HRS System

HRS is a Human Resources Solutions system, that provides a platform to manage the hiring process of a group of qualified cleaning-workers and drivers on an hourly basis. The *HRS* management system will begin with 10 employees (cleaning-workers and drivers). However, as a future plan, it will be expanded to reach 100 employees' records.

Table1. Shows the current employees' information in the system

	Residence Permit ID	Job title	Contract Date	Contract start Time	Contract end Time	Hired?	Hiring frequency
1.	9876543222	Driver	15/12/2020	16:00	22:00	yes	1
2.	9876543211	Cleaning Worker	15/12/2020	08:00	16:00	yes	4
3.	1234567899	Cleaning Worker	N/A	N/A	N/A	no	0
4.	2234567891	Cleaning Worker	17/12/2020	08:00	13:00	yes	2
5.	1334567892	Driver	11/12/2020	10:00	15:00	yes	1
6.	4412356789	Cleaning Worker	05/12/2020	12:00	16:00	yes	10
7.	3114567894	Driver	N/A	N/A	N/A	no	2
8.	8876543221	Driver	19/12/2020	08:00	14:00	yes	1
9.	7776543267	Cleaning Worker	N/A	N/A	N/A	no	3
10.	1176543266	Cleaning Worker	02/12/2020	17:00	21:00	yes	12

As shown in the previous table, each employee (worker/driver) is identified by his/her Residence Permit Identification Number. The job title is stored in the system for each employee (cleaning worker or driver). Also, it will store whether the employee is currently hired or not. And if so, the *HRS* system will keep track of the hiring process by storing the current contract's date, start time, and end time. The hiring frequency will be stored to indicate the total number of contracts the employee has completed.

The *HRS* system will displays the following menu to manage the hiring process for all the employees:

1. Add a new employee
2. Start a hiring contract
3. End a hiring contract
4. Display employee info
5. Display *HRS* system status
6. Exit

In this project, you are required to accomplish this task:

Writing a *Java* program and testing it, based on the given description, and the following classes' details:

The class (Employee) contains the following attributes:

- *id*: represents the Residence Permit Identification number of the employee.
- *jobTitle*: represents the job title: (cleaning worker, or driver).
- *contractDate*: represents the contract's date (Format dd/mm/yyyy)
- *startTime*: represents the contract's start time (Format HH:00) using 24-hours notation.
- *endTime*: represents the contract's end time (Format HH:00) using 24-hours notation.
- *hired*: true if the employee is currently hired, or false otherwise
- *frequency*: represents the number of times the employee was hired (number of completed contracts).

The class (Employee) contains the following methods:

- calculateDuration( ): computes and returns the total duration (in hours) of the current hiring contract. For example, if an employee starts at 10:00 and finishes at 13:00 then the duration is 3 hours.
- updateFrequency(): increment the current hiring frequency by one.
- displayInfo( ): displays the employee info on one line with the following format:
  - id: left aligned, with width: 12
  - jobTitle: right aligned, with width: 15
  - contractDate: right aligned, with width: 10
  - startTime: right aligned, with width: 6
  - endTime: right aligned, with width: 6
  - hired: left aligned, with width: 4
  - frequency: left aligned, with width: 4
  - the line starts and ends with "|" and each item is separated by "|"
  - Sample output:

```
|9876543211|Cleaning Worker|15/12/2020|8:00  |16:00 |yes |4   |
```

width:12 width:15 width:10 width:6 width:6 width:4 width:4

- Add setters and getters as needed.

Details for the functions in the menu:

