BIT203 Assignment 1

Release Date:	Week 4, 27/10/2023
Due Date:	Week 8, 25/11/2023, 4 PM
Value:	15%
Assessment Mode:	Individual Assignment

Rationale

This assignment has been designed to allow students to test and demonstrate their ability to write a Java program that uses a range of different concepts and facilities. This assessment relates to the following learning outcomes:

Expected Learning Outcomes Assessed

- CLO1 write programs using several classes based on UML class diagrams and other models
- CLO2 apply object-oriented concepts in the design and implementation of the programs

In particular, this assignment tests the students ability to use appropriate class hierarchies and collection classes.

HELP Aid

Due to the covid-19 pandemic and extended lockdowns, there is much economic hardship faced by communities all over the world. Fortunately, many initiatives have been developed to help such as the "White Flag" and "Kita Jaga Kita" movements (Tan and Khoo, 2021; Mohammad Radhi, 2021). These initiatives aim to deliver aid whether in the form of cash, food or services to the underprivileged.

HELP Aid is an information system that has been proposed to manage the registration of aid applicants and organizations that are helping them. Donors who want to contribute can then find information about the various organizations and their appeals for aid and make contributions which are either cash donations or goods. **Donor information is not captured**, however the organizations would have to record the contributions received and the disbursements given.

The use case diagram is shown in Fig. 1.

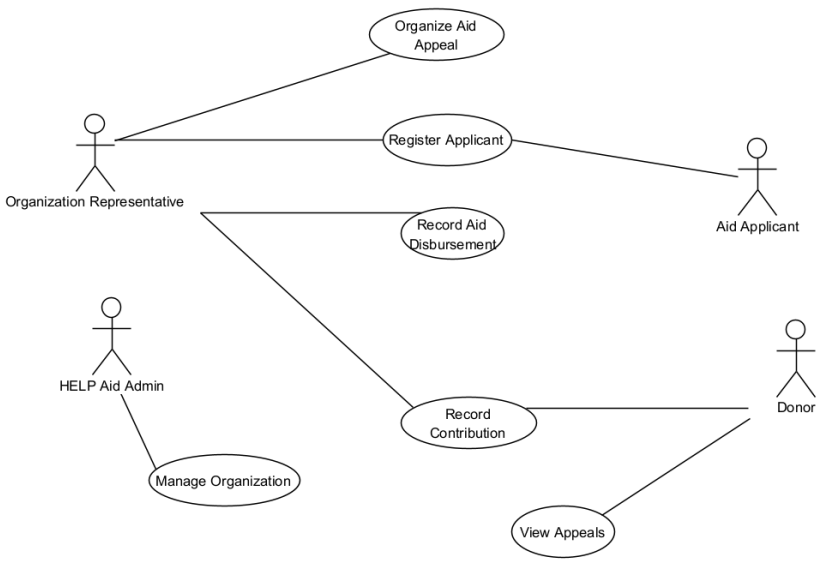

Figure 1: HELP Aid Use Case Diagram

The class diagram is shown in Fig. 2.

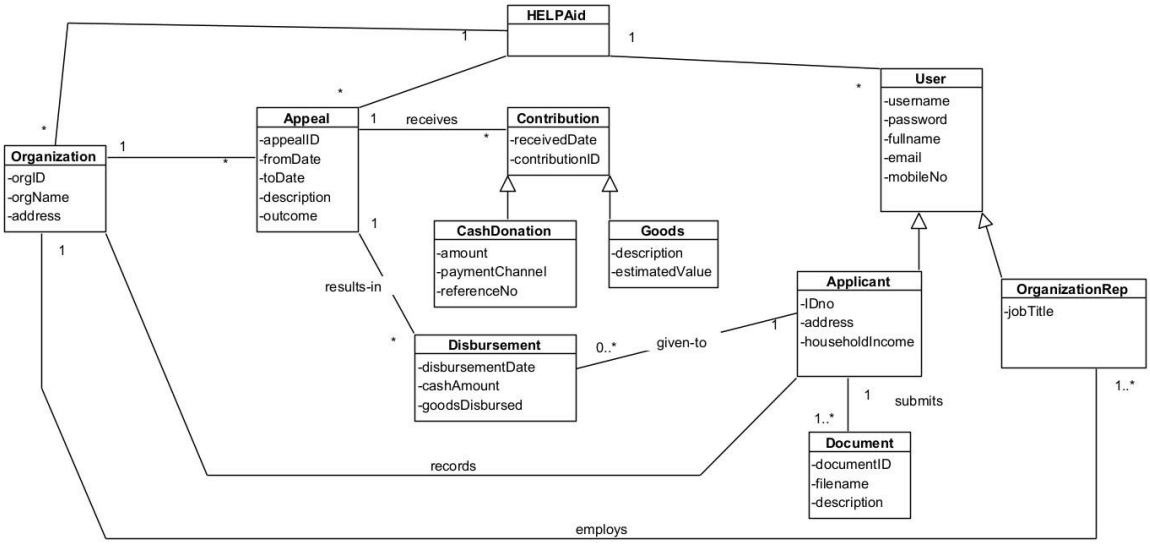

Figure 2: HELP Aid Class Diagram

The Conceptual Model shown above depicts the relationship that exists between classes in this assignment. The controller class HELPAid must maintain at least a collection of User objects, a collection of Organization objects, and a collection of Appeal objects. Note that User is an abstract class, and is the superclass to the two concrete subclasses – Applicant and OrganizationRep. For all classes, you may include additional methods if you wish to do so, for example the *toString*, and *equals* methods. Two users are considered equal if their *username* are the same.

