Designing A Payroll System

A Solution, NOT THE Solution

Daniel POP, Ph.D

General Description

- The system consists of a database with all company’s employees, and their associated data, such as time cards.
- The system must pay all employees the correct amount, on time, by the method that they specify.
- Also, various deductions must be taken from their pay.

General Description

Specifications / requirements

- Some employees work by the hour. They are paid an hourly rate that is one of the fields in their employee record. They submit daily time cards that record the date and the number of hours worked. If they work more than 8 hours per day, they are paid 1.5 times their normal rate for those extra hours. They are paid every Friday.
- Some employees are paid a flat salary. They are paid on the last working day of the month. Their monthly salary is one of the fields in their employee record.
- Some of the salaried employees are also paid a commission based on their sales. They submit sales receipts that record the date and the amount of the sale. Their commission rate is a field in their employee record. They are paid every other Friday.
- Employees can select their method of payment. They may have their paychecks mailed to the postal address of their choice, have their paychecks held by the paymaster for pickup, or request that their paychecks be directly deposited into the bank account of their choice.
- Some dues will be deducted from employee’s pay amount, such as taxes, health contribution, union etc. These charges must be deducted from the appropriate employee’s next pay amount. They can be flat rates or percentages applied to gross pay amount.
- The payroll application will run once each working day and pay the appropriate employees on that day. The system will be told what date the employees are to be paid to, so it will generate payments for records from the last time the employee was paid up to the specified date.

Analysis by Use Case

- It is the system's behavior what we are creating/modelling, not the system's data; system's data is a consequence

- **Use Cases (UC)** are ways to capture system's behavior; they are similar to user stories (in agile methodologies)
- For our system, the UCs, what customer wants, are:
  1. Add a new employee
  2. Delete an employee
  3. Post a time card
  4. Post a sales receipt
  5. Post a charge
  6. Change employee details (e.g., hourly rate, dues rate, etc.)
  7. Run the payroll for today