1. Add a new employee:
   Read the employee's (id) from the user, and validate that it does not already exist in the array otherwise display an appropriate message and go back to the menu. After validating the id, read from the user: the job title. Assign the value "N/A" to contractDate, startTime, and endTime. Then, assign value false to hired attribute, and the value 0 to the hiring frequency.
2. Start a hiring contract: (used when a customer requests an employee)
   Read the employee's (id) from the user. Make sure the employee exists and is not yet hired, then read the contract's information: contractDate, startTime, and endTime. Also, set the hired value to true, and the hiring frequency will be incremented by 1. The program should then display the employee's information and state that he/she is now ready for starting the assigned job. In case if the employee is not existed or he/she is already hired, then display appropriate messages and go back to the menu.
   Note: Assume that the user will enter the contract's information correctly, i.e. the date is today or in the future, and the start time is before the end time.
3. End a hiring contract: (used when the employee is done and checks back into the company)
   Read the employee's (id) from the user. Make sure the employee exists and is hired, then display the employee's information and contract's total duration. To end this hiring contract, reset the employee's values so that he/she can be hired again: assign the value "N/A" to contractDate, startTime, and endTime. Then, assign the hired attribute's value back to false. In case if the employee is not existed or he/she is not hired, then display appropriate messages and go back to the menu.
4. Display an employee's info:
   Read the employee's (id) from the user. If the employee is not existed, then display an appropriate message and go back to the menu. Otherwise, display the employee's information.
5. Display *HRS* system status:
   Display a summarized report of the *HRS* system, that includes:
   - The total number of employees who are currently registered in the system.
   - The current number of hired employees, and the number of available employees.
   - The id of the employee with the highest hiring frequency in the system.
   - A listing of all the employees in a table with a proper header.
6. Exit:
   Ask the user: "All info. will be lost. Are you sure you want to exit? (Yes/No)"
   Remember to ignore upper/lower cases and accept all (Yes/No) answers.
   If the user input is "Yes", then exit with an appropriate message. And if the input is "No", then go back to the menu. If the user entered any other input other than (Yes/No), then display this message: "only (yes/no) is accepted!", and go back to the menu again.

Note: If the user entered a selection other than the ones listed in the menu, then display this message: "Not a valid selection!", and go back to the menu again.

To test class (Employee): write a class (TestEmployee) that includes the following methods:

- maximumFrequency(Employee one, Employee two): returns the id of the employee that has the higher frequency, with the condition that both employees have the same job title. Otherwise, it returns -1 to indicate that employees have different job titles.
- minimumDuration(Employee one, Employee two): returns the id of the employee who is working less during the same contract date. If the employees' contract date is different, returns "Dates are Mismatched".
- Add methods as needed.

In the main() method of class (TestEmployee) , perform the following tasks:

1. Create an array of size 100 and name it: ***hiringEmployees***.
2. Fill-in the array with the 10 employees, as described earlier in (***Table 1***).
3. Display which employee has the maximum frequency: 9876543211 or 1234567899?
   if an id is returned, display message: "Employee with the id number: (xxxxxxxxxx) can take a break during the weekend". If -1 is returned, then display the message: "Employees have different job titles".
4. Display which employee worked for less duration: 9876543222 or 9876543211?
   if an id is returned, display message: "Employee with the id number: (xxxxxxxxxx) worked less on (dd/mm/yyyy) ". If "Dates are Mismatched" is returned, then display the message: "Employees worked in a different date".
5. Display which employee has maximum frequency: 9876543211 or 9876543222?
   if an id is returned, display message: "Employee with the id number: (xxxxxxxxxx) can take a break during the weekend. If -1 is returned, then display the message: "Employees have different job titles".
6. Display which employee worked for less duration: 9876543211 or 2234567891?
   if an id is returned, display message: "Employee with the id number: (xxxxxxxxxx) worked less on (dd/mm/yyyy) ". If "Dates are Mismatched" is returned, then display the message: "Employees worked in a different date".
7. end the program by exiting correctly.

Hints & Notes:

- All your output should be formatted properly and neatly
- Use clear prompts when asking for user input
- Start writing your program and leave the testing stage at the end.
- Implement one action at a time.
- Pay attention to repeated code and try to put it into a method
- Support your code by adding comments as needed.

When you finished, submit the following through BlackBoard:

✓ Your code (.java) files. (add your names & ID's as a comment in the beginning of the code)
✓ *Note:* Each group should have only one submission, that represents all members. (Don't submit it individually multiple times)

After submission:

✓ *Each group will be meeting with their lab's instructor to test and discuss their project*
✓ *Your lab's instructor will contact you to schedule the project's group-discussion.*
✓ *All group members must attend the discussion session.*
✓ *Make sure that you all prepare well for answering any project-related questions.*

Final Statement...

Remember to follow the academic integrity rules conducted by ***KSU***. Breaching these rules is strictly prohibited and a Zero credit will be given to groups who are partially/fully share or copy other's code. Manual and automotive checking will be performed to detect plagiarism and code similarity.

Wish you all the best,