



King Abdulaziz University
Faculty of Computing and Information Systems
Fall 2020 – 1stTerm 2020

Course Code: CPCS203

Course Name: Programming II

Assignment # 3 (KAU Hospital System)

Assigned Date: Sunday, 8th November 2020 (8/11/2020)

Delivery Date: Saturday, 28th November 2020 (28/11/2020)@11:00PM

WARNING:

- This program must ONLY be submitted on the Blackboard!
- This is an Individual assignment. You must solve it by yourself. Any form of plagiarism will result in receiving 0 (zero) in the project.
- This project worth 10% of the overall module marks (100%).

Objectives

- Learn how to use and implement the concept of Inheritance, polymorphism and abstract.
- Learn how to use and implement Passing Object concepts.
- Learn to use and implement String, StringBuilder, File I/O (Reading/Writing from/to files).

Delivery

- Submit your assignment on the Blackboard ONLY.
- **Make sure to add your names / IDs / Section / course name / Assignment number, as comment at the beginning of your program.**

Description

KAU Hospital System

What is KAU Hospital System?

KAU Hospital System is simulation software which simulates hospital functionality. Usually Room, Doctor, Nurse and Patient register in the system. System allocate Room, Doctor and medicines to the patient.

Further, this system print Billing report for each patient.

System Read ALL data from a given input file [**input.txt**] and generate result and reports in an output file like [**Output.txt**].

More Details are as follows:

You are required to make 3 different arrays to store all the data used in the system. The first array is **HRooms** to store all rooms records details. The second array is **HMedicines** to store all medicine records details. The last array is **HPersons** to store all Doctors, Nurse and Patients in the hospital in one array.

The Initial Procedure of the Program: Your program will use File I/O to read input from a given file name [**input.txt**]. Make sure the file exists or display a message that the file does not exist.

The file consists of **3 integers** to determine the size of the **HRooms**, **HMedicines** and **HPersons**. [**see the input file**]:

- a) The first number (10) in the file refers to the number of Rooms in the system [means system will accept ONLY TEN rooms records details]
- b) The second number (20) refers to the number of Medicine in the system [means system will create ONLY TWENTY medicine records details]
- c) The third number (15) refers to the number of **HPersons** (they could be Doctors, Nurse or Patients) in the System [means system will accept ONLY FIFTEEN persons records details]
- d)

The commands you will have to implement are as follows:

- + **Add_Room** – Makes a new Room which is added to the **HRooms** array. The command will be followed by the following information ALL on the same line:

an integer representing **roomNo**; a String representing **floor** of a room; a String representing **block** of a room, and double representing **charges** of a room as shown in the underneath example. [see input.txt]

Add_Room 101 First B 700.0

In above command: **roomNo** =101, **floor** = First, **block** = B, **charges** = 700.0

Note: Each room record must be saved in the **Output.txt** file as per given sample **Output.txt** file.

- + **Add_Medicine** -Makes a new Medicine which is added to the **HMedicines** array. The command will be followed by the following information ALL on the same line:

an integer representing **medicineCode** of a Medicine; a String representing **name** of a Medicine; and double representing **price** of a Medicine as shown in the underneath example. [see input.txt]

Add_Medicine 4001 Azithromycin 75.0

In above command: **medicineCode:** 4001, **name:** Azithromycin, **price:** 75.0

Note: Each Medicine record must be saved in the **Output.txt** file as per given sample **Output.txt** file.

- + **Add_Doctor** – Makes a new doctor which is added to the **HPersons** array. The command will be followed by the following information, ALL on the same line:


a double representing **consultationFees** of a doctor; a string representing **staffid** of a doctor; an String representing **specialization** of a doctor; a int

representing **id** of a doctor; a String representing **name** of a doctor; a String representing **nationality** of a doctor; a char representing **gender** of a doctor; a int representing **Phone** of a doctor; as shown in the underneath example. [see input.txt]

```
Add_Doctor 450.0 C335 Cardiologist 10005  
Mohammed_Ali_Zahrani Saudi M 5001234
```

In above command: **consultationFees** =450.0, **staffid** =C335, **specialization** = Cardiologist, **id** =10005, **name** = Mohammed_Ali_Zahrani , **nationality** = Saudi , **gender** = M and **Phone** = 5001234

Note: Each doctor record must be saved in the **Output.txt** file as per given sample **Output.txt** file.