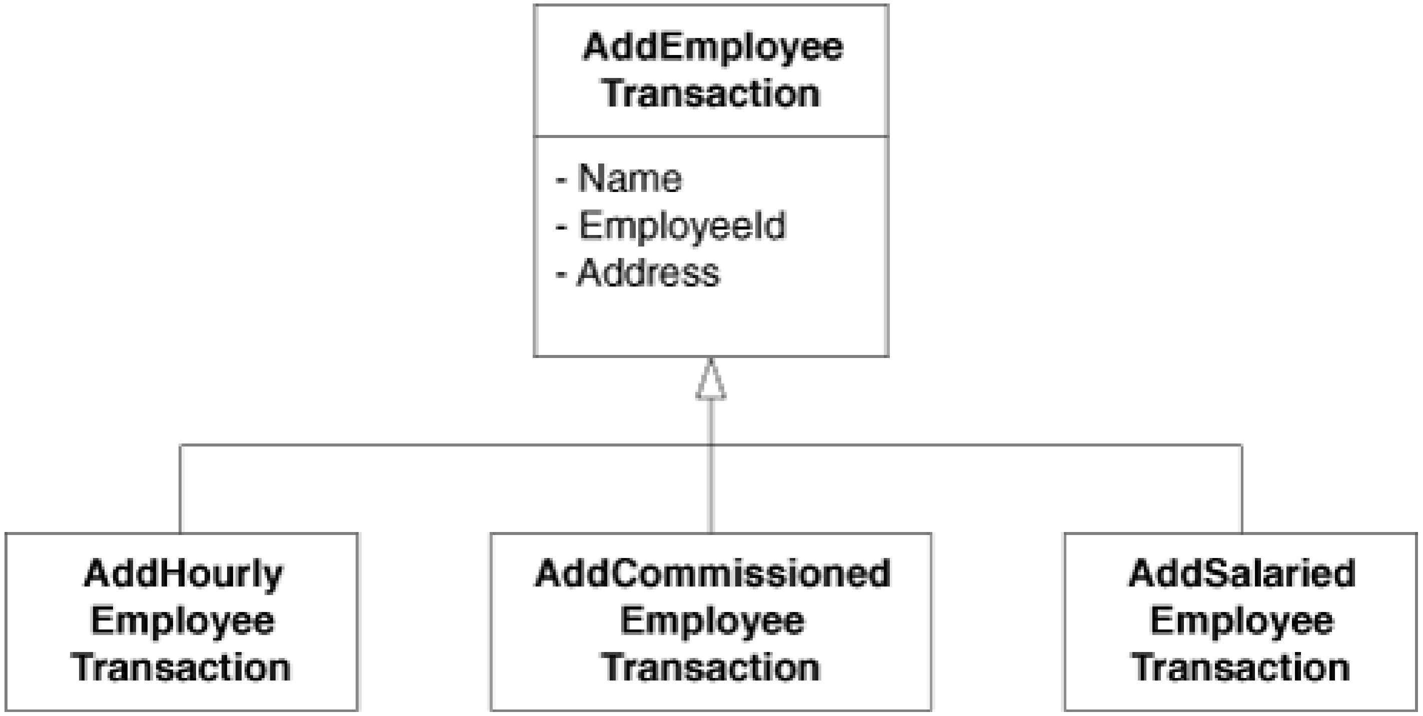
UC #1: Add a New Employee
Description

- A new employee is added by the receipt of an `AddEmp` command. This command contains the employee’s name, address, and assigned employee number. The command has three forms:
  - `AddEmp <EmpID> "<name>" "<address>" H <hrly-rate>`
  - `AddEmp <EmpID> "<name>" "<address>" S <mtly-slry>`
  - `AddEmp <EmpID> "<name>" "<address>" C <mtly-slry> <com-rate>`
- The employee record is created with its fields assigned appropriately
- If the command structure is inappropriate, it is printed out an error message, and no action is taken.

UC #1: Add a New Employee

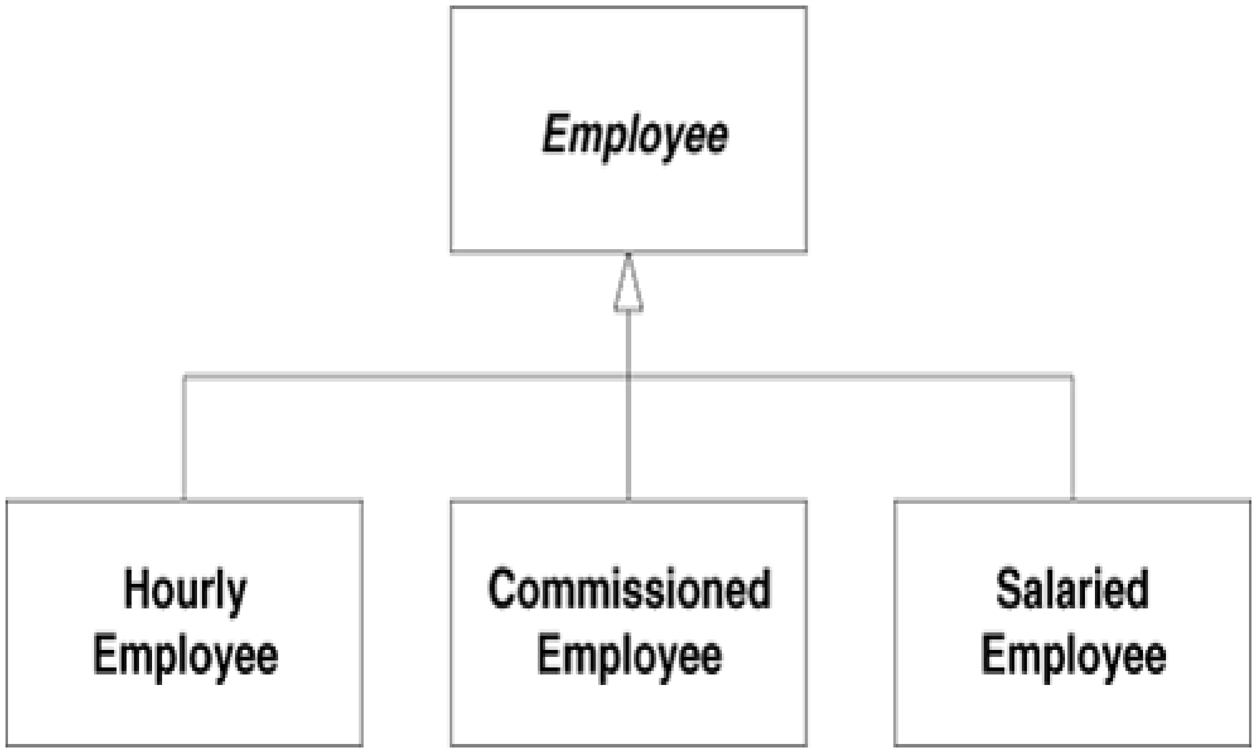
Inferred model

- Use Command pattern to model each possible type of command
- Enforces SRP


AddEmployee Transaction
- Name
- EmployeeId
- Address
AddHourly Employee Transaction
AddCommissioned Employee Transaction
AddSalaried Employee Transaction

UC #1: Add a New Employee
Inferred model

- Q: What do the three commands create?
- A: They create **three kinds** of employee objects



