Fall 2022– CS 203 Object-Oriented Programming

Homework # 3: Project Report

Name: Haeun Suh, BlazerId: hsuh

Introduction

This project is designed to implement a simple employee management system used in hospital in order to apply OOP concepts which have learned in CS203 class. The system will show overall list of employees with their name, id, and specific role (if any) they have. Each employee will be grouped by their role. Also, the system can add, delete one or more the employee from its system, and be conducted by receive user inputs from console. This system utilizes text file as its database and read and write to its predesigned text file to reflect any changes in the employee management system. The ultimate purpose of this project is to demonstrate the understanding of OOP concepts, an approach to applying OOP to the project, and the ability to implement code in general.

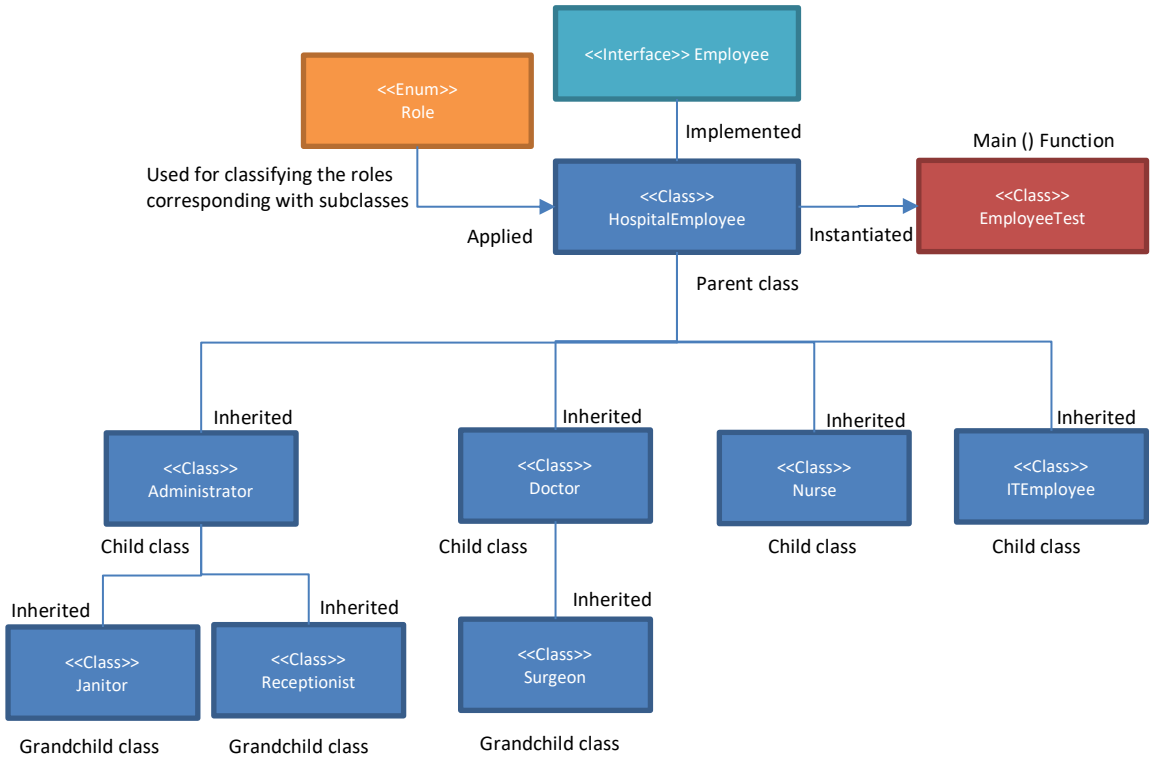
Code Explanation

1) Summary

   In this project, 1 Enum class, 1 interface, and 9 General class are included. OOP is applied to the project such as using override methods(toString()), calling 'super ()' methods from each child class's constructor to utilize their parent's properties, and implemented interface and made to inherit by its child class therefore abstract methods to be implemented within its child class. Getter methods are also used within subclasses to access their parent or grandparent properties.

