of table.
   - empID: foreign key referncing id of employee.
   - Place: to store organization name to which sales is made.
   - Amount: amount of sale done on which commission will be paid to Employee.

Sequence Diagram

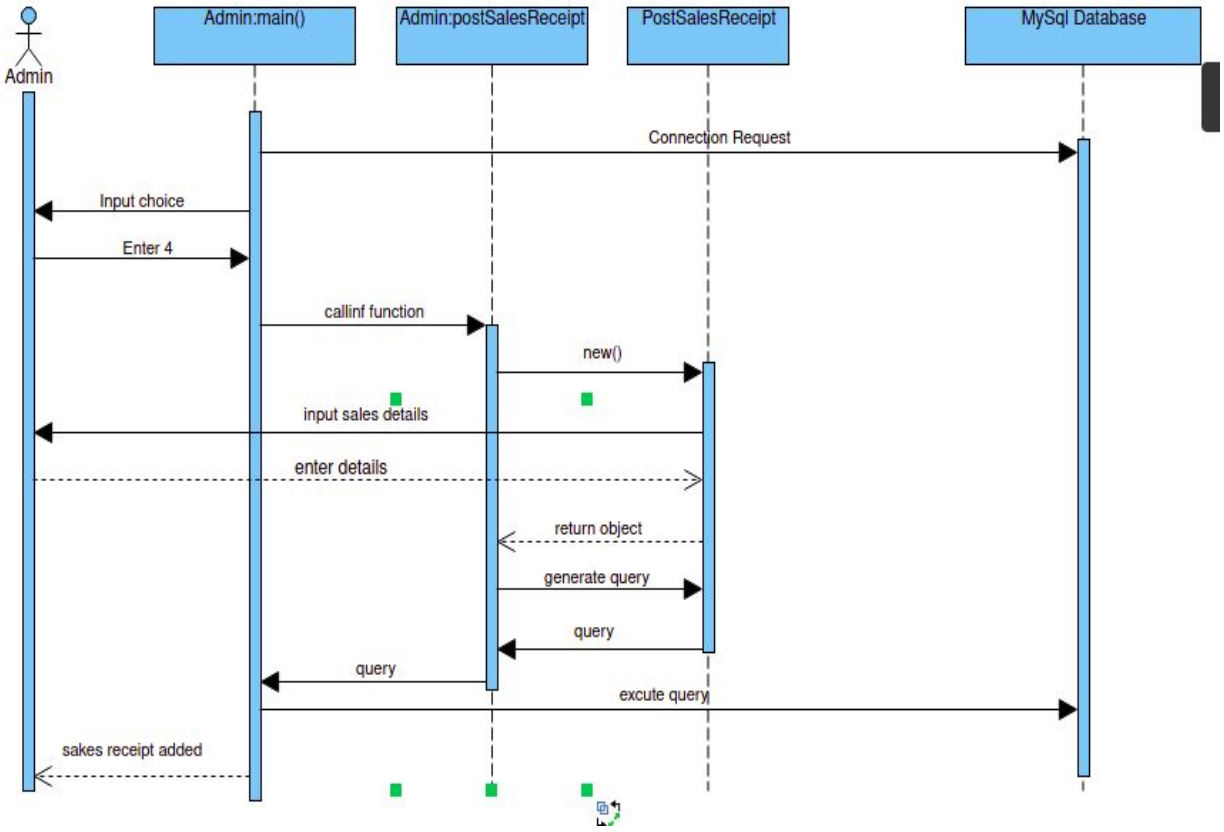
Add Employee


Admin
Admin:main()
Admin:addEmployee()
AddEmployee:obj
MySql Database
Connection Request
Input choice
Enter 1
callinf function
create object
input employee details
enter details
return object
generate query
query
query
excute query
return id
employee added

