Faculty of Engineering and Information Technology
Computer Science Department
COMP2311 Project (Phase 1)

Project title: The War on Gaza - A Humanitarian Information Management System

Project objectives: Apply object-oriented programming concepts using Java, including classes, inheritance, and polymorphism.

Project Description:

Develop a Java-based system to efficiently manage and analyze information about individuals affected by the war in Gaza. The system should include data on martyrs, orphans, live persons, and families. Utilize object-oriented principles to model these entities, establish relationships between them, and implement functionalities for data manipulation and statistical analysis.

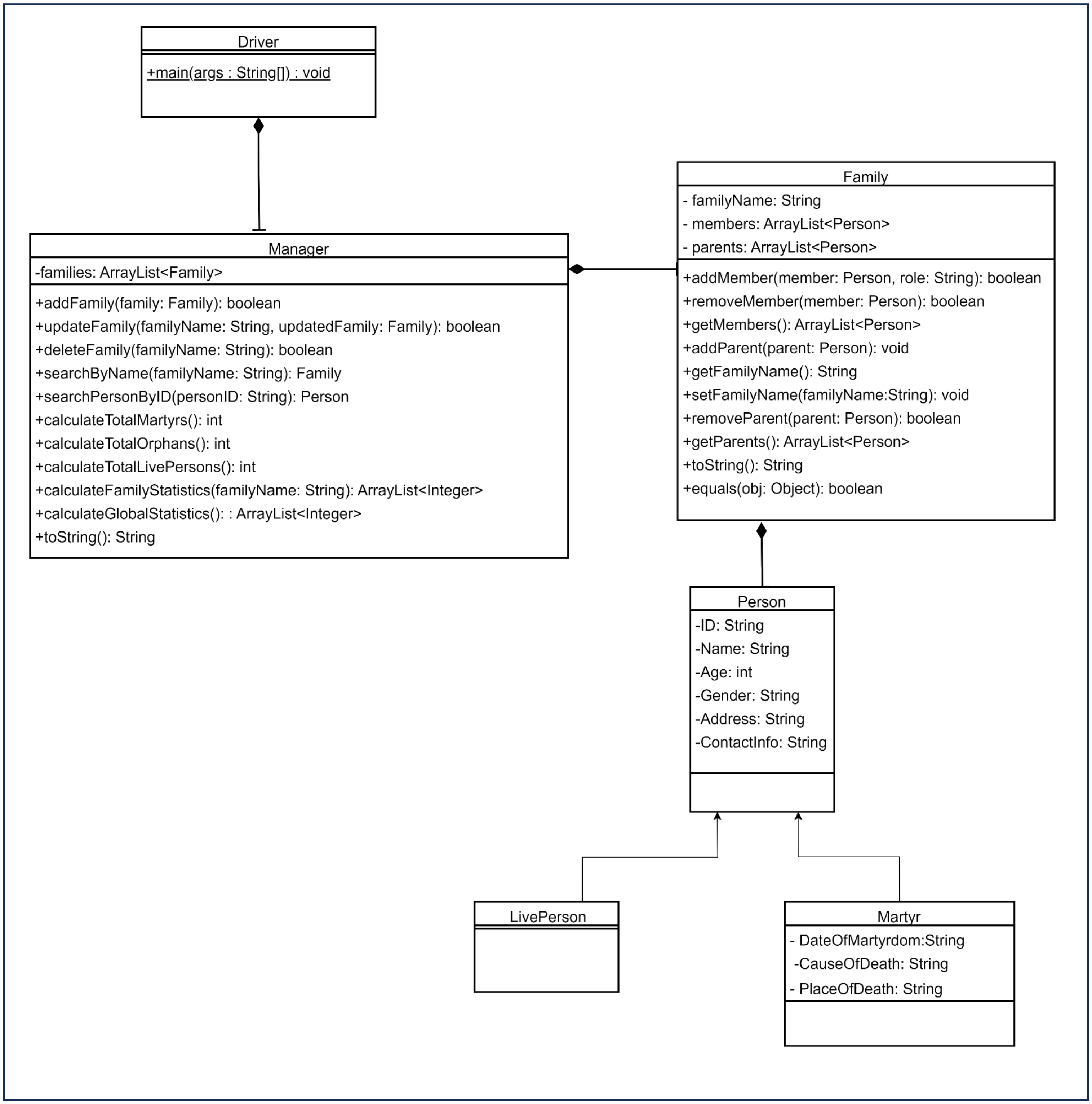
Instructions :

- Implement the classes according to the provided specifications. Pay attention to inheritance, polymorphism, and proper encapsulation.
- Provide a sample MainDriver class to demonstrate the functionality of your implemented classes.
- Show all the clarification messages for the user (updated successfully, added successfully, removed successfully, etc).

Classes to implement:

1. Person Class: Represents a generic individual affected by the war.
2. Martyr Class: Represents an individual who lost their life due to the war.
3. LivePerson Class: Represents an individual who survived the war.
4. Family Class: Represents a family unit affected by the war. A family is made up of parents (mom and dad) and siblings (son and daughter).
5. Manager Class: Manages the system's data and functionalities.