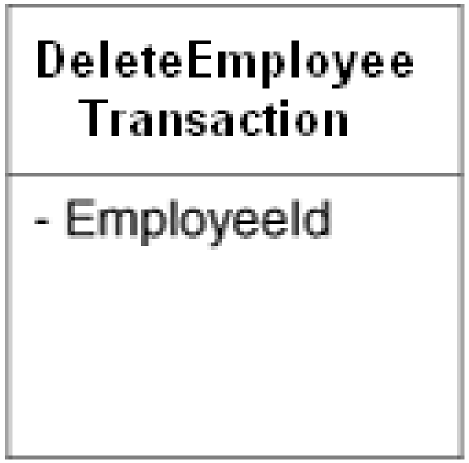
UC #2: Delete an Existing Employee
Description

- Employees are deleted when a DelEmp command is received. The form of this command is as follows :
  - `DelEmp <EmpID>`

- When this command is received, the appropriate employee record is deleted.
- If the <EmpID> field is not structured correctly or does not refer to a valid employee record, the command is printed with an error message, and no other action is taken.

UC #2: Delete an Existing Employee

Inferred model



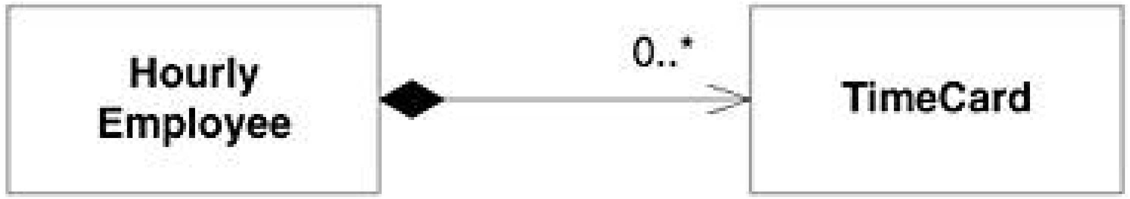
UC #3: Post a Time Card

Description

- On receipt of a TimeCard command, the system will create a time card record and associate it with the appropriate employee record:
  - `PostTimeCard <empid> <date> <hours>`
- If the <EmpID> field is not structured correctly or does not refer to a valid employee record, the system will print an appropriate error message and take no further action.

UC #3: Post a Time Card

Inferred model



UC #4: Post a Sales Receipt

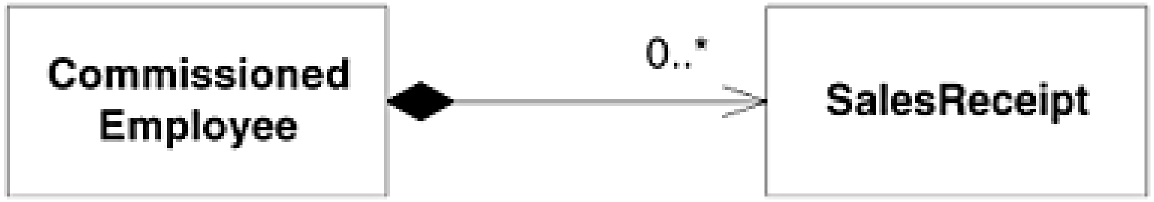
Description

- On receipt of the SalesReceipt command, the system will create a new salesreceipt record and associate it with the appropriate commissioned employee:

  - `PostSalesReceipt <empid> <date> <amount>`

- If the <EmpID> field is not structured correctly or does not refer to a valid employee record, the system will print an appropriate error message and take no further action.