* For space reasons, on the next page

2) Structure and hierarchy nature of implemented classes.
The structure of the project is like as below:



Interface 'Employee' Class defines the abstract functions to be implemented and 'HospitalEmployee' implements those functions by implementing the 'Employee' interface. 'Administrator', 'Doctor', 'Nurse', 'ITEmployee' inherited the 'HospitalEmployee' class therefore they can utilize their parent's property 'name' and 'blazerId'. 'Janitor' and 'Receptionist' inherited the 'Administrator' class because they are part of administrator team and thus, they inherit the property 'department' on the top of their grandparent's properties. 'Surgeon' class inherits the 'Doctor' class because only doctors can be surgeons. Therefore, surgeon inherits the property 'Specialty'. All subclasses override toString() to display their properties and utilize assessors that inherited from their parent and grandparent. Enum 'Role' is used to identify which class should be added or deleted from given user Input (By using switch statement). The project executed main () function from the class 'EmployeeTest'.

3) The way of working as a whole project.
From the 'EmployeeTest' class, main () function will be executed. From the main class, the class instance of 'HospitalEmployee' will be instantiated and utilized as arguments of the function run () from 'EmployeeTest' class. Firstly, given text file will be readed and its data will be loaded in the properties of 'HospitalEmplyee' then run () function displays menu and receives user input. The corresponding function of class 'HospitalEmployee' whill be executed as the selected menu item. Also, 'HospitalEmployee' will add, delete, display conditionally regarding to the corresponding

roles, which are them are defined as subclasses and stored in each arraylist property. By selection, the modification can be written to the text file. Program will be repeatedly displays its menu until used select ‘Terminate Menu’.

4) Properties and methods of each class.

a) Main Class ‘EmployeeTest’

Name	Type	Parameter	Return	Description
main	Method	String[] args	None	Main function of project
run	Method	String dataFileName, HospitalEmployee employee	None	Displays menu, receive user input, and call function of HospitalEmployee class

b) Interface ‘Employee’
* Only contains abstract methods. (Skip descriptions)

Name	Type	Parameter	Return
loadDatabase	Abstract Method	String dataFileName	None
add	Abstract Method	List<String> empInfo, boolean IsSystemUpdate, Scanner sc	Boolean isAdded
delete	Abstract Method	String role, String blazerId	Boolean isDeleted
updateDatabase	Abstract Method	String dataFileName	Boolean isUpdated
display	Abstract Method	None	None

c) Class ‘HospitalEmployee’
* Description of Assessor/Mutator is skipped.
* All private properties have their own assessor/mutator.
* This class implements Interface ‘Employee’

Name	Type	Parameter	Return	Description
name	String	* These are attributes (or Properties)		Employee’s name
blazerId	String			Employee’s ID
E	ArrayList <HospitalEmployee>			List of hospital employee that no specific role assigned
D	ArrayList<Doctor>			List of doctors
S	ArrayList<Surgeon>			List of surgeons
N	ArrayList<Nurse>			List of nurses
A	ArrayList<Administrator>			List of Administrators
J	ArrayList<Janitor>			List of janitors
R	ArrayList<Receptionist>			List of receptionists
I	ArrayList<ITEmployee>			List of IT employees
totEmpNum	int			Total number of employees
HospitalEmployee	Constructor Method	None	None	Initiate ArrayList attributes

HospitalEmployee	Constructor Method	String name, String blazerId	None	Instantiate and set values to corresponding properties
loadDatabase	Implemented Method	String dataFileName	None	File read and set the values to the corresponding ArrayList properties
add	Implemented Method	List<String> empInfo, boolean IsSystemUpdate, Scanner sc	Boolean isAdded	Add the requested role of employee with given information
delete	Implemented Method	String role, String blazerId	Boolean isDeleted	Delete the requested role of employee with given blazerId
updateDatabase	Implemented Method	String dataFileName	Boolean isUpdated	File write of current status of ArrayList properties
display	Implemented Method	None	None	Loop and print the each object class method toString() from each corresponding ArrayList properties.
toString	Override Method	None	String toString	Return predesigned string utilizing its attributes

d) Enum 'Role'
   * Enum Role contains the items like as below:
   E, D, S, N, A, J, R, I
   * Each item represents the role of employee.

