

**UNIVERSITY OF GHANA**

(All rights reserved)

**DEPARTMENT OF COMPUTER ENGINEERING**

**SCHOOL OF ENGINEERING SCIENCES**

**COLLEGE OF BASIC & APPLIED SCIENCES**

**FIRST SEMESTER 2022/2023 ACADEMIC YEAR**

**COURSE CODE AND TITLE: CPEN 201- C++ PROGRAMMING**

**LECTURER: DR MARGARET RICHARDSON ANSAH**

**GROUP 2**

ABSTRACT

The objective of this project was to design and implement a basic elevator system using object-oriented programming principles in C++. The elevator system allows users to request floors, tracks the current floor, and manages the motion status of the elevator. The implementation is based on the Elevator class, which encapsulates the necessary functionality.

INTRODUCTION

Elevator systems play a vital role in multi-story buildings, enabling efficient vertical transportation. This lab focused on creating a simplified elevator system using object-oriented programming in C++. The system provides the ability to handle floor requests, track the current floor, and manage the motion status.

DESIGN

The elevator system is designed around the Elevator class and Building class, which serves as the core component. These class encapsulate the elevator's behaviour and properties.

IMPLEMENTATION

The Elevator class represents an individual elevator car in the building. It contains private member variables to track the elevator's current floor, direction, loading and unloading timers, and a Boolean array to represent the floors that are currently requested by passengers. The public member functions of the Elevator class include methods to move the elevator, update its state, and display information about its current state and destination requests.

The Building class represents the building that contains multiple elevator cars. It contains private member variables to track the number of elevator cars in the building, an array of pointers to each elevator car, and a Boolean two-dimensional array to represent the floor requests made by passengers in the building. The public member functions of the Building class include methods to handle floor requests, update elevator state, and display information about the state of the floor requests in the building.

The code also defines several constants that are used to control the behavior of the elevator system, including the time it takes for passengers to load and unload the elevator (LOAD\_TIME), the minimum spacing between elevator cars (SPACING), and the maximum length of a floor request queue (BUFF\_LENGTH).

CONCLUSION

In this lab, a basic elevator system was designed and implemented using object-oriented programming principles in C++. The Elevator class encapsulates the necessary functionality, including handling floor requests, tracking the current floor, and managing the motion status.

While this implementation provides a solid foundation for simulating elevator operations, it is important to note that it is a simplified version and lacks advanced features such as error handling, queue management, and multiple elevators. Enhancements can be made to add these features, improving the system's robustness and realism.

This lab demonstrates the practical application of object-oriented programming concepts in designing and implementing an elevator system. By utilizing encapsulation and modularity, the system becomes easily expandable and maintainable, laying the groundwork for further development and refinement.