UC #4: Post a Sales Receipt
Inferred model

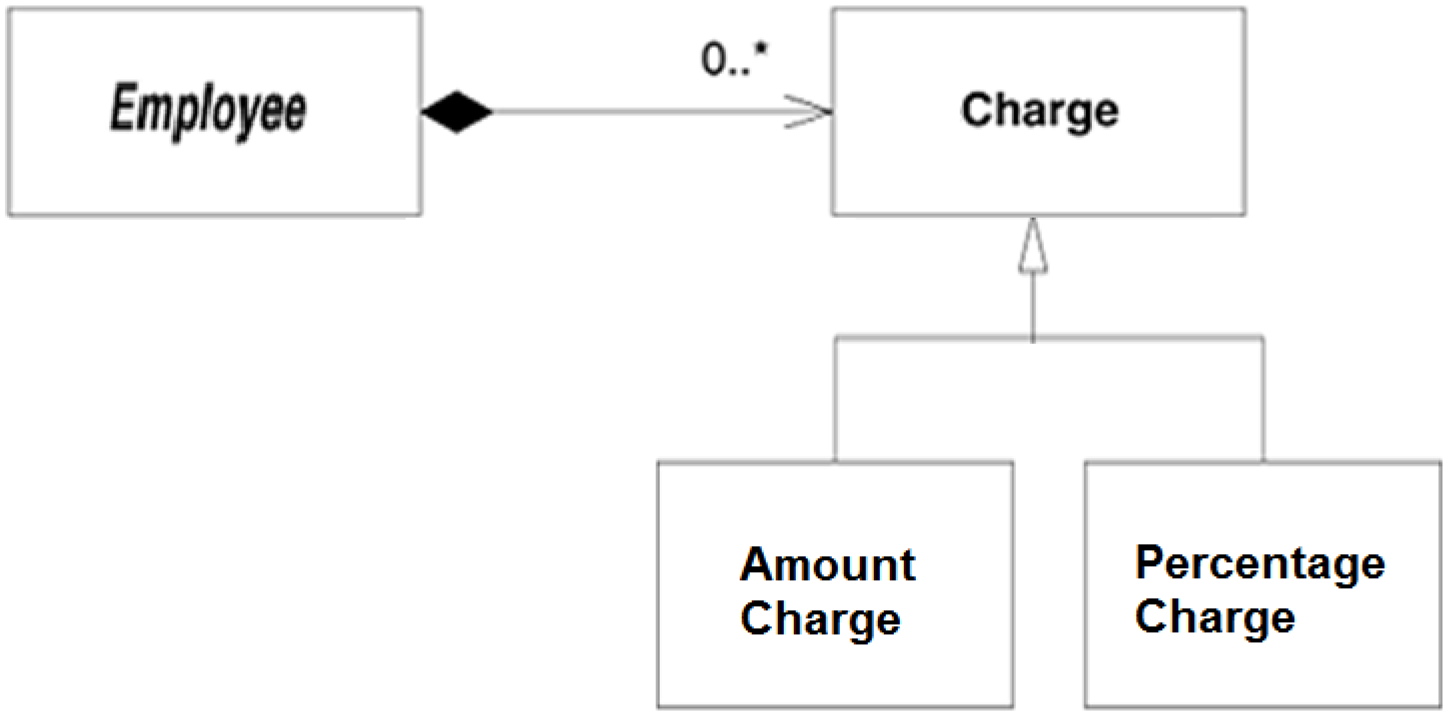


UC #5: Post a Charge

Description

- Charge may be pension contribution, health system contribution, union contribution etc.
- On receipt of this command, the system will create a charge record and associate it with the appropriate employee:
  - `AddCharge <empID> <chargeID> <amount>`
  - `AddCharge <empID> <chargeID> <percentage>`
- If the command is not well formed or if the <empID> does not refer to an existing employee, the command is printed with an appropriate error message.

UC #5: Post a Charge
Inferred model



UC #6: Change Employee Details
Description

- Upon receipt of this command, the system will alter one of the details of the appropriate employee record. There are several possible variations to this command:
  - `ChgEmp <EmpID> Name <name> – Change employee name`
  - `ChgEmp <EmpID> Address <address> – Change employee address`
  - `ChgEmp <EmpID> Hourly <hourlyRate> – Change to hourly`
  - `ChgEmp <EmpID> Salaried <salary> – Change to salaried`
  - `ChgEmp <EmpID> Commissioned <salary> <rate> – Change to commissioned`
  - `ChgEmp <EmpID> Hold – Hold paycheck`
  - `ChgEmp <EmpID> Direct <bank> <account> – Direct deposit`
  - `ChgEmp <EmpID> Mail <address> – Mail paycheck`
  - `ChgEmp <EmpID> Charge <chargeID> <amount> – Change a fixed charge`
  - `ChgEmp <EmpID> Charge <chargeID> <perc> – Change a percentage charge`
- If the structure of the command is improper, <EmpID> does not refer to a real employee, the system will print a suitable error and take no further action.

UC #6: Change Employee Details
Inferred model

Lot of information for one UC:

