Dormitary Management System

Object Oriented Programing - Java

BEREKETEAB BIRKU..... 1303022

EZEDIN GULTE 1303690

KALEAB SHEWANHE..... 1305026

MELKAMU MISENE..... 1304233

SAMUEL MELKIE..... 1307444

March, 2023

Contents

Acknowledgement 1

Introduction 2

Problem Statemet 3

Implementation 4

Link of Project 6

Acknowledgement

We would like to address our acknowledgement to Mr. _____ for giving us this challenge. We were challenged to choose one problem of the University and it accordingly. And we chose to address the Student-Proctor communication issue. Working on the project has made us learn how to watch problems from the world and solve them.

We believe that our project will have an impact on minimizing the time wasted on some unwanted and repetitive tasks. So, as we believe this project will reduce this problem, we would like to improve it a lot and implement in on the University's server.

Much thanks for those who help us gain this skills.

Thanks,

Introduction

Having seen the problem in communication between students and proctors and receiving our OOP assignment, we thought on working on a simple project to reduce this problem with the skills we have learned.

This program digitalizes students dormitory registration, and simplify proctor-student interaction for many silly issues. It also help the student get status of the building, and on the future, will have many issues regarding to the dormitory environment.

Students and proctors sign-in, watch and change information, interact by reading and sending notification to and from others.

We have used and Object Oriented approach using the **Java programing language**. It was made with the *Apache NetBeans IDE 15* as a development IDE for writing the codes. Different Functions, Classes, Objects, are used for the project.

We have tried to document the program properly, and as it is a beginner's project, we would like to hear many comments as possible.

Thanks,

Problem Statement

To survive in campus, we need to make various interactions with different people to get allowances, ask and get help on various issues. Among this, students are required to meet with their building's proctor to properly survive on dormitory.

This Proctor-Student interaction is mainly done on face-to-face approach or through mobile calls. Dorm registrations are manual which are prone to redundancy and are not easily accessible.

As availability of some resources around campus needs manual checkup, they take very huge time and energy from the tiny limited time and energy of the student.

From this, we have tried to digitalize some portion of this interaction with the Java lesson that we take recently.

Implementation

Classes

For this problem, we used an Object Oriented approach. We used 6 classes to perform our required tasks. Within some of the classes, we have used abstract classes that need not be instantiated by the user.

{Person, Proctor, Student, UI, Authorization, DMS}

On landing, the *DMS* class *main* function calls the *UI* class which prints the landing options.

Person:

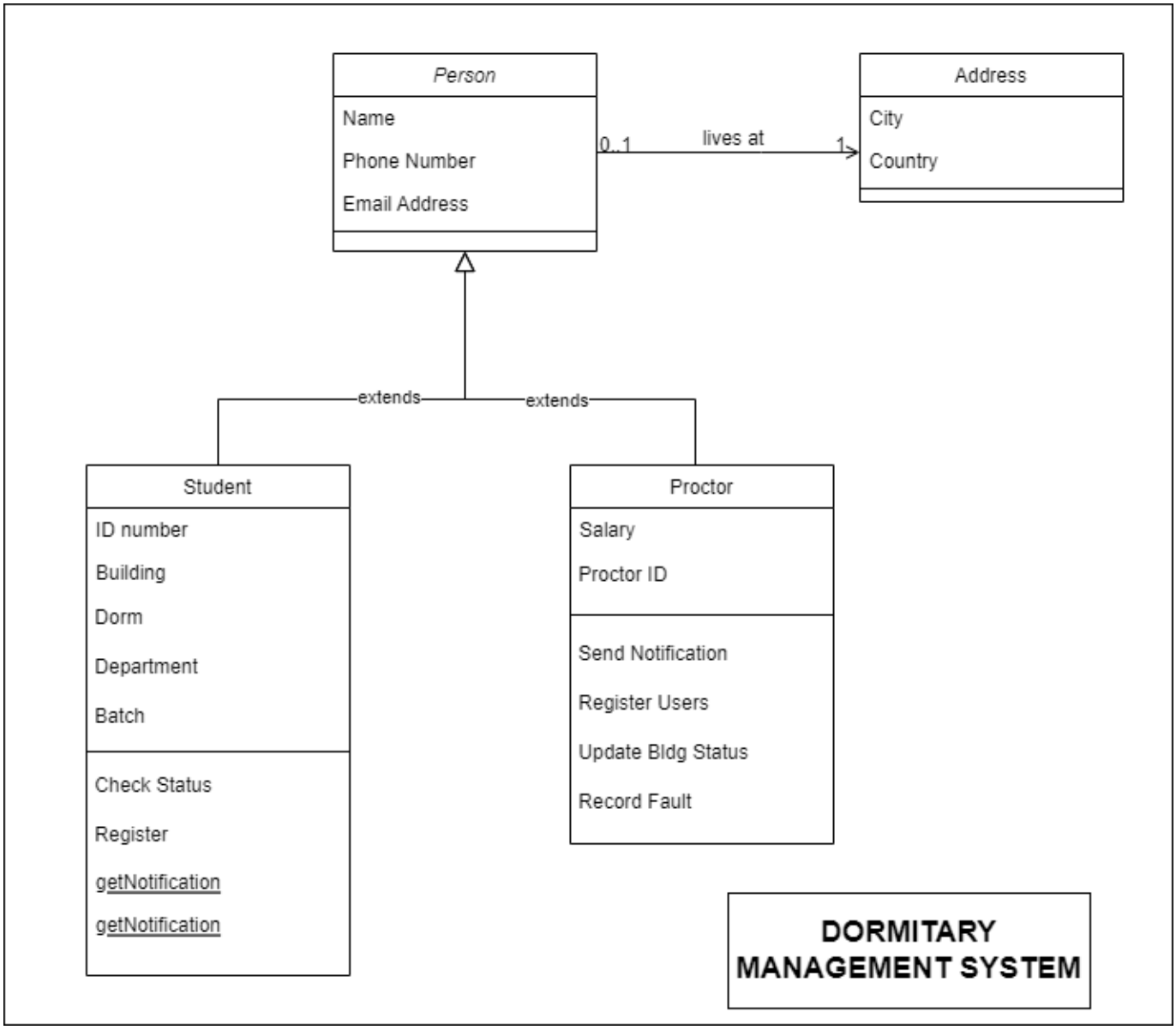
- Public
- Super Class extended by *Student* and *Proctor*

Student:

- Public
- Subclass that extends *Person*
- The *student* registers if the *Proctor* sets the *static regOpen* variable to 1, else the registration stays closed.
- *Checks Status* of the dormitory status by getting the *static variables* set by the *Proctor*.
- *Checks notifications* by *BufferedReading* the file written by the *Proctor*.

Proctor:

- Public
- Subclass that extends *Person*
- Signs In with his/her corresponding username and password.
  To login to the program, the *Proctor* class calls the class *Authentication* to get its password and username checked.
- Contains nested abstract class *BuildingStatus* which contains static variables of data related to the dormitory environment.
- The proctor updates the data by setting its corresponding variables *0* if the response is *Negative* and, 1, if *Positive*.
- The proctor allows Registration and closes it when the deadline passes.
- Proctor class *sends notification* and *record faults* by *Streaming* an output using *BufferedWriter* to a *file path* in which this path is then read by the *Student* class.
- The *BufferedWriter* and *BufferedReader* were surrounded by a try-catch error handling mechanism to check for errors.


Person
Name
Phone Number
Email Address
0..1
lives at
1
Address
City
Country
extends
extends
Student
ID number
Building
Dorm
Department
Batch
Check Status
Register
getNotification
getNotification
Proctor
Salary
Proctor ID
Send Notification
Register Users
Update Bldg Status
Record Fault
DORMITARY MANAGEMENT SYSTEM

Github link for the project:

https://github.com/BereketeAbB/Dormitory-Management-System

እና ማከማናለን