Lab 1: Object-oriented Programming

Instruction

1. Click the provided link on CourseVille to create your own repository.
2. Open Eclipse and then “File > new > Java Project” and set project name in this format **2110215_Lab1_2020_2_{ID}_{FIRSTNAME}**
    - Example: **2110215_Lab1_2020_2_6131234521_Samatcha**.
3. Initialize git in your project directory
    - Add .gitignore.
    - Commit and push initial codes to your GitHub repository.
4. Implement all the classes and methods following the details given in the problem statement file which you can download from CourseVille.
    - You should create commits with meaningful messages when you finish each part of your program.
    - Don’t wait until you finish all features to create a commit.
5. Test your codes with the provided JUnit test cases, they are inside package **test.grader**
    - If you want to create your own test cases, please put them inside package **test.student**
    - Aside from passing all test cases, your program must be able to run properly without any runtime errors.
6. After finishing the program, create a UML diagram using **ObjectAid** and put the result image (UML.png) at the root of your project folder.
7. Export your project into a jar file called **Lab1_2020_2_{ID}** and place it at the root directory of your project.
    - Example: **Lab1_2020_2_6131234521.jar**
8. Push all other commits to your GitHub repository

1. Problem Statement: Guild Member Database

You have been transported into a fantasy world by a magical truck, and was found by a desperate guildmaster who does not know how to use computers. As the only person in this fantasy world who knows how to use a computer, you are asked to implement a system that allows him to manage guild data, including member information and department information.

The program example is shown below. The program should be run from the Main class in the package main.

```
===== Welcome to the Guild Member Database! =====
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
```

There are 6 options to do.

1. View Departments and Members. By default, the first department (department 0) is the Unassigned Department. It is used to contain members that are not assigned to any departments.

```
===== Welcome to the Guild Member Database! =====
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
0
====================
Choose a Department:
0) Unassigned Department
1) Exit
```

2. New Department. Creates a new department of the guild. However, department name cannot be duplicate. If a department with a certain name already exists, then it will be rejected. Department name also cannot be blank.

```
====================
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
1
====================
Choose a new department name:
Fighters
Fighters Department created!
====================
```

```
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
1
====================
Choose a new department name:
Fighters
Department name cannot duplicate!
Failed to create department. Returning to Main Menu.
====================
```

```
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
1
====================
Choose a new department name:

Department name cannot be blank!
Failed to create department. Returning to Main Menu.
====================
```

3. Remove Department. Removes a chosen department. If the removed department has members in it, all of the members in that department will be moved to the unassigned department at index 0.

```
====================
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
2
====================
Choose a department to remove:
0) Unassigned Department
1) Fighters Department
2) Cancel
1
===== Fighters Department =====
This department is empty!
====================
Are you sure you want to remove this department? All members will be moved to Unassigned Department. (y/n)
y
Department removed. 0 member(s) added to Unassigned Department.
====================
```

4. New Member. Adds a new guild member to the list by inputting name, job title, and salary. If left blank, the program will input “Anon”, “Adventurer”, and “0” respectively. Then, the user chooses the department to put the new member in.

```
====================
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
3
====================
Please input name:
Joseph
Please input job title:
Hermit Purple
Please input salary:
300
Choose a department:
0) Unassigned Department
0
Joseph the Hermit Purple of the Unassigned Department (Salary: 300) added.
====================
```