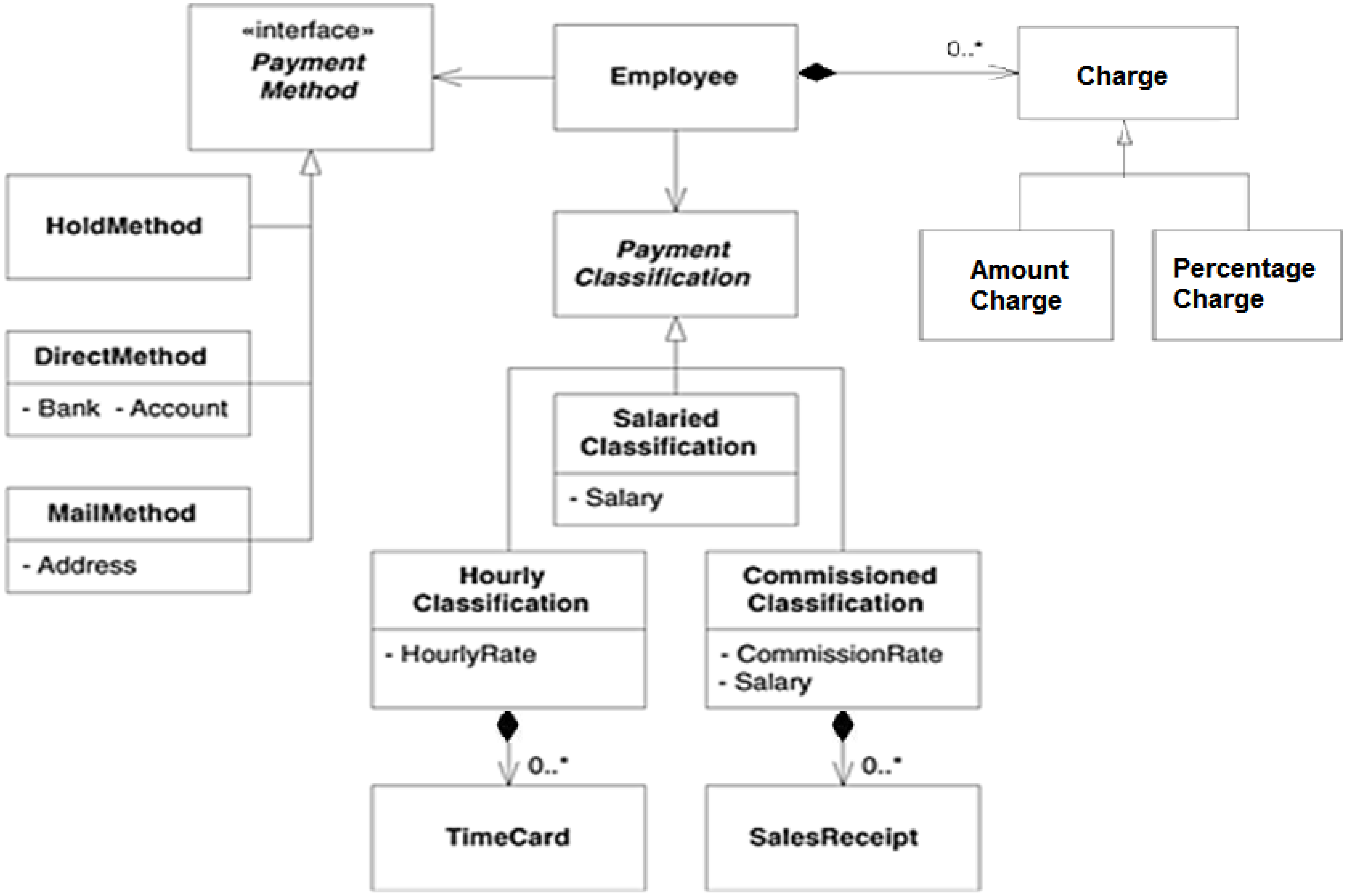
1. The fact that the type of employee is changeable implies that the diagram inferred from UC #1 is invalid; the inheritance is not appropriate => need to use object composition instead; Strategy pattern is a good candidate for pay computation

2. The same approach can be used for method of payment (it is also changeable)

- All these patterns enforces OCP principle in our design

UC #6: Change Employee Details

Inferred model



UC #7: Run The Payroll For Today

Description

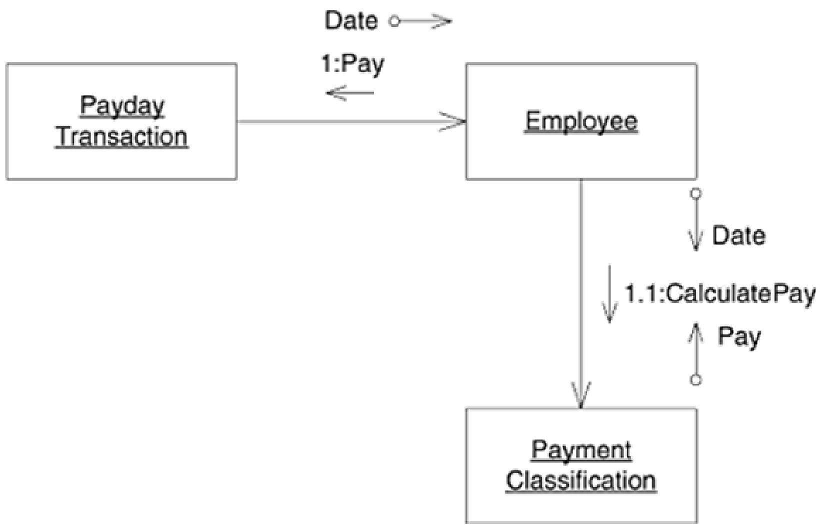
- On receipt of the payday command, the system finds all those employees that should be paid on the specified date. The system then determines how much they are owed and pays them according to their selected payment method. An audit-trail report is printed showing the action taken for each employee:

    - `Payday <date>`

UC #7: Run The Payroll For Today

Calculating the pay

- First, how does the Employee object know how to calculate its pay?