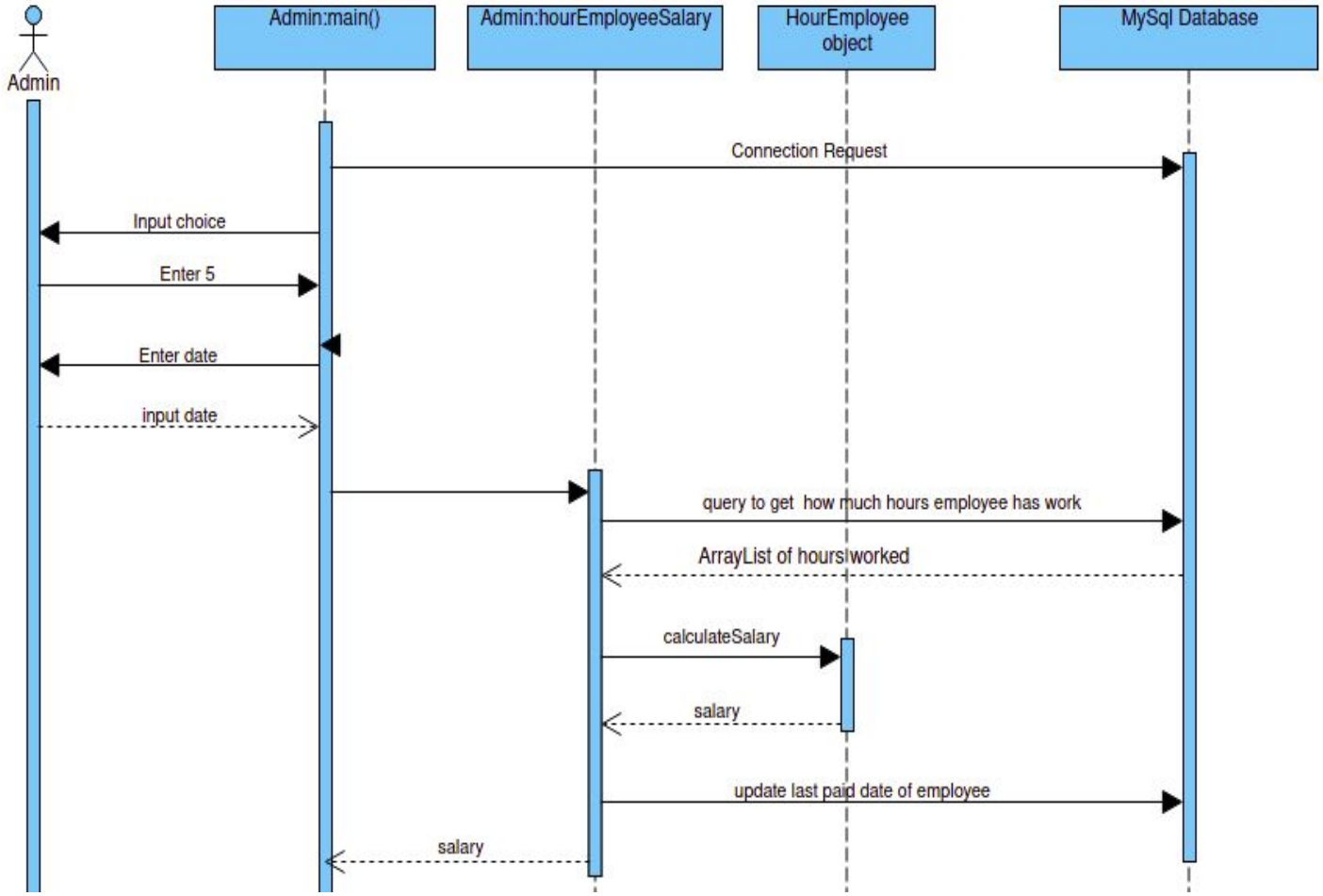
Post time Card



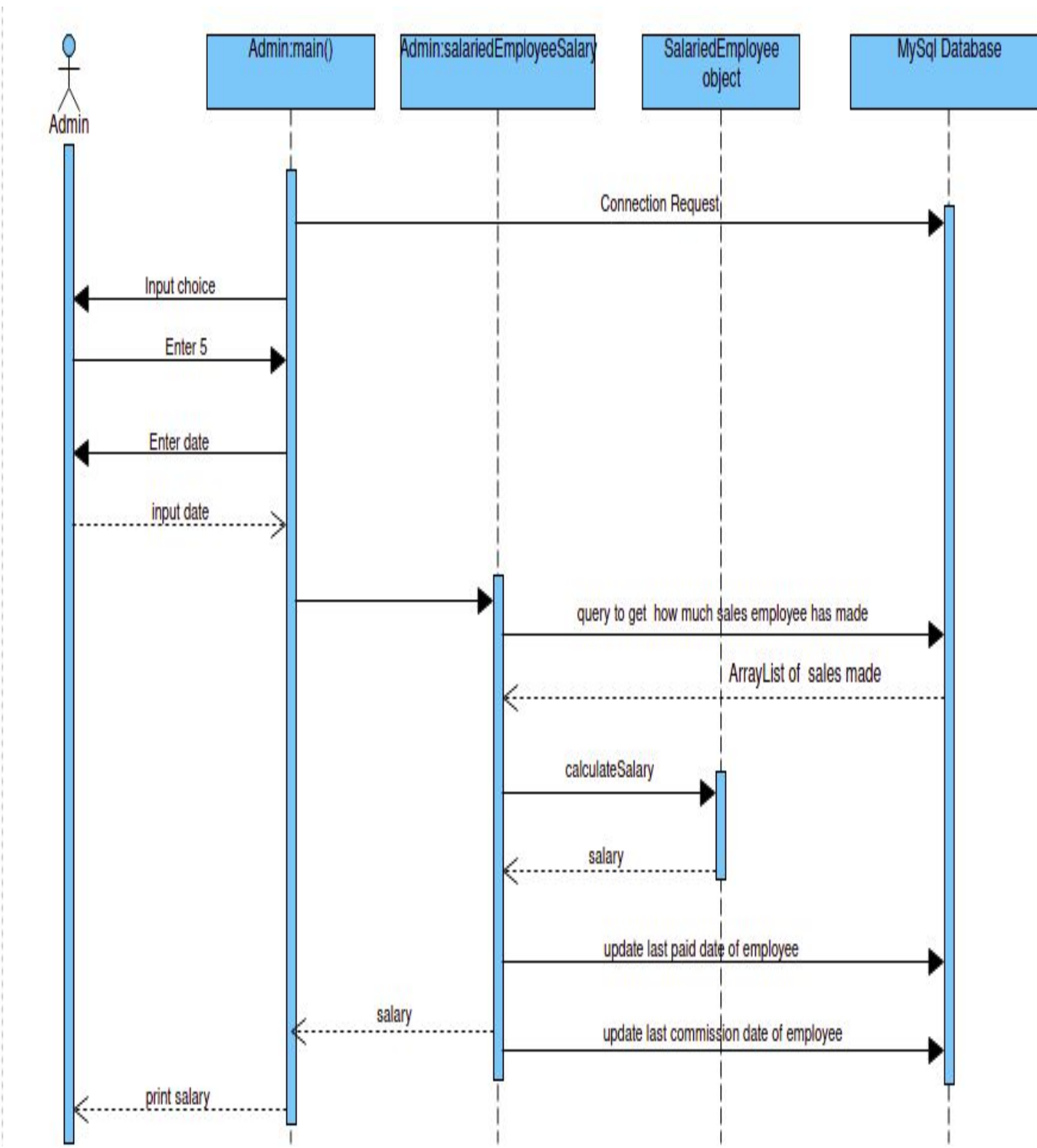
Post Sales Receipt



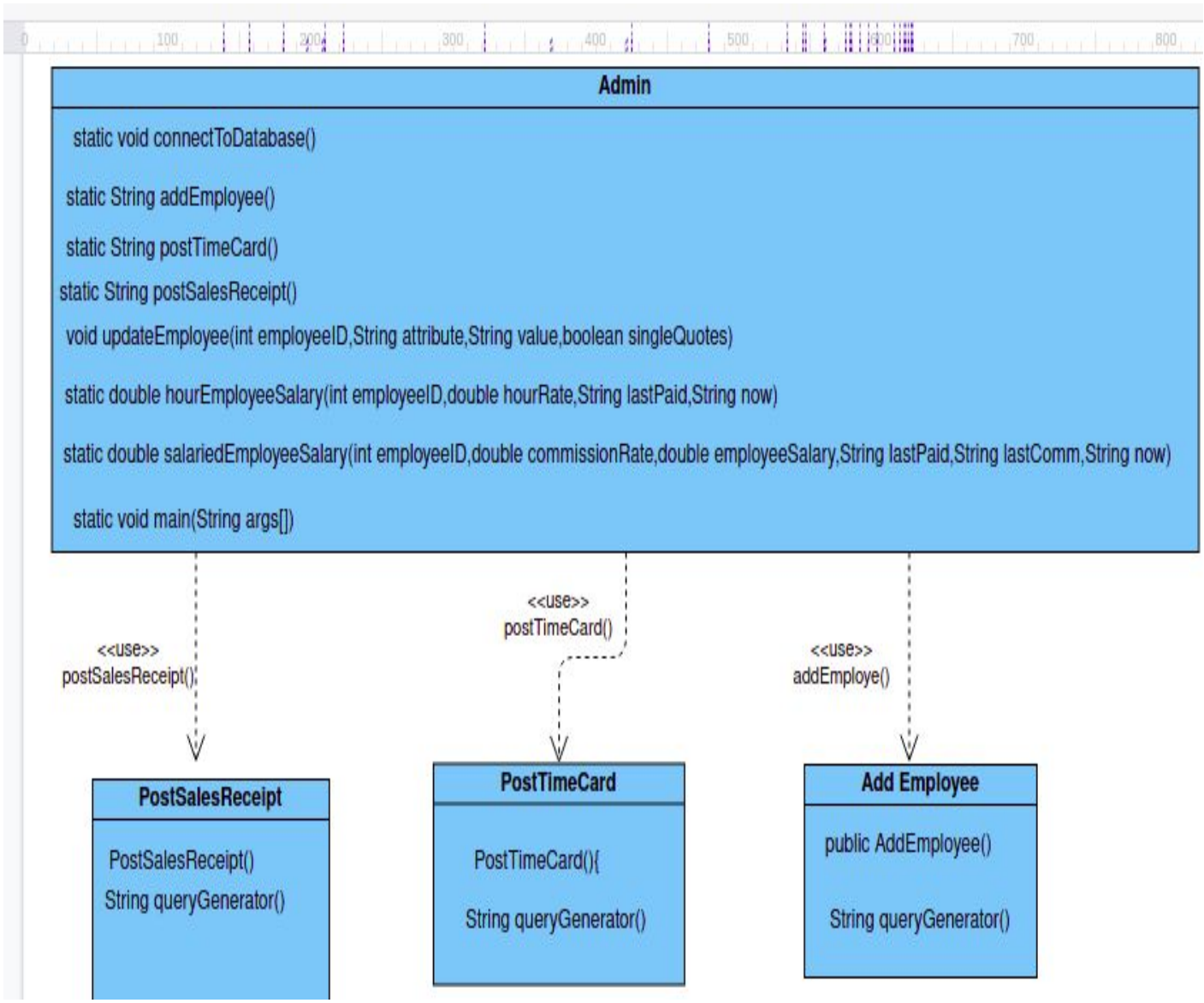
Print Type 1 employee salary

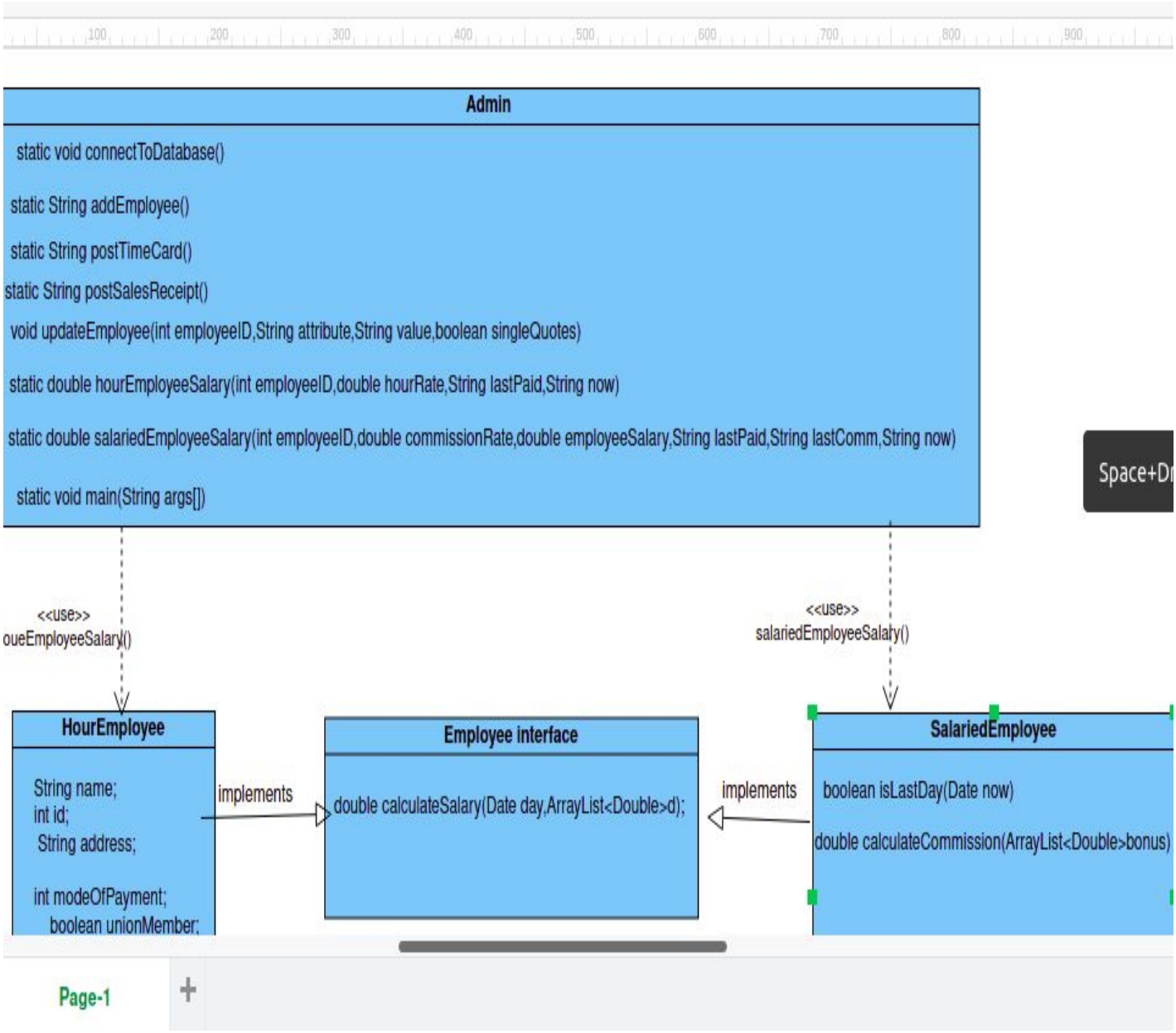


Print type 2 salary


Admin
Admin:main()
Admin:salariedEmployeeSalary
SalariedEmployee object
MySql Database
Connection Request
Input choice
Enter 5
Enter date
input date
query to get how much sales employee has made
ArrayList of sales made
calculateSalary
salary
update last paid date of employee
salary
update last commission date of employee
print salary

CLASS Diagrams


Admin
static void connectToDatabase()
static String addEmployee()
static String postTimeCard()
