### Project Documentation

**Group Members:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **ID** | **Name** | |
| **1** | 20200060 | Hesham Omar Mahmoud | (Team Leader) |
| **2** | 20200235 | Abdallah Mohamed Hussein | |
| **3** | 20200153 | Ahmed Mohamed Ali | |
| **4** | 20210713 | Mostafa Ayman Bahaa | |
| **5** | 20210714 | Mahmoud Mohamed Yahya | |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Introduction:

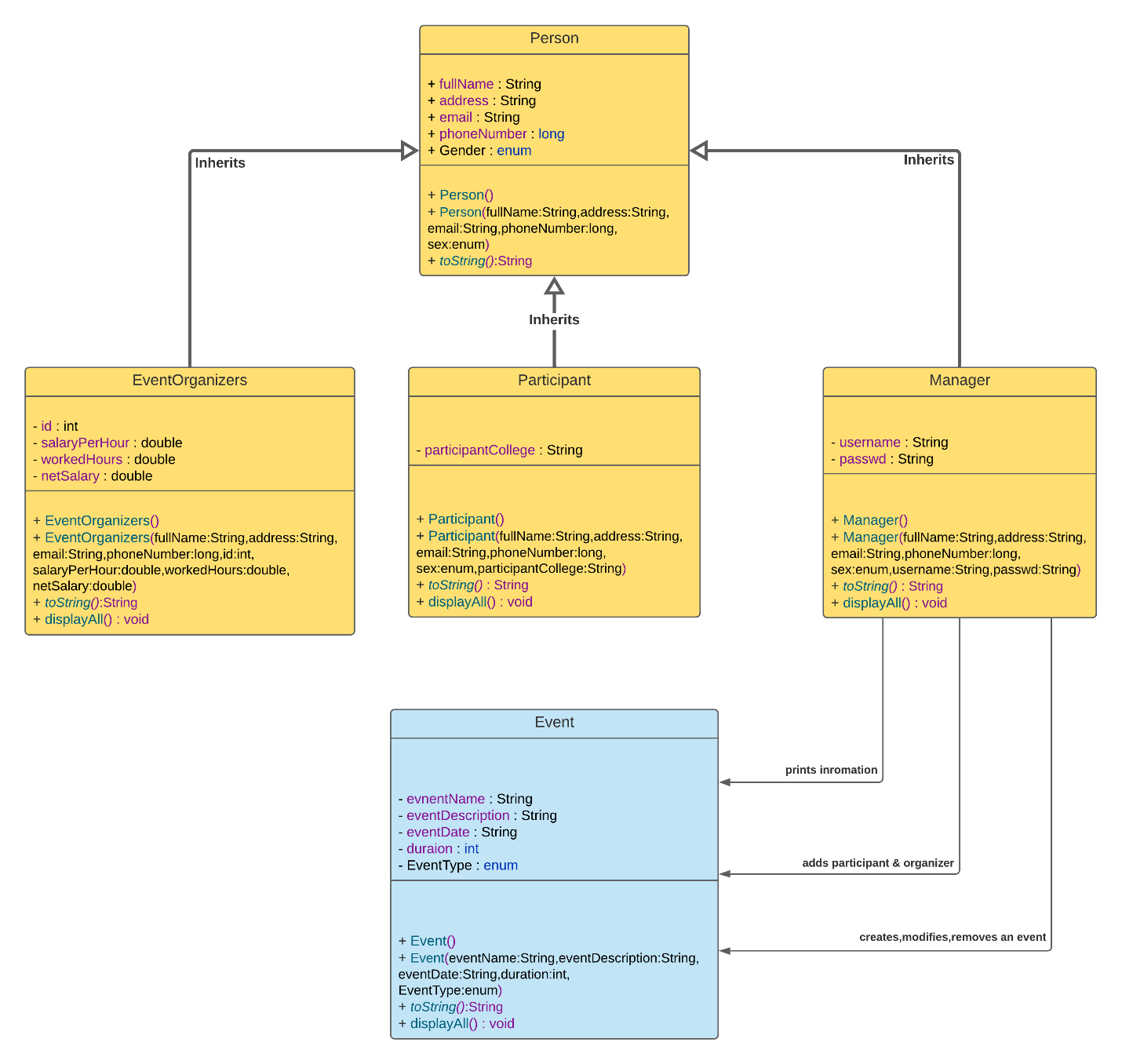
The program is used to manage various types of events such as sports events, competitions, parties or any kind of social gatherings.

The program prints out the event information, allows the admin to book reservations for customers, it keeps track of the participants information, outputs date and total price of the event and so on.

Our program stores and retrieves information about the manager, the event, the participants and the event organizer; therefore, we’ve created the following:

* Abstract Class Peron(superclass):
* This class contains the basic shared information that every person on the system possesses such as (full name, email, phone number, etc.…).
* It also contains the (toString()) method as an abstract method which will be overridden by all subclasses that inherit from the person class.
* Class Manager:
* This class inherits its basic attributes from the (Person Class).
* It also adds its own unique attributes which are the username and password.
* It overrides the (toString()) method to display all its attributes (displays all attributes inherited form the super class as well as its own unique attributes).
* Class Participant:
* Much like the pervious class (Manager) this class behaves very similarly.
* It inherits its attributes from the (Person Class) and adds its own attribute which is participantCollege
* It overrides the (toString()) method like before.
* Class EventOrganizers:
* Behaves like the two previous classes.
* Class Event:
* This class stores, modifies and displays all the details about the event itself.
* It contains and arraylist that stores called (ParticipantsAndOrganizersList) which stores all the participants and event organizers for each instance of the class event.
* It has an overridden (toString()) method as usual but it also has a method called (displayAll) which displays the organizers and participants of an event by looping over the ParticipantsAndOrganizersList ArrayList.
* This requires us to use **type casting** when looping over the arraylist to be able to access either the participant or the organizer instances inside the arraylist.
* Without type casting we won’t be able to access objects from the arraylist because the participant and organizer classes both inherit form the base class (Person) and the variable type of the arraylist is (Person).

1. UML Diagram:



1. Description:

* **Enums** have been used in the (Person Class) to set two constants’ values for the gender of person {Male, Female}. Also, in the Event class to set to constant states of an event {Weekly, Monthly}.
* **Arraylist:** was used to store participants and organizers in (Event Class).
* **For loop:** was used to access elements in arraylist.
* **If Statement:** was used for typed casting to distinguish between participant and organizer instances.

1. References:

<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>

<https://www.youtube.com/watch?v=xk4_1vDrzzo>

<https://www.w3schools.com/java/java_enums.asp>

<https://www.w3schools.com/java/java_polymorphism.asp>

<https://www.w3schools.com/java/java_inheritance.asp>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*