e) Class 'Doctor'
   * Description of Assessor/Mutator is skipped.
   * All private properties have their own assessor/mutator.
   * This class inherits the class 'HospitalEmployee'

Name	Type	Parameter	Return	Description
specialty	String	* This is attribute (or Property)		The field of specialty that the employee has
Doctor	Constructor Method	String name, String blazerId, String specialty	None	Instantiate and set values to corresponding properties
toString	Override Method	None	String toString	Return predesigned string utilizing its attributes

f) Class 'Surgeon'
   * Description of Assessor/Mutator is skipped.
   * All private properties have their own assessor/mutator.
   * This class inherits the class 'Doctor' since surgeon is a doctor.

Name	Type	Parameter	Return	Description
isOperating	String	* This is attribute (or Property)		Whether the employee conducts operation
Surgeon	Constructor Method	String name, String blazerId, String specialty, String isOperating	None	Instantiate and set values to corresponding properties
toString	Override Method	None	String toString	Return predesigned string utilizing its attributes

g) Class 'Nurse'
* Description of Assessor/Mutator is skipped.
* All private properties have their own assessor/mutator.
* This class inherits the class 'HospitalEmployee'

Name	Type	Parameter	Return	Description
numOfPatients	String	* This is attribute (or Property)		Number of patients that the employee manages
Nurse	Constructor Method	String name, String blazerId, String numOfPatients	None	Instantiate and set values to corresponding properties
toString	Override Method	None	String toString	Return predesigned string utilizing its attributes

h) Class 'Administrator'
* Description of Assessor/Mutator is skipped.
* All private properties have their own assessor/mutator.
* This class inherits the class 'HospitalEmployee'

Name	Type	Parameter	Return	Description
department	String	* This is attribute (or Property)		Department to which the employee belongs
Administrator	Constructor Method	String name, String blazerId, String department	None	Instantiate and set values to corresponding properties
toString	Override Method	None	String toString	Return predesigned string utilizing its attributes

i) Class 'Janitor'
* Description of Assessor/Mutator is skipped.
* All private properties have their own assessor/mutator.
* This class inherits the class 'Administrator' since janitor is a part of administrative team.

Name	Type	Parameter	Return	Description
isSweeping	String	* This is attribute (or Property)		Whether janitor has role of sweeping

Janitor	Constructor Method	String name, String blazerId, String department, String isSweeping	None	Instantiate and set values to corresponding properties
toString	Override Method	None	String toString	Return predesigned string utilizing its attributes

j) Class 'Receptionist'
* Description of Assessor/Mutator is skipped.
* All private properties have their own assessor/mutator.
* This class inherits the class 'Administrator' since janitor is a part of administrative team.

Name	Type	Parameter	Return	Description
isAnswering	String	* This is attribute (or Property)		whether the employee currently answering
Receptionist	Constructor Method	String name, String blazerId, String department, String isAnswering	None	Instantiate and set values to corresponding properties
toString	Override Method	None	String toString	Return predesigned string utilizing its attributes

k) Class 'ITemployee'
* Description of Assessor/Mutator is skipped.
* All private properties have their own assessor/mutator.
* This class inherits the class 'HospitalEmployee' because its role is independent from administrative team.

Name	Type	Parameter	Return	Description
team	String	* This is attribute (or Property)		Team in which the employee belongs to
ITemployee	Constructor Method	String name, String blazerId, String team	None	Instantiate and set values to corresponding properties
toString	Override Method	None	String toString	Return predesigned string utilizing its attributes

5) Benefits of the implementation
* Easy to distinguish roles by their categorized inheritance. For example, the employee who is part of administrator.
* Reusability and minimizing duplication. For example, when doctor gets more attributes, surgeon will automatically inherit.
* Using Interface can control exist methods therefore, prevent unused or duplicated methods.