The use cases are given briefly below:

Use Case 1	Manage Organization
Goal in Context	To allow the HELP Aid Admin to manage Organizations and their representatives.
Primary Actor	HELP Aid Admin
Trigger	The HELP Aid Admin wants to manage organization details
Typical Course of Events Actor Action	**System Response**
1. This use case begins when an HELP Aid Admin wants to add a new organization representative	
2. The HELP Aid Admin selects the organization ID.	The organization's name is displayed
3. The HELP Aid Admin enters the username, fullname, mobile no and jobtitle for the Organization Representative.	The Organization Representative is recorded for the organization. An email is sent to the Organization Representative with a unique username plus a generated default password to inform them that they have been registered into the system.

Alternative Course of Events
Line 2a. If the organization is not registered, the HELP Aid Admin enters the organization's name and address. A new organization ID is generated and the new organization is recorded.
Line 3a. If the HELP Aid Admin would like to add another representative, repeat line 3.

Use Case 2	Register Applicant
Goal in Context	To allow the Organization Representative to register Aid Applicants
Primary Actor Secondary Actor	Organization Representative Aid Applicant
Trigger	The Organization Representative wants to record a new applicant, or the applicant wants to register under an organization.

Typical Course of Events Actor Action	**System Response**
1. This use case begins when an Organization Representative wants to register an Aid Applicant who has come to the Organization's premises.	
2. The Organization Representative logs in with a valid username and password.	The organization name is shown.
3. The Organization Representative enters the full name, IDno, address, household income of the aid applicant.	A username and password is generated for the Aid Applicant. The Aid Applicant is recorded for the organization.
4. The Organization Representative uploads a document with the Aid Applicant's proof of household income by entering the filename and description.	The document is recorded for the Aid Applicant.

Alternative Course of Events
Line 1a. If the Aid Applicant is able to register themselves online, the Aid Applicant selects an organization from the list of organizations, then skip to line 3 to complete the registration. An Applicant can only be recorded with one organization.
Line 4a. If there are additional documents, repeat line 4.

Use Case 3	Organize Aid Appeal
Goal in Context	To allow the Organization Representative to record a new appeal for Aid.
Primary Actor Secondary Actor	Organization Representative
Trigger	The Organization Representative wants to record a new appeal for aid.

Typical Course of Events Actor Action	**System Response**
1. This use case begins when an Organization Representative wants to record a new appeal for aid.	
2. The Organization Representative logs in with a valid username and password.	The organization name is shown.
3. The Organization Representative enters the from date, to date and description of the appeal.	The appealID is automatically generated.

Use Case 4	View Appeals
Goal in Context	To allow Donors to view appeals for Aid
Primary Actor	Donor
Trigger	A Donor wants to check if there are any appeals

Typical Course of Events Actor Action	**System Response**
1. This use case begins when a Donor wants to check if there are any appeals	.
2. The Donor selects to view current appeals.	A list of appeals is shown with the from date, to date and description.
3. The Donor selects one of the appeals.	The organization name and address is shown.

Alternative Course of Events
Line 2a. The Donor may select to view past appeals. The from date, to date, description and outcome is shown.

Use Case 5	Record Contribution
Goal in Context	To record contributions from Donors.
Primary Actor	Organization Representative Donor
Trigger	A Donor wants to make a contribution, either online or at the organization's premises.

Typical Course of Events Actor Action	**System Response**
1. This use case begins when a Donor wants to make a contribution.	
2. The Organization Representative logs in with a valid username and password.	The organization name is shown with a list of current appeals.
3. The Organization Representative selects the appeal ID for the contribution.	The from date and to date of the appeal is shown.
4. The Organization Representative records the description and estimated value of the goods.	The Goods contribution is recorded for the appeal. A contributionID is generated and receivedDate is set to the current date.

Alternative Course of Events
Line 2a: If the Donor is contributing cash through the payment provider, skip to line 3 to enter the appealID.
Line 4a. If the Donor is contributing cash, the cash amount, payment channel and referenceNo is recorded. A contribution ID is generated and the received date is set to the current date.

Use Case 6	Record Aid Disbursement
Goal in Context	To record disbursement of aid to applicants.
Primary Actor	Organization Representative
Trigger	The Organization Representative wants to record aid that has been disbursed.

Typical Course of Events Actor Action	**System Response**
1. This use case begins when the Organization Representative wants to record the disbursement of aid.	

