Entity

Interface **Userable**	**User**
• Define abstract methods for classes that implement this interface.	Account_manager

Interface

Payable

● Responsibilities (In progress) ● Plan : an interface for the future uses case.	Collaborators(In progress) ● Pay_Manager(In progress)

HR system CRC model

User	Userable
• Store the name, id, account information, password, record of vaccination, address, phone number of the employee • Getters for everything above • Setters for everything above • Equal based on the id of the user • Print the user's identity and name	Account_manager

Abstract

Employee

FullTimeEmployee, PartTimeEmployee

- Store the wage, attendance, department, and level of the employee
- Getters for everything above
- Setters for everything above
- Print the department of the employee

Account_manager

FullTimeEmployee	**Employee**
● Store the total vacation with salaries and the vacation used that the full-time employee has ● Getters for everything above ● Setters for everything above ● Print the information of the full time employees' department, position, and level in strings.	

PartTimeEmployee	Employee
● Store the schedule of the part-time employee ● Getter for schedule above ● Setter for schedule above ● Print the information of the part time employees' department, position, and level in strings.	

Uses Case

AccountManager

- Store a Hashmap called **employee list** that include the userable as the key and employee as the value.
- Counter for recording the IDs that have been issued
- Store and count the total number of employees
- Store and count the total number of part-time employees
- Store and count the total number of full-time employees
- Override the toString method (i.e Account_manager)
- Create new user
- Delete user
- Get total number of employees
- Get total number of part-time employees
- Get total number of full-time employees

- Userable
- Employee
- Verifier

Verifier	
● Store a Hashmap called **employee list** that include the Userable type object as the key and employee as the value. ● Verify whether the account exists ● Verify whether the account has the level (i.e: authority) ● Print “this is the verifier” ● Verify the account existence ● Verify the level (authority)	● AccountManager ● Admin_System

Pay_Manager	
Responsibilities (In progress) ● Calculate the wage of the employees.	Collaborators(In progress) Admin_System

Interface Adapter

Admin_System	
Responsibilities (In progress) ● Store all the manager classes (I.e: Account_Manager, Pay_Manager) ● Controller for the manager to operate on the use cases	Collaborators (In progress) ● Command_User_Interface ● AccountManager ● Pay_Manager (In progress) ● Verifier

Employee_System	
Responsibilities (In progress)	Collaborators(In progress)

Framework

Command_User_Interface	
Responsibilities (In progress) ● Provides a command line interface to user for accessing the systems.	Collaborators(In progress) Admin_System Employee_System Pay_Manager(In progress)

Question

HR system ideas

- Calculates wages according to the position of worker.
- Records attendance and absences
- Distributes daily tasks and works to workers
- Distributes project and programs to teams and groups
- Calculates KPI for each workers
- Address vacation application

Some detailed ideas

- Authority → attributes (every worker have this attribute in their class)
  - High level workers (managers) have a method in their class called setAuthority which can set the authority of lower level workers.
  - Low level workers do not have this method in their class

Some Questions

- Is an user abstract class necessary?
- Hour to work
- Management ---> How to prom..
- Review a task