FACULTY OF COMPUTING & INFORMATION TECHNOLOGY
KING ABDULAZIZ UNIVERSITY

FCIT KAU

كلية الحاسبات وتقنية المعلومات
جامعة الملك عبدالعزيز

Fall 2023 – 1st Term

Course Code: CPCS 203	Course Name: Programming II

Health Management System (HMS)

Assignment 2

Assigned Date: Saturday, October 15, 2022
Delivery Date: Tuesday, November 08, 2022

Instructions

- This program must ONLY be submitted on the Blackboard!
- This project worth 15% of the overall module marks (100%).
- NO assignment will be accepted after 11:59 pm for any reason
- Students can submit their assignment between 11 and 11:59 PM but in this case, it will be

considered as late submission, and they will lose 2 points from the total mark of the assignment.

- For discussion schedule, check the captain name, date and time on the BlackBoard.
- Further information is provided in the course syllabus.
- ***Further information is provided in the course syllabus.***

Objectives

- Practice on the use of the **inheritance**, and **dynamic binding**.
- Learn to use and implement **polymorphism** and **object explicit casting**.
- Learn to use and implement **ArrayList** class.
- Learn to use the **instanceof** operator.
- Learn to use **abstract Classes** and **interface**

How to submit your assignment?

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

Files provided with assignment

- One input file: Input.txt
- One output file: Output.txt

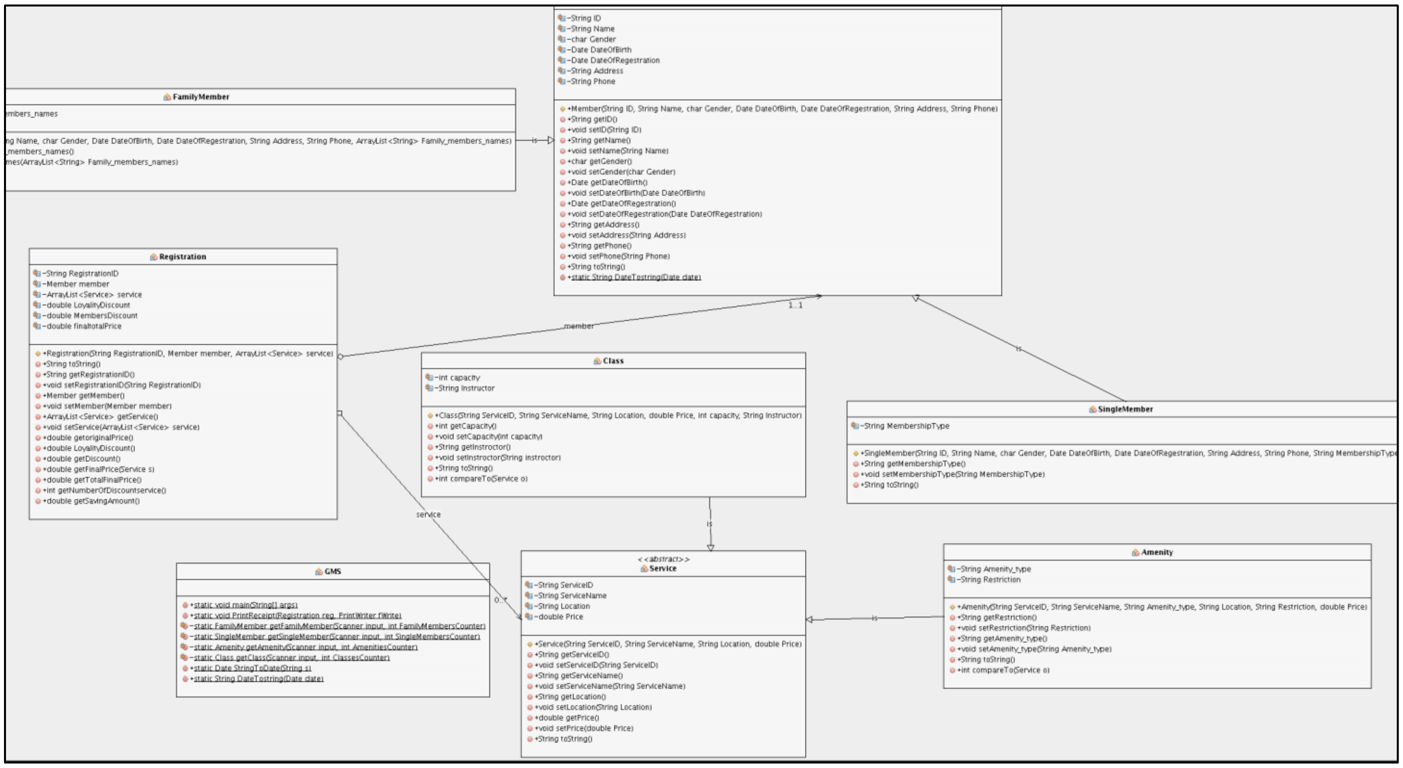
1.1 The HMS Software Description

HMS is a gymnasium management system that helps any gym admin or help desk staff to **manage the gym members' accounts** by 1) **registering** members who can be **singles** or **families**, 2) **applying** special discounts to some members, 3) **registering** members in physical activity classes and services, 4) **printing** invoices and related search results. The HMS has two types of members: **single** and **family**. The **member class** is a super-class **abstract** that has the common attributes and methods in all types of members (see the figure below). The two sub-classes **SingleMember** and **FamilyMember** each extends the super-class and define other more specific attributes and methods. The gym system also offers many **services** (abstract class) that can be divided into two subclasses **classes (e.g., cardio, and cycling)** and **amenity (e.g., swimming-pool, and steam-room)**.

