Python Programming

Company Payroll Project

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Big Picture

- The goal of this project is **high-level design** NOT full-implementation
  - There is no menu of options. Don’t provide deep details. Goal is design
  - Also another goal is to deal with requirements that might seems **vague**
- To make it easy, the project is provided in 4 parts
  - After each part, it is recommended to understand my solution and build-over it for the next part

Part 1: Employees

- We need to represent a company and its payroll:
  - Working people: either volunteers or employees
    - The minimal common between them is name and address information
    - Any working person is paid money based on its type
    - Each employee is paid in a specific day (varying from a person to another)
  - Employees could be hourly based or salaried based.
    - Also a commision salaried employee takes extra money as the ratio of the sales s/he did
- Develop the classes (high-level)
  - **Features** for each class type you figure out
  - **amount_to_pay** property that returns the salary

Part 2: Invoices & Payroll

- Let’s extend the system
- There are invoices: each invoice has set of items (e.g. books, food, etc)
  - Each item has description, total quantity and price per item
  - Each item has its own details (e.g. book author name)
  - Invoice price: sum of the items’ prices
- The payroll consists of a payables list
  - Each payable is either employee or invoice
  - The total paid money is the total paid money for the added employees and invoices
- Create class Company
  - It creates several types of payables, add to Payroll and compute total paid money

Part 3: Validations

- Background: In practice, we may need to verify invoices.
  - E.g. an invoice is out-of-date relative to some deals with suppliers
  - E.g. the computed taxes doesn't utilize some advantages in the country
- We can create several **validation rules**, one for every purpose
  - In future, more rules might be added
  - *You don't need to care about the validation rules logic*
- A Validator Group consists of a **subset** of the available validation rules
  - E.g. Mandatory-Validator has a very few validation rules to be fast in evaluation
  - E.g. Complete-Validator has all the validation rules coded so far
  - In short: the validator just make sure all its rules are valid
- Before paying for an invoice, it must pass all validation rules in its validator group

Part 4: Payroll Printing

- We would like to be able to print the payroll content (repr), however we would like them to be ordered
- Ordering Criteria
  - If the object type is not the same (e.g.invoice vs HourlyEmployee / SalariedEmployee vs CommissionSalariedEmployee) ⇒ Compare based on class name string
  - If they are the same:
    - If it is a working person, then compare based on name then salary (increasing)
    - If it is an invoice: then compare based on invoice ID, then # of items in an invoice
- Printing
  - Employee ⇒ Class name, Employee Name, Employee Address
  - Invoice ⇒ Class name, ID, # of items

Beyond

- Feel free to go for full implementation if you are done with design scope
  - Menu, Specific Defined Exceptions, Save/Load with Disk

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."