Task 1:HR System
Task 2: Specification

Running the project allows users to first interact with an empty HR System. While running, the HR System will prompt a message saying please enter what you want to do indicating that users can type in commands that create a new worker or department head or change the worker's salary or schedule. All commands are case sensitive and the order of information needed must be the same.

When the user type exit, then the program will quit itself.

When the user type create worker followed by its name, salary, department and schedule separated by a single space, then the program will create a worker with the same attribute. Passing in the schedule needs to start with the day of the week that the work works on followed by the start time (time combined with AM or PM without space)and then the end time (same format as start time) separated by space. Here you are only allowed to enter a single day for the schedule. After the worker is created, the program will display the worker with name, id, salary and department is created. Note here that the user must create a department head before they create worker for that department otherwise, the program will return no department head for the department and the program will not create a worker.

When the user type create department head followed by its name, department, the program will create a department head with the same attribute.
1. If there is already a head for the department, then the program will give a message that says there is already a department head for the department.
2. If the department head is created successfully, then the program will display department head with name, id and department is created.

When the user type change salary followed by the id of the worker, the id of the department head that wants to change the salary and the percent that should be changed by as a decimal where increase salary is a positive decimal and decrease salary is a negative decimal.
1. If the department head does not have a worker with the given id, then the program will display that the department head has no right to change salary for the worker.
2. If the department head id entered doesn't match any department head, then the program will display that department head not found.
3. If everything is correct, then the program will change the worker's salary and display the salary of worker name with id has been changed to current salary by permission from the name of the department head.

When the user type change schedule followed by worker id, department head id, and the new schedule in the format mentioned above, then the program will display

1. no department head found if the department head id doesn't match to a department head
2. no right to change schedule when the department head does not have a worker with the given id

3. schedule changed for worker name, id, in its department with current schedule on day of the week from start time to end time if the department head and worker are valid and in the same department.

The way IDs are generated is that for worker and department heads, IDs start with 0 and increase by 1 when the user creates one more worker or department head which keeps the ID unique.

The four entity classes are worker, departmentHead, employee and schedule (worker and departmentHead extends employee) with 1 parent class called employee which is the parent class of worker and departmentHead. The 2 use case classes are workerManager and departmentHeadManager. Then we have an InOutHandler which is the controller (takes in input and calls a use case class method with the input) and the presenter (displays output to UI) together (this only works since the program is still fairly small). Finally, we will have a class called HRSystem which connects the whole program together. It keeps track of workers and department heads created and generate unique ids and make sure the department entered for the department head does not have duplicates. It also displays a welcome and ending message.

Task 3: CRC model

Entity Classes:

Class name: employee (parent class for worker and departmentHead)	
Responsibilities Stores 3 attributes about employees which are name, id and department. Provide the basic setter and getter for these new attributes	**Collaborators**

Class name: worker (child class of employee)	
Responsibilities Stores the same information as employee with 2 extra information which are salary and schedule Provide the basic setter and getter for these new attributes	**Collaborators** Schedule workerManager

Class name: departmentHead (child class of employee)

Responsibilities Stores the same information as employee with 1 extra information which is workerInDept which stands for worker in the department which is an arraylist that stores all worker under the department of the department head Provide the basic setter and getter for the attributes	**Collaborators** workerManager departmentHeadManager

Class name: schedule	
Responsibilities Stores information about worker schedule like week of the day, start time and end time Provide basic getter and setter for these attributes and overridden the toString function to better display schedule	**Collaborators** Worker workerManager

Use Case Classes:

Class name: workerManager	
Responsibilities Create new workers, change (increase enter positive decimal and decrease enter negative decimal) worker's salary with permission from department head and change work's schedule with permission from department head return the message (for changeSalary and changeSchedule since it doesn't change any information in HRSystem) that the user should see to the controller or return a map of one integer and an arraylist of department head which should be returned further to the HRSystem that indicates whether the HRSystem should increase the next available worker ID or not (1 being should be increased and 0 being not changing).	**Collaborators** Worker departmentHead schedule InOutHandler

Class name: departmentHeadManager	
Responsibilities Create new department heads and return a map of one integer and an arraylist of department head to indicate whether the list of all department head created stored in HRSystem should be modified (add the new department head created). The integer takes a value of 1 if the list should be modified by setting it to the arraylist that is the value of the key 1, and the integer takes a value of 0 if the department head creation fail and nothing should be changed.	**Collaborators** departmentHead InOutHandler

Controller and Presenter:

Class name: InOutHandler	
Responsibilities Take in input from UI and pass information into the correct use case method based on the input and display the result on the UI The controller should print either the creation of worker or department head is successful or not based on the integer in the returned map (1 being successful and 0 being not) Return a map that only contains an integer as the key and an arraylist consist of departmentHead as the value to HRSystem to tell the HRSystem whether user entered exit (key is -1) or whether HRSystem need to update information in the system (key is 1) or do nothing (key is 0)	**Collaborators** workerManager departmentHeadManager HRSystem

Frameworks and Drivers

Class name: HRSystem	
Responsibilities Stores all the department heads created by the user Keeps track of the id of worker and department head (storing the next available id	**Collaborators** InOutHandler

for each) to make sure no duplicate id Update the list of all department heads created as well as the next available id for worker and department head based on the map returned from the controller (1 plus an arraylist that is not empty means update list of department heads to the non-empty arraylist and increase the next head ID by 1, 1 plus an empty arraylist means increase the next worker ID by 1 and 0 plus any string means nothing need to be changed) Keeps program taking user input (calling controllers) until user enters exit (returned map has -1 as key) Display the welcome message and message that ask user to enter input and ending message	

Task 4: Scenario Walk-Through When a user wants to change schedule for one worker

1. The controller class which is InOutHandler should take in user's input as a whole string and use the split method to split the input by space
2. Check what is the first word the user entered and since the user entered change as the first word, it goes into the case where the first word is change which is the second case
3. Check what is the second word that the user entered to see if user want to change salary or schedule and then it will go the the case where the second word is schedule
4. The controller creates an instance of the workerManager which is a use case class and call the method that change the schedule
5. Then it proceed to the changeSchedule method in the workerManager class and the method takes the worker's ID, the department head's ID, the list of all department heads created, and the day of week, start time, end time the user want to change to.
6. In the changeSchedule method, it first iterate through the whole list of department head to get the department head with the given ID and if the ID is not in it, then the method directly return the message that the user should see which is no department head found back to the controller
7. If the department head with that ID is found, then the method gets the list of all worker under that department by calling the getter method for workerInDept in an entity class which is departmentHead.
8. Then the program need to go through the list to see if the worker is in the same department and if not, then the program will return the message that the user will see which is department head has no right to change schedule back to the controller.
9. If the worker is found, then the user case class workerManager will call a setter method in the worker entity class called setSchedule which sets the three attributes (day of the week and start time and end time) to the given three input from the user
10. Then, the method will return the message that the user should see back to the controller

11. After this method complete its run, it is back to the controller class that is InOutHandler
12. The controller will directly display the message from the changeSchedule method to the UI and return a map signal that nothing need to be changed (a map with key 0) in HRSystem
13. Then it get backs to the HRSystem and since the map has a key of 0, the system will not do anything.