After adding the **members and services** to the system. The help desk staff would be able to **register** any member in any **services**. The help desk staff are also able to apply a **special discount** on some memberships if the member **is registered for 10 years or more**. The gym also offers the **VIP costumers additional discount in some of classes only**. The following sections provide more information about the classes and commands in the **HMS** System.

Step 1: Creating the Classes

The first step in this project is **creating the required classes (eight classes including the main class),** and **one interface** as shown on the UML diagram. Note that the figure below shows the inheritance relationships only. You can add any additional method to the main class but all the other classes should include the methods specify in the figure below.



Zoom to see enlarge of this image

Step 2: Adding the System Elements

Before registering the member in any class and to print his/her receipts, the help desk staff need add all the members **single and families**. Four main commands need to be implemented in this step **Add_Single_Member**, **Add_Family_Member, Add_Classes,** and **Add_Amenity**. The following sections described each command in details.

Command: Add_Single_Member

This command is used to add a single member to the system with all its related information. To add a member as a single, the following information need to be specified: **ID**, **Name, DateOfBirth, DateOfRegestration, Address, Phone, and MembershipType**. Check the following example and table. Note that the **Single_Member_ID** will be a **unique 4- digit number starts with 1001.**

Command Example
Add_Single_Member Ahmed_Mohammed M 15/08/1980 20/8/2008 Alsalamah +96677700023 VIP

Field name	Type	Example
MemberID	String	`1001 (auto-generated)`
Name	String	`Ahmed Mohammed`

Gender	Char	M
DateOfBirth	Date	15/08/1980
DateOfRegestration	Date	20/8/2008
Address	String	Alsalamah
Phone	String	+966577700023
MembershipType	String	VIP

Command: Add_Family_Member

This command is used to add a single member to the system with all its related information. To add a family member, the following information need to be specified: **ID**, **Name, DateOfBirth, DateOfRegestration, Address, Phone, and Family_members**. Check the following example and table. Note that the **family_member_ID** will be a **unique 4- digit number starts with 2001.**

Command Example
Add_Family_Member Mohammed_Kareem M 11/10/1977 22/02/2005 Alsafa +966566660000 Mai_Kareem/Maha_kareem