Result

1) display () function

```
Problems  @ Javadoc  Declaration  Console ×
EmployeeTest (1) [Java Application] C:₩Program Files₩Java₩jdk-18.0.1.1₩bin₩javaw.exe  (2022. 10. 14. 오후 9:42:41) [pid: 17768]
========= MENU SELECTION ================
1. Display Employee List
2. Add New Employee
3. Update DataBase
4. Delete Employee
5. Terminate Menu
=========================================

SELECT: 1

========= Display Employee List =========

****************************************
** Welcome to the UAB Employee System! **
****************************************

The UAB Hospital System has the following employees:

Total Number of employees=10

Hospital Employees: 1
Name: Lucy BlazerId: 1995

Doctors: 2
Name: John BlazerId: 1187 Specialty: Gastro
Name: Samuel BlazerId: 4891 Specialty: Radiology

Surgeons: 1
Name: Steve BlazerId: 4345 Speciality: Brain Operating Y

Nurses: 1
Name: Julia BlazerId: 1254 Number of Patients: 4

Administrators: 2
Name: Jacob BlazerId: 4123 Department: Business
Name: Catherine BlazerId: 4125 Department: Management

Receptionist: 1
Name: Alice BlazerId: 4101 Department: Talking Answering: Y

Janitors: 1
Name: David BlazerId: 4789 Department: Maintenance Sweeping: N

IT Team: 1
Name: Mark BlazerId: 1997 Team: Developers
```

[Select: "1. Display Employee List"]

2) add () function

```
Problems  @ Javadoc  Declaration  Console ×
EmployeeTest (1) [Java Application] C:₩Program Files₩Java₩jdk-18.0.1.1₩bin₩javaw.exe  (2022. 10. 14. 오후 9:40:48) [pid: 23376]
========= MENU SELECTION ================
1. Display Employee List
2. Add New Employee
3. Update DataBase
4. Delete Employee
5. Terminate Menu
=========================================

SELECT: 2

========= Add New Employee ==============

*** Enter the Role as a Char like as below ***
A: Administrator, D: Doctor, E: Hospital Employee,
I: IT professinal N: Nurse, R: Receptionist, S: Surgeon

The New Employee's Role: I
The New Employee's Name: haeun
The New Employee's BlazerId: hsuh
*** Enter the Team as a Char like as below ***
D: Developers, W: Web services, N: Networking
The New Employee haeun's Team: D
The new employee successfully added!

=========================================

========= MENU SELECTION ================
1. Display Employee List
2. Add New Employee
3. Update DataBase
4. Delete Employee
5. Terminate Menu
=========================================

SELECT:
```

[Select: "2. Add New Employee"]

3) delete () function

```
========= MENU SELECTION ================
1. Display Employee List
2. Add New Employee
3. Update DataBase
4. Delete Employee
5. Terminate Menu
=========================================

SELECT: 4

========= Delete Employee ===============

*** Enter the Role as a Char like as below ***
A: Administrator, D: Doctor, E: Hospital Employee,
I: IT professinal N: Nurse, R: Receptionist, S: Surgeon

Role of The Employee to be Deleted: i
BlazerId of The Employee to be Deleted: 1997
The employee is successfully deleted!

=========================================
```

[Select: "4. Delete Employee"]

4) update () function

```
========= MENU SELECTION ================
1. Display Employee List
2. Add New Employee
3. Update DataBase
4. Delete Employee
5. Terminate Menu
=========================================

SELECT: 3

========= Update DataBase ===============

****************************************
********** DataBase Updated!! ***********
****************************************

=========================================
```

