BIRZEIT UNIVERSITY

Faculty of Engineering and Information Technology
Computer Science Department
COMP2311 Project (Phase 2)

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

Project objectives: Apply advanced object-oriented programming concepts in Java, encompassing classes, inheritance, polymorphism, abstract classes, interfaces, deep copy implementation, and exception handling, while also integrating file operations.

Project Description:

In this phase of the project, you are responsible for adding different features to improve phase 1 and make it more logical. This time, abstract classes, interfaces, exception handling, copying objects, and reading/writing from files will be utilized.

Additional features and updates to consider for this phase:

- Modify the **Person class** to be **abstract**.
  - Do any necessary changes.

- Create an interface named **Sortable**, as shown below, to handle sorting functionality (sorting should be in **descending order**).

  - Do any necessary updates (you can import any necessary packages).

```
public interface Sortable {

  /* The sorting should be based on the number of Martyrs. Keep the original
     family list as it is. */

 public  ArrayList<Family> sortByMartyrs(ArrayList<Family> families);

  /* The sorting should be based on the number of Orphans. Keep the original
     family list as it is. */

 Public Array<Family>  sortByOrphans (ArrayList<Family> families);

}
```

- Let the Family class implement the Sortable interface. Ensure that creating an object from the Family class is still necessary.
  - Do any necessary updates.

- Implement a **deep copy** mechanism that enables making copies of **Martyr, LivePerson**, and **Family objects**.
  - Identify the suitable class(s), where you need to implement this method.

- Introduce exceptions to check any invalid inputs. For example, invalid age, incorrect type, among other exceptions.
    - You are responsible for checking **any exceptions** that may occur in your application.
- Introduce **custom exceptions** to address specific scenarios, such as ensuring that the parents ArrayList in the Family class includes only two persons and preventing the addition of family members to the members ArrayList without including parents in the parents ArrayList.
    - Create classes as you want.
- Modify the program to enable the user to read input data from both a **text file** and **console**, and write the output to both a text file and console.
    - Do any necessary updates.
- Implement a menu that allows users to choose from various functions. For instance, provide options like:

This menu is just an example; manage it as you find suitable.

1) Create a new family
2) Add parents to a family
3) Add a family members
.
.
...

n-2) Print to the file all families with their members in sorted order. First, list the name of the family, followed by the parents, and then the members of the family.

Example for printing output to file:

Family 1: number of martyrs (5)
.....
Parents: Dad name, mom name
Members: members names

Family 2: number of martyrs (3)
.....
Parents: Dad name, mom name
Members: members names
:

- This is the format for organizing your file and is not part of the menu.
- The file is sorted based on the number of martyrs in descending order.

n-1) Copy an object of type Martyr ...
n) Exit

    - Organize the menu options logically and suitably to encompass all the functionalities present in different classes. The menu should offer a comprehensive set of choices reflecting the diverse operations available.
- Let the user decide whether they want to read/write the data from the **console** or from a **text file**.

Students may ask some questions; here are the answers in advance 😊:

1. Please don't ask about file reading/writing methods, formats, or criteria. We encourage you to be autonomous in handling these aspects.

2. Don't ask if it's possible for us to add new methods or classes. Sure, you can, but don't use anything outside of the course material. Using anything not covered in the course will be treated as cheating.

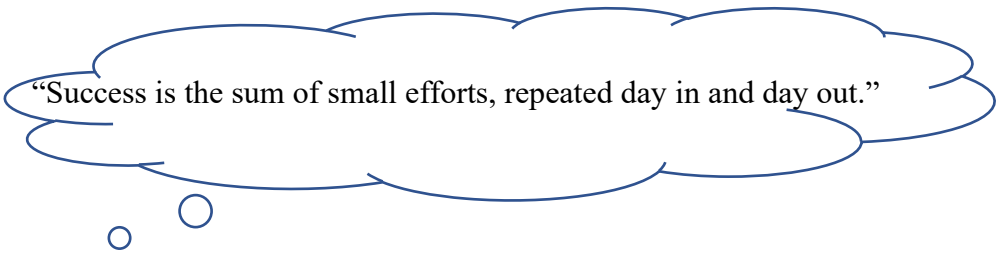
3. We have taught you various guidelines to follow when creating a Java program. Although these guidelines are not explicitly mentioned in this project document, they are present in the lecture notes. You are responsible for following them, including adding no-argument constructors, proper spacing, indentation, comments, and organizing code, among other things. We will ask you about these aspects.

4. Don't inquire about what needs to be tested in the driver class. You have a menu with different options; use this menu to test all the functionalities you have implemented.

5. For any questions not mentioned above, we encourage you to be autonomous and find the solutions yourself. We aim to teach you how to think about problems and find solutions independently.

Please note the Followings:

1. 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**.
2. Make sure you submit your project through **ITC** before the deadline (Submit all **Java files (.java) and your text files (.txt)**).
3. Due date is Wednesday 3/01/2023, at 11:00 pm.



Robert Collier

Best of Luck