UML Diagram:


Driver
+main(args : String[]) : void
Manager
-families: ArrayList<Family>
+addFamily(family: Family): boolean
+updateFamily(familyName: String, updatedFamily: Family): boolean
+deleteFamily(familyName: String): boolean
+searchByName(familyName: String): Family
+searchPersonByID(personID: String): Person
+calculateTotalMartyrs(): int
+calculateTotalOrphans(): int
+calculateTotalLivePersons(): int
+calculateFamilyStatistics(familyName: String): ArrayList<Integer>
+calculateGlobalStatistics(): : ArrayList<Integer>
+toString(): String
Family
- familyName: String
- members: ArrayList<Person>
- parents: ArrayList<Person>
+addMember(member: Person, role: String): boolean
+removeMember(member: Person): boolean
+getMembers(): ArrayList<Person>
+addParent(parent: Person): void
+getFamilyName(): String
+setFamilyName(familyName:String): void
+removeParent(parent: Person): boolean
+getParents(): ArrayList<Person>
+toString(): String
+equals(obj: Object): boolean
Person
-ID: String
-Name: String
-Age: int
-Gender: String
-Address: String
-ContactInfo: String
LivePerson
Martyr
- DateOfMartyrdom:String
-CauseOfDeath: String
- PlaceOfDeath: String

<u>Additional details for methods in each class:</u>

You are required to implement the following method in each class, following its respective description

<u>Person Class</u>

- Appropriate getter and setter methods.
- toString(): Override this method to provide a meaningful string representation of the object.

<u>Martyr Class</u>

- Appropriate getter and setter methods.
- toString(): Override this method to provide a meaningful string representation of the object.

<u>LivePerson Class:</u>

- toString(): Override this method to provide a meaningful string representation of the object.

<u>Family Class:</u>

- addMember(Person member, String roleInFamily): Adds a person to the family with a specified role (The permitted roles include mom, dad, son, and daughter).
- removeMember(Person member): Removes a person from the family.
- getMembers(): Retrieves the list of family members.
- getFamilyName(): Retrieves the Family Name.
- setFamilyName(String familyName): set a name for family.
- addParent(Person parent): Adds a parent to the family.
- removeParent(Person parent): Removes a parent from the family.
- getParents(): Retrieves the list of family parents.
- toString(): Override this method to provide a meaningful string representation of the object.
- equals(Object obj): Override this method to consider two families as equal if they have the same number of martyrs.

<u>Manager Class:</u>

- addFamily(Family family): Adds a new family to the system.
- updateFamily(String familyName, Family updatedFamily): Updates information about a family based on Family name.
- deleteFamily(String familyName): Deletes a family from the system based on Family name.
- searchByName(String familyName): Searches for a family based on Family name.
- searchPersonByID(String personID): Searches for a person based on their ID.
- calculateTotalMartyrs(): Returns the total number of martyrs in the system.
- calculateTotalOrphans(): Returns the total number of orphans in the system. Orphans are individuals whose parents (both mom and dad) have passed away.
- calculateTotalLivePersons(): Returns the total number of live persons in the system.

- calculateFamilyStatistics(String familyName): Returns statistics for a specific family, including the number of martyrs, orphans, and live persons. Store the returned values in ArrayList.
- calculateGlobalStatistics(): Returns overall statistics for the system. Store the returned values in ArrayList.
- toString(): Override this method to provide a meaningful string representation of the object.

A sample for the main method:

```
 public static void main(String[] args) {
     // Create a manager
     Manager manager = new Manager();

    // Create persons
    Person person1 = new Person(Complete the function call with suitable inputs);
    Person person2 = new Person(Complete the function call with suitable inputs);

   // Create martyrs
    Martyr martyr1 = new Martyr(Complete the function call with suitable inputs);
    Martyr martyr2 = new Martyr(Complete the function call with suitable inputs);

  // Create live persons
    LivePerson livePerson1 = new LivePerson(Complete the function call with
                                            suitable inputs);
    LivePerson livePerson2 = new LivePerson(Complete the function call with
                                            suitable inputs);

  // Create families
     Family family1 = new Family(Complete the function call with suitable inputs);
     family1.addMember(Complete the function call with suitable inputs
     family1.addMember(Complete the function call with suitable inputs);
     family1.addMember(Complete the function call with suitable inputs);
     family1.addMember(Complete the function call with suitable inputs);

     Family family2 = new Family(Complete the function call with suitable inputs);
     family2.addMember(Complete the function call with suitable inputs, "dad");
     family2.addMember(Complete the function call with suitable inputs, "Mom");

   // Add families to the manager
      manager.addFamily(family1);
      manager.addFamily(family2);

   // Display global statistics
      System.out.println("Global Statistics:");
      System.out.println(manager.calculateGlobalStatistics());

  // Display family statistics
     System.out.println("\nFamily Statistics (Family Name):");
     System.out.println(manager.calculateFamilyStatistics(Complete the function
                                                      call with suitable inputs));
}
```

Please note the Followings:

1. Your program **should be well commented** based on Java formal documentation.
2. Input all data **from the console** and ensure that **the output is displayed on the console**. You may also need to incorporate useful **display methods at appropriate locations for printing the results or data on the console**.
3. This is an **individual assignment**. Disciplinary action will be taken against those who **cheat**. Additionally, the use of **AI tools** for generating solutions or **copying from websites** is strictly prohibited. Students found in violation of these policies will face severe consequences. It is crucial to ensure that all work submitted is **your own** and adheres to the **guidelines provided for this assignment**.
4. **Make sure you submit your project through ITC before the deadline.**
5. **Due date is Saturday, 2/12/2023, at 11:00 pm.**

"Success is the sum of small efforts, repeated day in and day out."

Robert Collier

Best of Luck