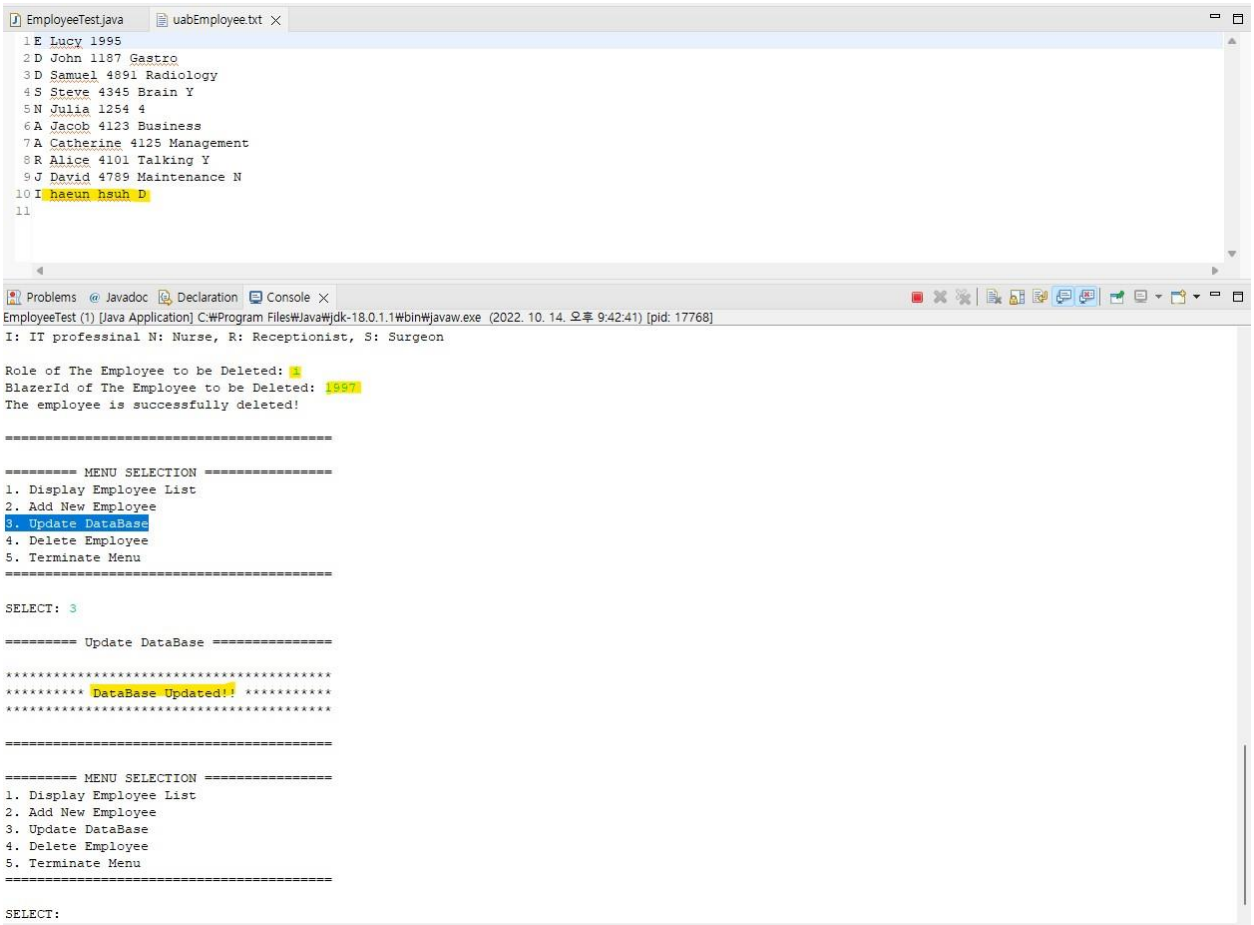
[Select: "3. Update DataBase"]

5) terminate () function

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> EmployeeTest (1) [Java Application] C:₩Program Files₩J
==========================================

========= MENU SELECTION =================
1. Display Employee List
2. Add New Employee
3. Update DataBase
4. Delete Employee
5. Terminate Menu
==========================================

SELECT: 3

========= Update DataBase ================

******************************************
********** DataBase Updated!! ************
******************************************

==========================================

========= MENU SELECTION =================
1. Display Employee List
2. Add New Employee
3. Update DataBase
4. Delete Employee
5. Terminate Menu
==========================================

SELECT: 5

========= Terminate Menu =================

******************************************
**** Thank you for using our system! ****
******************************************

==========================================
```

[Select: "5. Terminate Menu"]

6) After 1) ~ 5) Execution



[File updated after executions]

References

1. Class materials including lab slides.
   * In particular, reference was made to ‘CS203_Lab07_Slides.pdf’.
2. String format usage.
   [javatpoint]https://www.javatpoint.com/java-string-format
3. Java documentation for ‘Path’.
   [Path]https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/nio/file/Path.html