Field name	Type	Example
ID	String	2001 (auto-generated)
Name	String	Mohammed Kareem
Gender	Char	M
DateOfBirth	Date	11/10/1977
DateOfRegestration	Date	22/02/2005
Address	String	Alsafa
Phone	String	+966566660000
Family_members	Arraylist <String>	Mai_Kareem,Maha_kareem

Command: Add_Class

This command will add a class to the system. To add a class to the system, we need the **Class ID, class name, location, price, and capacity**. Note that the **class ID** will be a **unique 4- digit number starts with 5001.**

Command Example
Add_Class Cardio Female-branch 160 50 Dana_Rami

Field name	Type	Example
ServiceID	String	`5001 (auto-generated)`
ServiceName	String	`Cardio`
Location	String	`Female-branch`
Price	double	`160.0`
Capacity	int	`50`
Instructor	String	`Dana_Rami`

Command: Add_Amenity

This command will add an amenity to the system. To add an amenity to the system, the following information are required: **Amenity ID, Amenity name, location, restrictions, and price**. Note that the **class ID** will be a **unique 4- digit number starts with 6001.**

Command Example
Add_Amenity jacuzzis comfort male-branch above-18 0

Field name	Type	Example
Amenity ID	String	`6001 (auto-generated)`
Amenity Name	String	`jacuzzis`
Amenity type	String	`Comfort`
Location	String	`male-branch`
Restriction	String	`Above 18`
Price	double	`0`

Step 3: Register_class

After entering all the current members, classes, and amenities. Help desk staff can register the member either single or family in the available classes and amenities in their locations.

Command: Register

This command is used to register a gym member in classes and amenities. Note that when registering a member in a class or an amenity, the system will check if there are any restrictions on the member (e.g., age restrictions) and show an appropriate error message if so.

The system also **apply a special discount** for VIP members (see additional note below regarding the discount) and **members who their is membership more than** 10 years will receive a **10% discount**.
After finishing all the ***Registration*** commands and reading command ***submit***, the system will automatically **print a receipt** which has the members ID, names, their original prices, applied discount. (see PrintReceipt command section).

Not that you need to create an object of **Registration** for all the registration made by a given members. The **receiptID** in the registration object should be automatically generated as follows: ID number+ two digit (a counter of the number of registration by the member).
For example, if this is a member with ID `1001` requested his first order the receiptID will be equal to **`1001001 [1001(customer number)001 first(request)].`**

Command Example
Register 1001 **In cardio** **And Zumba** **And cycling** **And sauna** **submit**

Field name	Type	Example
MemberID	String	`1001`
ServiceName	String	`cardio`
ServiceName	String	`zumba`
ServiceName	String	`Cycling`
ServiceName	String	`Suna`

Note that:
- This command first check if the if the member id existed in the system or not. If the member does not exist, the system shows a message saying, "the member does not exist in the system".
- The system then checks if there is any restriction on the member age. For example, if the user age is below 18 and he want to register in sauna an error message will appear saying "The requested service Sauna is not allowed for members below 18 years old".
- The system then checks if there is any restriction on daycare. For example, if the user is a SingleMember and he want to register in child daycare services. The system will show an error message saying, "The requested service Daycare is not allowed for Single members".
- The system then searches for the first service in the system (e.g., class cardio in the above example) and check if the class located in the member gym or not (e.g., if the member is

female her gym will be located in the female-branch). If no class existed in the desired location a message will appear saying, "The requested service cannot be found in the member location".

- If the services and member were available, the system will create an object of registration for the member and the services.

- Note that, the system will apply a **special discount** on classes according to the following rules:
  - VIP members receive a **discount rate of 50%.**
  - Members with membership of 10 years or more **receive 10% discount**.
  - Members with more than 3 family members **receive 5% discount.**

- The method then will call a method **PrintReceipt** to print an invoice of the member fees.