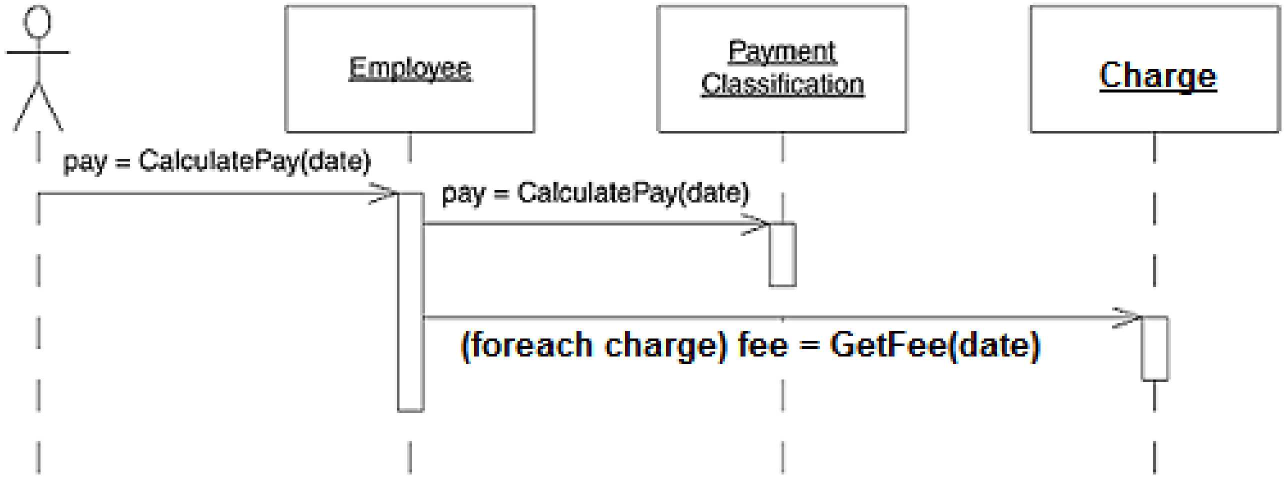


UC #7: Run The Payroll For Today

Calculating net / gross salaries

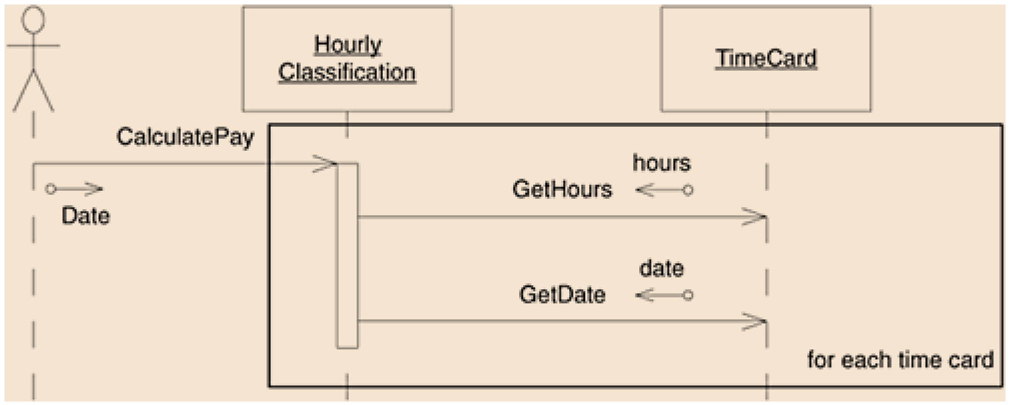
NetSalary = GrossSalary – Deductions

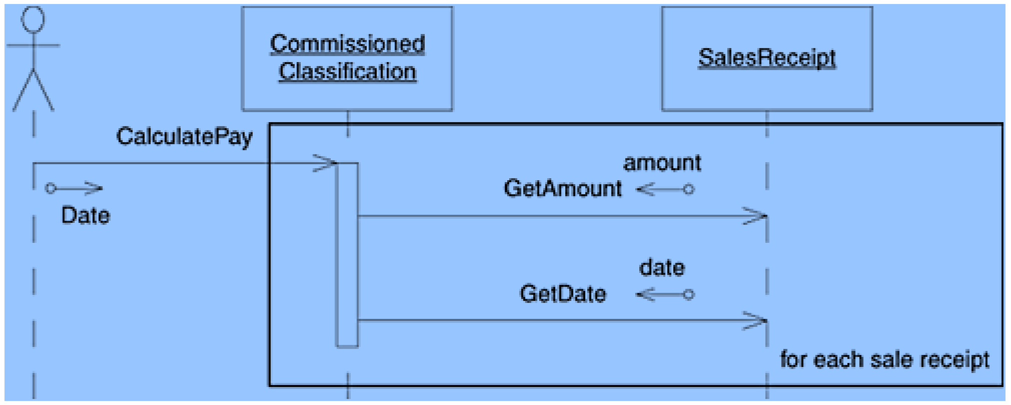
Deductions = Sum of all charges

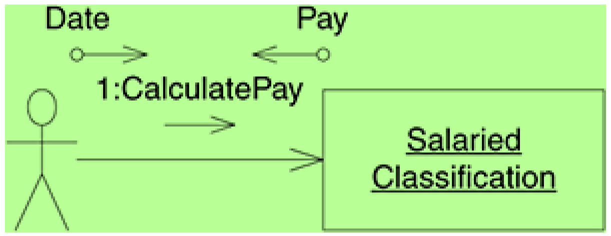


UC #7: Run The Payroll For Today

Calculating gross salary







- The system will sum up an Hourly employee's time cards and multiply by the hourly rate.
- Similarly, the system will sum up a Commissioned employee's sales receipts, multiply by the commission rate, and add the base salary.
- For Salaried employee, the system will pay the fixed monthly amount

UC #7: Run The Payroll For Today

Digging ‘hidden’ abstractions