static String postSalesReceipt()
void updateEmployee(int employeeID,String attribute,String value,boolean singleQuotes)
static double hourEmployeeSalary(int employeeID,double hourRate,String lastPaid,String now)
static double salariedEmployeeSalary(int employeeID,double commissionRate,double employeeSalary,String lastPaid,String lastComm,String now)
static void main(String args[])
<<use>>
postSalesReceipt()
<<use>>
postTimeCard()
<<use>>
addEmploye()
PostSalesReceipt
PostSalesReceipt()
String queryGenerator()
PostTimeCard
PostTimeCard(){
String queryGenerator()
Add Employee
public AddEmployee()
String queryGenerator()


Admin
static void connectToDatabase()
static String addEmployee()
static String postTimeCard()
static String postSalesReceipt()
void updateEmployee(int employeeID,String attribute,String value,boolean singleQuotes)
static double hourEmployeeSalary(int employeeID,double hourRate,String lastPaid,String now)
static double salariedEmployeeSalary(int employeeID,double commissionRate,double employeeSalary,String lastPaid,String lastComm,String now)
static void main(String args[])
<<use>>
oueEmployeeSalary()
<<use>>
salariedEmployeeSalary()
HourEmployee
String name;
int id;
String address;
int modeOfPayment;
boolean unionMember;
implements
Employee interface
double calculateSalary(Date day,ArrayList<Double>d);
implements
SalariedEmployee
boolean isLastDay(Date now)
double calculateCommission(ArrayList<Double>bonus)
Page-1

Future Scope

- This project implementation is more backend than object orientation.
- Change implementation to more object orientation.
- There are 2 layers right now. Language and Database.
- Introduce middle layer of JSon so that language implementation(frontend) and backend (database) can be easily separated. (Learning JSON right now).
- In this project first I design ER diagram and then implements my java classes based on that . That’s why object orientation is less used i.e number of classes are less.
- Redesign with objects definition first then database at last step is a good way of designing systems.