Step 4: Print information

1. Print_receipt

After the member is registered in all the requested services the system prints a receipt. The receipt should have all the member services, their original price, the special discount, and the final price. The following figure shows an example of a printed receipt and other example are provided in the ***output.txt*** file.

Note that:

- You **MUST** use ***polymorphism***, ***casting***, **and** ***isinstanceof*** when you are implementing the method ***Print_receipt.***
- The method ***Print_receipt*** should take an object of type **Registration** which has an object of member and an array **of the registered services.**

The output of the print_ receipt command:

```
------------------------- Invoice Details -------------------------
Registration Reference Number:200101
Member ID:2001
Member name:2001
-------------------------------------------------------------------
Service                  Original_Price     Discount     Final_price
Cardio                   160.00             5.0%         152.00
Zumba                    150.00             5.0%         142.50
-------------------------------------------------------------------
Number of discounted items: 2
- Original Total Price: 310.00
- Final Price: 294.50
- Saving Amount: 15.50
-------------------------------------------------------------------
```

2. Print Sorted Available Services

Command: Print_Sort_Classes

Print_Sort_Classes will sort all the **classes** by **price** and print the list of **sorted classes**. Note that you **MUST** use the interface **comparable**, **collection**, and **Arraylist** to sort and print the list of the classes. An example of the output of **Print_Sort_Classes** is given below:

The output of the Print_Sort_Classes command:

```
----------------------------------------- Classes Sorted By Price ---------------------------------
Class name            Location       Instructor             Price
---------------------------------------------------------------------------------------------------
Zumba                 Female-branch  cindy_jones            150.0
Cycling               Female-branch  Tamara_Jameel          150.0
Zumba                 Female-branch  kathleen_johnson       150.0
Zumba                 Male-branch    Jhone_jay              150.0
Cycling               Male-branch    Tareg_Alzahrani        150.0
Zumba                 Male-branch    Mohamed_Alhomod        150.0
Cardio                Female-branch  Dania_rami             160.0
Cardio                Female-branch  Madison_jones          160.0
KickBoxing            Female-branch  Manal_Tawfeeq          160.0
Cardio                Female-branch  Margaret_gonzalez      160.0
Cardio                Male-branch    Ahmed_Jameel           160.0
Cardio                Male-branch    David_jones            160.0
KickBoxing            Male-branch    Kamel_belal            160.0
Cardio                Male-branch    Madi_gonzalez          160.0
Tennis                Female-branch  Maria_sharapova        200.0
Tennis                Male-branch    Shadi_Moneeb           200.0
---------------------------------------------------------------------------------------------------
```

Command: Print_Sort_Amenities

Print_Sort_Amenities will sort all the amenities by **name** and print the list of **sorted amenities**. Note that you **MUST** use the interface **comparable**, **collection**, and **Arraylist** to sort and print the list of the amenities. An example of the output of **Print_Sort_Amenities** is given below:

The output of the Print_Sort_Amenities command:

```
----------------------------------------- Amenities Sorted By Name --------------------------------
Amenity name          Location       Restriction           Price
--------------------------------------------------------------------------------------------------
Daycare               Female-branch  have-children         50.0
Jacuzzis              Female-branch  above-18              0.0
Jacuzzis              male-branch    above-18              0.0
Massage               Female-branch  above-18              30.0
Massage               male-branch    above-18              30.0
Sauna                 Female-branch  above-18              0.0
Sauna                 male-branch    above-18              0.0
Steam-room            male-branch    above-18              0.0
Steam-room            male-branch    above-18              0.0
Swimming-pool         male-branch    above-18              0.0
Swimming-pool1        Female-branch  above-18              0.0
Swimming-pool2        Female-branch  above-18              0.0
Tanning-Bed           Female-branch  above-18              0.0
-------------------------------------------------------------------------------------------------
```

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
- 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.
- 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: BA1587412P2_HMS .java where BA is your section, 1587412 your ID and P2 is program 2.
- **NOTE: your name, ID, and section number 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.