2. The Organization Representative logs in with a valid username and password.	The organization name is shown with a list of current appeals.
3. The Organization Representative selects the appeal ID.	The from date and to date of the appeal is shown, together with the list of contributions and the details of the contributions.
4. The Organization Representative selects to view Applicants.	A list of Applicants registered with the Organization is shown.
5. The Organization Representative selects the IDno of an applicant.	The applicant's name, address and household income is shown.
6. The Organization Representative enters the disbursement date, cash amount and the goods to be provided to the applicant.	The disbursement is recorded for the Applicant.
7. The Organization Representative records the outcome of the Appeal.	
Alternative Course of Events	
Line 5a. The Organization Representative may choose to view the Applicant's documents.	
Line 6a. If there are additional disbursements to applicants, repeat lines 5 and 6.	

References:

Mohamed Radhi, N.A. (2021, June 29). New #benderaputih movement, to assist those in need. *New Straits Times.* https://www.nst.com.my/news/nation/2021/06/703342/new-benderaputih-movement-assist-those-need

Tan, R. and Khoo, N. (2021, July 1). #KitaJagaKita: Covid-19 initiatives and food drives in Malaysia to support. Buro. https://www.buro247.my/lifestyle/news/covid-19-coronavirus-malaysia-initiatives.html

Assignment Requirements:

User interface specifications

A Java application fulfilling the role of a user interface should be initiated in a class called `HELPAidConsole`. This class is not shown in the diagram (it is not a problem domain class) but is a view class that interacts with the `HELPAid` controller. The class should provide a *console style* user interface. That is, all output for the user should be directed to standard output (and appears on the screen) and all input should be obtained from standard input (read from the keyboard).

NOTE:

1. Prior to the display of menu, a ‘single’ working HELPAid object should be created, and the HELP Aid Admin is created.
2. The details of some tasks are shown in the high level use cases in previous pages.

This console interface class will provide a menu that allows a user of your program to perform the following operations:

- allows HELP Aid Admin to manage Organization;
- ADD an Applicant to the created HELPAid object;
- ADD an OrganizationRep to the created HELPAid object;
- allows an OrganizationRep to organize aid Appeal;
- allows an OrganizationRep to record contribution;
- allows an OrganizationRep to record aid Disbursement;
- allows a Donor to record contribution;
- allows a Donor to view aid Appeal;
- allows an Applicant to view given Disbursement;
- display detail of all users, either in original sequence, or sorted according to fullname;
- display detail of all aid Appleals, sorted according to Organization name, followed by aid Appeals (sorted according to from date).

The `HELPAidConsole` class must contain the application's **main method** so that the application can be launched with a command equivalent to

```
java HELPAidConsole
```

User interaction and output

It is a specific requirement of this assignment that **none** of the problem domain classes listed above may contain any user interaction code, including the reading of values from a keyboard or interactive input device and none of the problem domain classes may generate any output for the user, such as screen messages or prompts.

It is acceptable (encouraged) to have screen output messages generated by problem domain classes for **debugging** purposes. These can be very useful during the implementation and testing phases of development. Debugging code should be removed or commented out of all problem domain classes in the final version of the application.

Design and implementation

In completing this assignment you should carefully consider the design of the system, one use case at a time. Once you have coded each class, test its functionality completely. You should only make use of 'getters' and 'setters' to access and alter attributes, and provide a toString method for each class.

You will need to design for and include the appropriate collection objects as well. You will also need to ensure that the data that is input by users is valid.

Coding style and comments

Your source code should be clear and readable, using correct indentation, meaningful identifiers and comments. Include javadoc comments and tags as follows:

- for public classes, to indicate their purpose;

- for public methods, to indicate their effect, parameters and return values;
- for public fields, to indicate their purpose.

You will be expected to generate and submit the javadoc documentation for your classes as part of this assignment – included inside the submitted compressed file (together with the Java source code).

SUBMISSION REQUIREMENT

Your assignment has to submit to TurnItIn:

1. All your Java source files, printed in Word document format
2. Printed output (showing your interactivity with your program) is to be included at the end of your Java source files, in Word document created in (1)
3. Generate a runnable jar file of your project, and compress the jar file together with all your java source files into ONE file.
4. Submit your solutions to the Turnitin link created in LMS:
   - The Word document created in (1) is uploaded to Part 1 in Turnitin.
   - The compressed file created in (3) is uploaded to Part 2 in Turnintin.

Marking Scheme

Refer to the Excel file, *203A1MS_SO23.xlsx*, for detailed breakdown of the marks allocated for the requirements.

Note:

If your program does not meet the requirements by the due date you should obtain help from the lecturer and notify the lecturer that you will submit the assignment late (marks will be deducted).

Note about testing and plagiarism

It is very important that you **complete** this assignment **alone**. You may of course obtain general assistance from the lecturing staff in the subject and your peers, but the coding must be carried out yourself. It is normally quite easy to detect when two or more students work together on their coding.

It is also very important that the demonstration of the results of your program using the given test data is produced using the identical version of the program to the printout of your source code. **Students who hand in substantially similar assignments or whose programs do not match their demonstration of testing will fail the assignment.**

Any student suspected of copying, or of not producing the work himself or herself, can be called for **oral examination**, where the student will be expected **to demonstrate sufficient knowledge of the application** to show that it is his or her own original work.