 **Add_Nurse** Makes a new Nurse which is added to the **HPersons** array. The command will be followed by the following information ALL on the same line:

an integer representing **experience** of a nurse; a string representing **staffid** of a nurse; an String representing **specialization** of a nurse; an int representing **id** of a nurse; a String representing **name** of a nurse; a String representing **nationality** of a nurse; a char representing **gender** of a nurse; a int representing **Phone** of a nurse; as shown in the underneath example. [see input.txt]

```
Add_Nurse 10 V345 Ventilator_care 9001 Faisal_Zahrani Saudi M  
4001234
```

In above command: **experience** =10, **staffid** =V345, **specialization** = Ventilator_care, **id** =9001, **name** = Faisal_Zahrani, **nationality** = Saudi , **gender** = M and **Phone** = 4001234

Note: Each nurse record must be saved in the **Output.txt** file as per given sample **Output.txt** file.

 **Add_Patient**

Makes a new Patient which is added to the **HPersons** array. The command will be followed by the following information ALL on the same line:

an String representing **illness** of a patient; a String representing **bloodGroup** of a patient; a int representing **id** of a patient; a String representing **name** of a patient; a String representing **nationality** of a patient; a char representing **gender** of a patient; a int representing **Phone** of a patient; a int representing **total Medicine** of a patient; as shown in the underneath example. [see input.txt]

Add_Patient Suffering_from_High_Blood_Sugar O+ 20001 Meshal
Saudi M 50123456 3

Note: Each Patient record must be saved in the **Output.txt** file as per given sample **Output.txt** file.

Assign_Doctor_Patient

This command will be used to assign a doctor to a patient [see input.txt]

Assign_Doctor_Patient 10005 20001

Doctor with ID: 10005 is assigned to Patient ID: 20001.

You must check that both patient and doctor exist.

Note: Each Doctor-Patient assigned action must be saved in the **Output.txt** file as per given sample **Output.txt** file.

Assign_Room_Patient

This command will be used to allocate a room to a patient [see input.txt]

Assign_Room_Patient 101 20001

In above command: Room with roomNo: 101 is allocated to Patient ID: 20001.

You must check that both patient and the room exist.

Note: Each Room-Patient allocation action must be saved in the **Output.txt** file as per given sample **Output.txt** file.

Assign_Medicine_Patient

This command will be used to allocate Medicine to a patient [see input.txt]

Assign_Medicine_Patient 20001 3 4001 4004 4005

In above command: 3 Medicines (4001, 4004 and 4005) will be allocated to Patient 20001

You must assure that the patient exists, and only 3 medicines should be assigned to it. You also should check that all medicine numbers are exist.

Note: Each Medicine to Patient allocation action must be saved in the **Output.txt** file as per given sample **Output.txt** file.

PrintBill

This command will be used to print ALL Patients details and bill from the **HPersons** array (only patients). The Information must be sorted by Person ID.

Note:

- You should use instanceof to select the patients from the **HPersons** array.
- To sort the **HPersons** array you must override interface comparable and override the compareTo method.

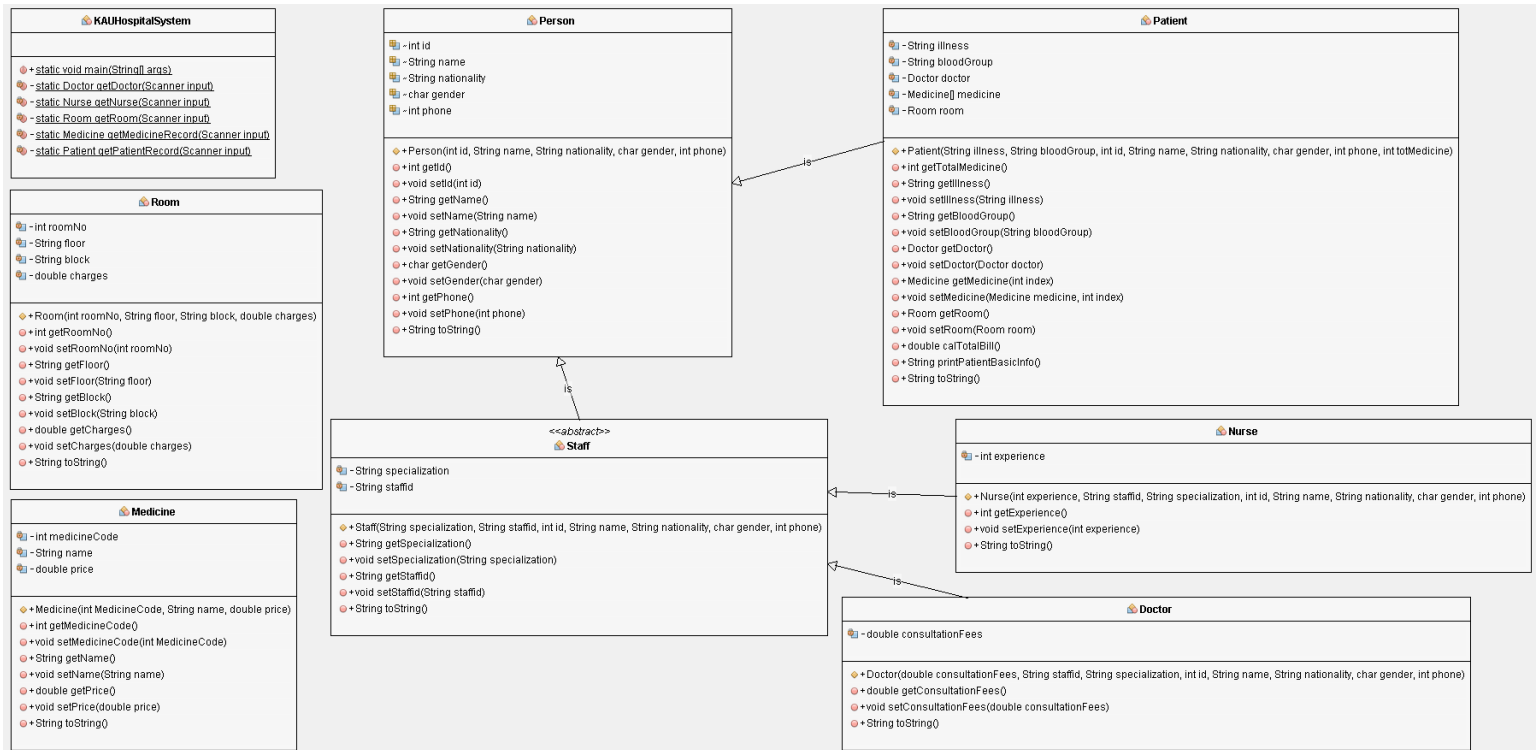
The patient should pay the doctor's fees, the room charge and all his medicines price.