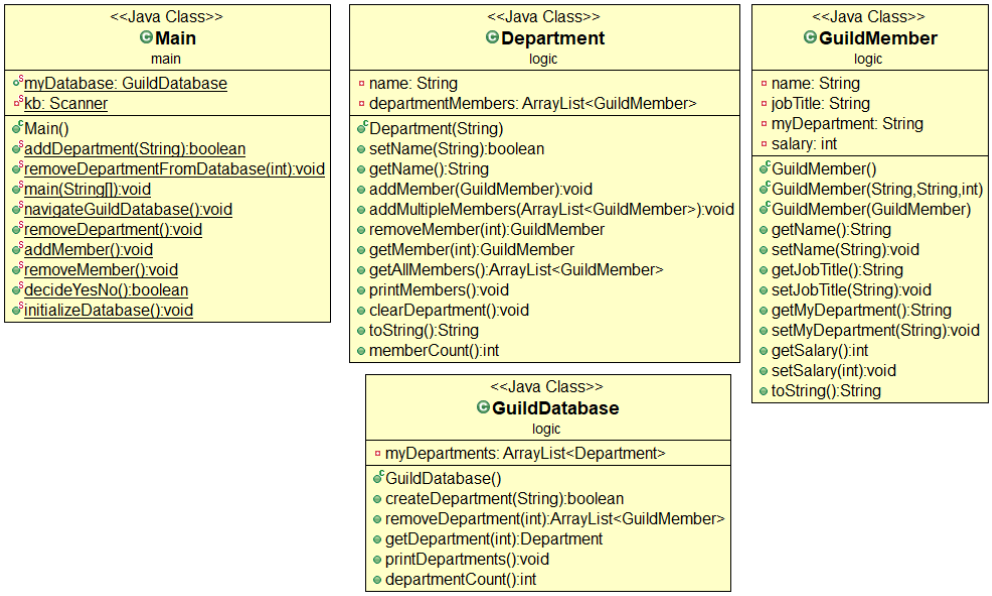
5. Remove Member. Simply removes a member from the guild. Unlike remove department, removed members are permanently removed.

```
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
4
====================
Choose a department:
0) Unassigned Department
1) Cancel
0
===== Unassigned Department =====
Choose a member to remove:
0) Joseph the Hermit Purple of the Unassigned Department (Salary: 300)
1) Back
0
Are you sure you want to remove this member? (y/n)
y
Member removed.
====================
```

6. Exit. Exits the program.

2. Implementation Details:

The diagram of the program is illustrated below. There are 4 classes: Main, GuildDatabase, Department, and GuildMember.



** Note that Access Modifier Notations can be listed below*
** Also note that while other methods have been provided, only methods that you have to implement are listed here*

+ (public)
 # (protected)
- (private)

2.1 Class GuildMember (package logic)

2.1.1 Fields

- String name	The member's name. Can contain space. Cannot be blank. If blank, will be set to "Anon".
- String jobTitle	The member's job title. Can contain space. Cannot be blank. If blank, will be set to "Adventurer".
- String myDepartment	The member's department.
- int salary	The member's salary. Must be greater than 0 and lesser than 100000.

2.1.2 Constructors

+ GuildMember(String, String, int)	**/*Fill Code*/** Initialize **all fields**. For Department, set it as "Unassigned" here. **Note: You should use setter to set the value of all fields in order to handle negative value case and special case.**

2.1.3 Methods

+ void setName(String)	**/*Fill Code*/** This method will set name of the member. If the String is empty, the method will set the name as "Anon". **Hint: There is a non-static method called isBlank() in the String class. It returns true if a String consists of only whitespace.** **Example:** **String exampleString = " ";** **exampleString.isBlank() will return true.**

+ void setJobTitle(String)	**/*Fill Code*/** This method will set the job title of the member. If the String is empty, the method will set the job title as “Adventurer”.
- void setSalary(int)	**/*Fill Code*/** This method will set the salary of the member. If the salary is less than 0, it will be set as 0. If the salary is greater than 100000, it will be set as 100000.
Getter & Setter methods for all remaining variables	**/*Fill Code*/**

2.2 Class *Department (package logic)*

2.2.1 Fields

- String name	The name of the department. It should never be empty.
- ArrayList<GuildMembers> departmentMembers	The GuildMembers in this department, contained in a list.