- Some abstractions are easy to find; for example, from “Some employees work by the hour” and “Some employees are paid a flat salary” we derived the PaymentClassification abstraction.

- Some others are “hidden”; for example, from “They are paid every Friday,” “They are paid on the last working day of the month,”, or “They are paid every other Friday.” we may derive the notion of Schedule of Payment, that is not specifically mentioned in requirement!

UC #7: Run The Payroll For Today

Dealing with payment schedule

- We can wire the payment schedule (= whether a day is a pay day or not) into PaymentClassification classes

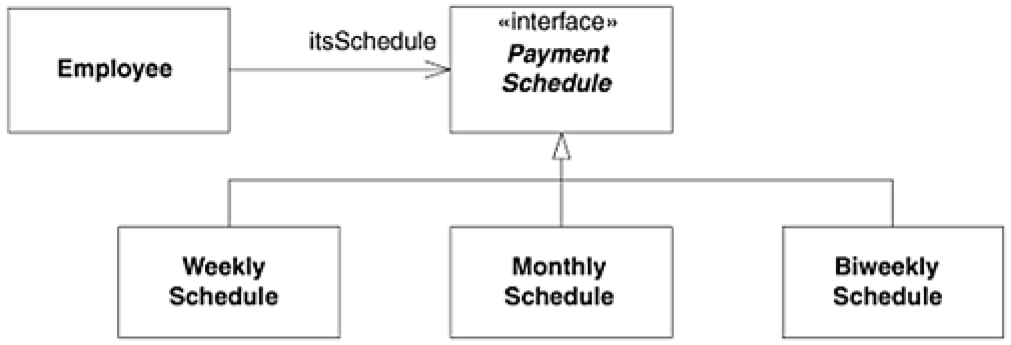
- …but this is will tight couple 2 unrelated concepts:
  - Salary computation algorithm
  - Payment schedule algorithm

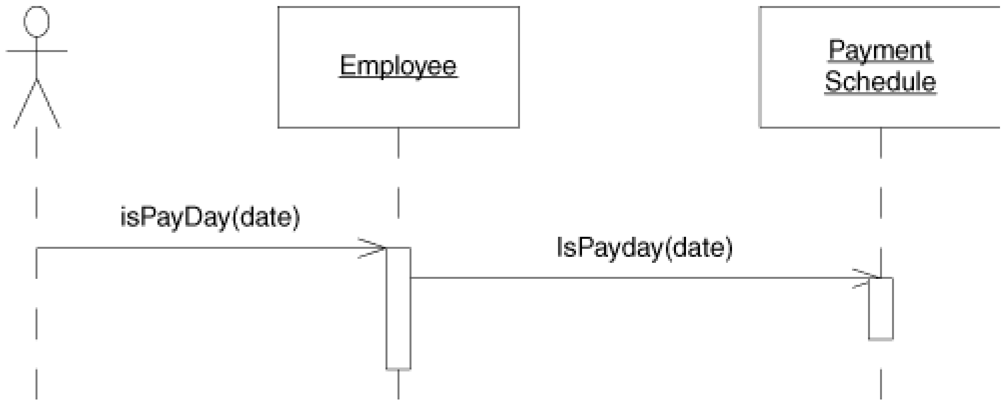
breaking down several OOD principles (SRP, OCP, DRY).