Further Details are as follows:

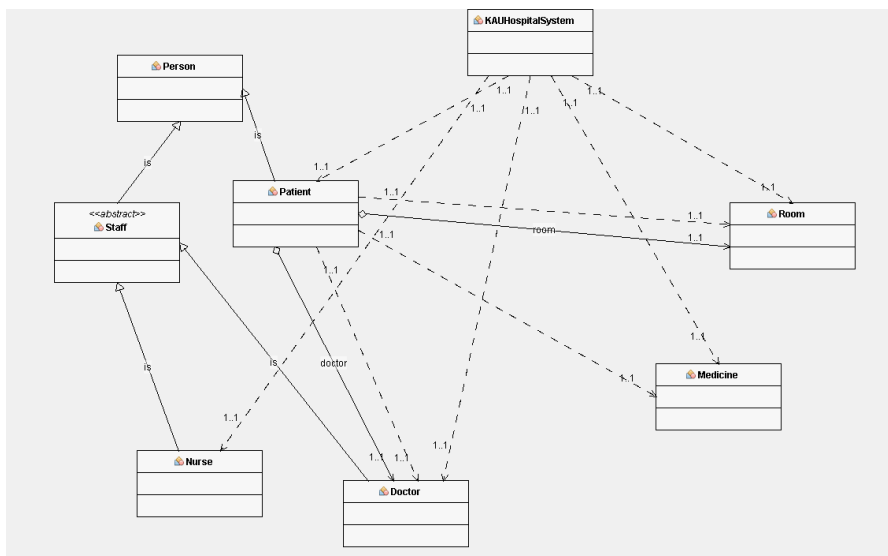
You must create 8 classes in this program.

- Person class is super class of Patient and the Staff classes.
- Staff class is a sub-class of Person.
- Doctor class is a sub-class of Staff.
- Nurse class is a sub-class of Staff.
- Patient class is a sub-class of Person.
- Room class to store the room details.
- Medicine class to store medicine details.

- **Tester class (KAUHospitalSystem)** to create objects and invoke appropriate methods for program to execute successfully.
- See the **UML Diagram underneath** to know basic class details.



UML Diagram of KAUHospital System



Relations UML Diagram of KAUHospital System

Zoom word file if you are unable to see the above UML Class diagram.

Important Notes:

- Use of class & object, arrays of Object, passing object to method and Inheritance is mandatory.
- Use of Files, Reading/Writing from/on files and String & StringBuilder methods.
- Your program output must be exactly same as given sample output files.
- Your display should be in a readable form.
- Organize your code in separated methods.
- Repeat the program until command=Quit.
- Document your code with comments.
- Use meaningful variables.
- Use dash lines between each method.
- **(Delayed submission will not be accepted and there will not be any extension of the project).**

Deliverables:

- You should submit **one zip file containing all java codes:**

NameIDKAUHospitalSystem.java

NOTE: your name and ID should be included as comments in all files!

Input and Output Format

Your program must generate output in a similar format to the sample run provided.

Sample input: See sample input file.

Sample output: See sample output files.

Good Luck and Start Early!