2.2.2 Constructors

+ Department(String)	**/*Fill Code*/** Initialize the list, as well as set the name of the department. **Note: You should use setter to set the value of all fields in order to handle negative value case and special case.**

2.2.3 Methods

+ boolean setName(String)	**/*Fill Code*/** Set the name of the department and returns

	whether or not the name change was a success. - If the name is not blank, then change the department name and return true. - If the name is blank, then **DO NOT** change the department name and return false.
+ String getName()	**/*Fill Code*/** Returns the name of the department.
+ void addMember(GuildMember)	**/*Fill Code*/** Add the given member to the list of members, as well as call setMyDepartment(String) to set their department to match this department's name.
+ void addMultipleMembers(ArrayList<GuildMember>)	**/*Fill Code*/** Add all the members in the given arraylist to this list. Make sure you call setMyDepartment(String) to set all of their deparments to match this department's name.
+ GuildMember removeMember(int)	**/*Fill Code*/** Removes a member from the given index from the list of members, and return the removed member.
+ GuildMember getMember(int)	**/*Fill Code*/** Returns a member from the given index.
+ ArrayList<GuildMember> getAllMembers()	**/*Fill Code*/** Returns all members in the department.

2.3 Class GuildDatabase (package logic)

2.3.1 Fields

- ArrayList<Department> myDepartments	The list of departments contained in this database.

2.3.2 Constructors

+ GuildDatabase()	**/*Fill Code*/** Initialize myDepartments as an empty ArrayList.

2.3.3 Methods

+ boolean createDepartment(String)	**/*Fill Code*/** Attempts to create a new department with the given name. Returns true or false depends on whether or not a department was successfully created. - If the department's name duplicates another department inside myDepartments, **DO NOT** create the department and return false. - Otherwise, create a new department with the given name and return true.
+ boolean isExists(String)	**/*Fill Code*/** This method runs a check with all departments currently in the databas, using the String given. - If there is a department with the same name as the given name, return true. - If there are no departments with the same name as the given name, return false.
+ ArrayList<GuildMember> removeDepartment(int)	**/*Fill Code*/** Remove a department from the given index, and return a list of all members that was in that department before it got removed.

2.4 Class Main (Static class): (package Main)

Details on DuplicateGuildNameException and EmptyGuildNameException will be given to you in the next part.

2.4.1 Fields

+ GuildDatabase myDatabase	**/*Fill Code*/** The database system.
+ Scanner kb	**/*Fill Code*/** Scanner for getting input.

2.4.2 Constructors

This class does not need a constructor.

2.4.3 Methods

public static boolean addDepartment(String)	**/*Fill Code*/** Create a new department using the given name. Returns true or false depending on whether or not the department is successfully created. - If the given name is blank, **DO NOT** create the department. Throw EmptyGuildNameException and return false, as well as print the exception to tell the user that the name cannot be blank. - If the given name duplicates a department's name that already exists, **DO NOT** create the department. Throw DuplicateGuildNameException and return false, as well as print the exception to tell the user that the name cannot be duplicate. - Otherwise, create the department and return true. NOTE: Program should still be running after throwing exception. It should return you to the main menu.

	NOTE 2: Don't throw exceptions in the method's declaration. Instead, you should use try/catch to throw exceptions here.
public static void removeDepartmentFromDatabase(int)	**/*Fill Code*/** Remove a department at the given index from the database, and put all the members that used to be in that department into the Unassigned department, which can be called with the getUnassignedDepartment() method. Finally, confirm the user the number of members that are moved to the unassigned department.

2.5 Exception Classes (package exception)

There are two exception classes provided for you in the package exception.