UC #7: Run The Payroll For Today

Dealing with payment schedule, a flexible approach

- Instead, we'll do like this



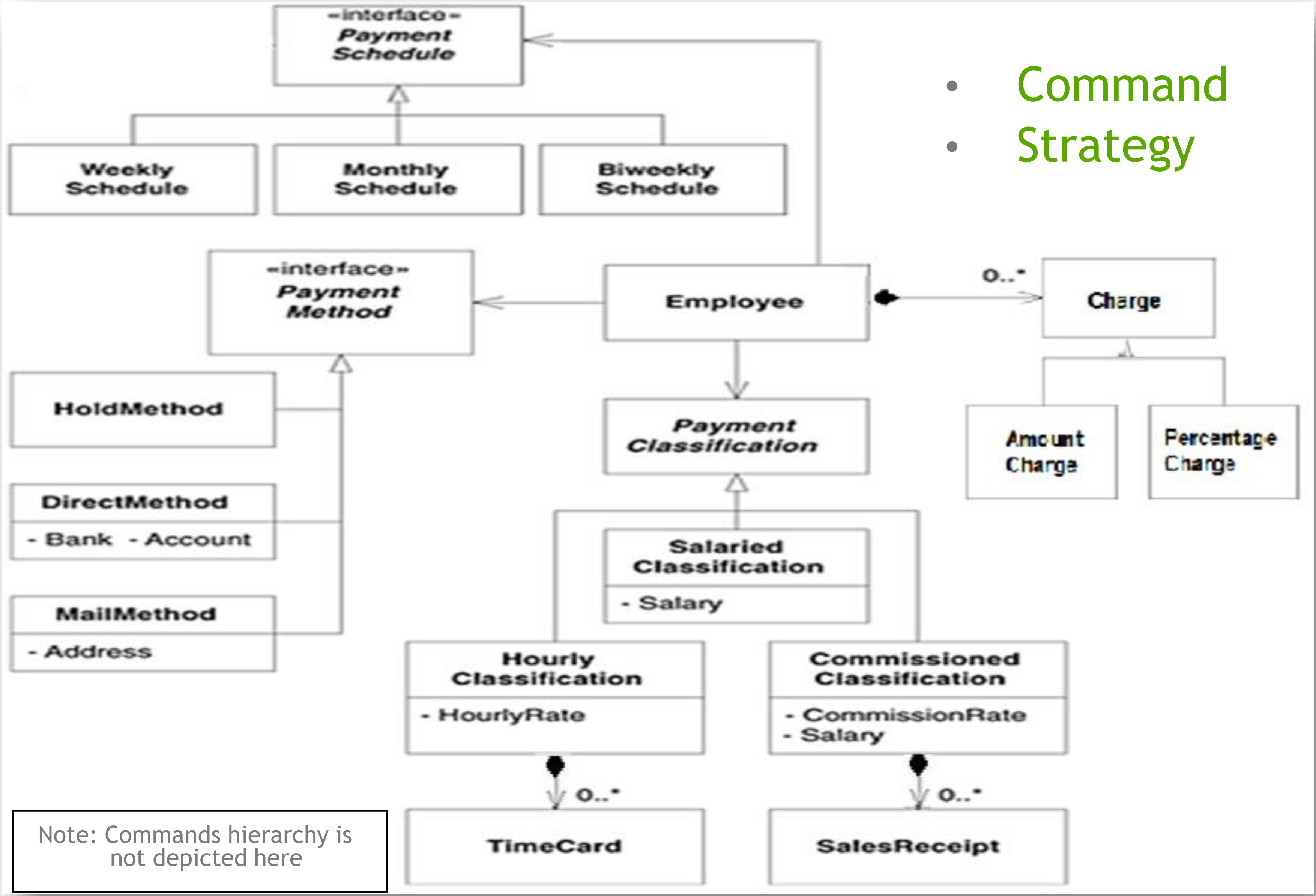


- What pattern did we use?

- Easily to combine any payment schedule with any salary computation algorithm
- Independent evolution of the 2 concepts (add, change, remove)
- Comply with OOD principles

Summary

Final Design



- Command
- Strategy

Note: Commands hierarchy is not depicted here

Summary
Java Sources

- This example has been (partially) implemented in Java
- Download the Java sources and binaries from the following address:

  http://web.info.uvt.ro/~danielpop/dp/payroll.zip

- To run the example, extract the zip file and execute run.bat file

Overreliance on tools and procedures and underreliance on intelligence and experience are recipes for disaster.