- EmptyGuildNameException should be thrown when inputting an empty guild name. When printed, it will display an error message that says guild name should not be empty.
- DuplicateGuildNameException should be thrown when inputting a duplicate guild name. When printed, it will display an error message that says guild name should not be duplicate.
- To throw an exception, do:
    - throw new EmptyGuildNameException();
    - throw new DuplicateGuildNameException();

3. Test Scenario

(User input is in green.)

```
===== Welcome to the Guild Member Database! =====
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
1
====================
Choose a new department name:
Animals
Animals Department created!
====================
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
3
====================
Please input name:
Pikachu
Please input job title:
Electric Mouse
Please input salary:
25
Choose a department:
0) Unassigned Department
1) Animals Department
1
Pikachu the Electric Mouse of the Animals Department (Salary: 25) added.
====================
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
3
====================
Please input name:
Ed
Please input job title:
Talking Horse
Please input salary:
40
Choose a department:
0) Unassigned Department
1) Animals Department
1
Ed the Talking Horse of the Animals Department (Salary: 40) added.
====================
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
3
====================
Please input name:
Edward
Please input job title:
Talking Dog
Please input salary:
50
Choose a department:
0) Unassigned Department
1) Animals Department
1
Edward the Talking Dog of the Animals Department (Salary: 50) added.
====================
```

```
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
1
====================
Choose a new department name:
Swordfighter
Swordfighter Department created!
====================
 ===== MAIN MENU =====
 0) View Departments and Members
 1) New Department
 2) Remove Department
 3) New Member
 4) Remove Member
 5) Exit
 3
 ====================
 Please input name:
 Link
 Please input job title:

 Please input salary:
 999
 Choose a department:
 0) Unassigned Department
 1) Animals Department
 2) Swordfighter Department
 2
 Link the Adventurer of the Swordfighter Department (Salary: 999) added.
 ====================
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
3
====================
Please input name:
Zelda
Please input job title:

Please input salary:
999
Choose a department:
0) Unassigned Department
1) Animals Department
2) Swordfighter Department
2
Zelda the Adventurer of the Swordfighter Department (Salary: 999) added.
====================
```

```
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
4
====================
Choose a department:
0) Unassigned Department
1) Animals Department
2) Swordfighter Department
3) Cancel
2
===== Swordfighter Department =====
Choose a member to remove:
0) Link the Adventurer of the Swordfighter Department (Salary: 999)
1) Zelda the Adventurer of the Swordfighter Department (Salary: 999)
2) Back
1
Are you sure you want to remove this member? (y/n)
y
Member removed.
====================
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
2
====================
Choose a department to remove:
0) Unassigned Department
1) Animals Department
2) Swordfighter Department
3) Cancel
1
===== Animals Department =====
0) Pikachu the Electric Mouse of the Animals Department (Salary: 25)
1) Ed the Talking Horse of the Animals Department (Salary: 40)
2) Edward the Talking Dog of the Animals Department (Salary: 50)
====================
Are you sure you want to remove this department? All members will be moved to Unassigned Department. (y/n)
y
Department removed. 3 member(s) added to Unassigned Department.
====================
```

```
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
0
====================
Choose a Department:
0) Unassigned Department
1) Swordfighter Department
2) Exit
0
===== Unassigned Department =====
0) Pikachu the Electric Mouse of the Unassigned Department (Salary: 25)
1) Ed the Talking Horse of the Unassigned Department (Salary: 40)
2) Edward the Talking Dog of the Unassigned Department (Salary: 50)
====================
Choose a Department:
0) Unassigned Department
1) Swordfighter Department
2) Exit
1
===== Swordfighter Department =====
0) Link the Adventurer of the Swordfighter Department (Salary: 999)
====================
```

4. Exception Scenarios

Throwing DuplicateGuildNameException

```
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
1
====================
Choose a new department name:
Minecraft YouTubers
Minecraft YouTubers Department created!
====================
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
1
====================
Choose a new department name:
Minecraft YouTubers
exception.DuplicateGuildNameException: Department name cannot duplicate!
Failed to create department. Returning to Main Menu.
====================
```

Throwing EmptyGuildNameException

```
===== MAIN MENU =====
0) View Departments and Members
1) New Department
2) Remove Department
3) New Member
4) Remove Member
5) Exit
1
====================
Choose a new department name:

exception.EmptyGuildNameException: Department name cannot be blank!
Failed to create department. Returning to Main Menu.
====================
```