**WANJIKU NDORIA –github link https://github.com/WanjikuLynn/Developer-Practical-Test.git**

**AN EFFICIENT PROGRAM FOR OPERATING 8 ELEVATORS AND ENSURES A PERSON HAS TO WAIT FOR MINIMUM TIME WHILE WAITING FOR THE LIFT TO ARRIVE**

Program Name**: Smartlift**

Programming Language: Java

Database Management System: MySQL

Algorithm: SCAN:

Data Structure: Queue

Software Development Methodology: Waterfall Methodology – This methodology is preferred because the program do not anticipate user reuirements change in future i.e, the 8-storeyed building will always have 8 floors.

STEP 1: Requirements Specification and Analysis Analysis

* User should be able to request the Elevator from anywhere in the building.
* User should be able to alight anywhere in the building.
* The elevator return to the ground floor and stay there if idle.
* If another user requests for the elevator from a different floor and wanting to go to a different/similar direction with the elevator, the elevator should be smart/efficient enough to prioritize.

STEP 2: SYSTEM DESIGN

This is the stage where the program/software’s conceptual model is developed.

In this case, I used Use cases Pseudo code and flow simulation to come up with the system design.

**Pseudo Code:**

Master

// declare arrays for the slaves (queues are implemented using arrays)

void setup(){

SPI.begin; // Begin SPI protocol

digitalWrite(SS,HIGH);

SPI.begin();

SPI.setClockDivider(SPI\_CLOCK\_DIV8);

}

void loop(){  
 // create as many functions as there are keypads in the building.  
 // inside each function

keypad1(){

// take keypad input.  
// check all the filters and assign a lift  
SPI.transfer(destination); // goes to selected slave  
enqueue(destination);

}

keypad2( ){

// take keypad input.  
// check all the filters and assign a lift  
SPI.transfer(destination); // goes to selected slave  
enqueue(destination); // added

}

keypad3( ){

// take keypad input.  
// check all the filters and assign a lift  
SPI.transfer(destination); // goes to selected slave  
enqueue(destination); // added

}

keypad4(){

// take keypad input.  
// check all the filters and assign a lift  
SPI.transfer(destination); // goes to selected slave  
enqueue(destination); // added

}

keypad5(){

// take keypad input.take out k  
// check all the filters and assign a lift  
SPI.transfer(destination); // goes to selected slave  
enqueue(destination); // added

}

Keypad6(){

//take keypad input.

//check all the filters and assign a lift

SPI.transfer(destination); // goes to selected slave  
enqueue(destination); // added

}

Keypad7(){

//take keypad input.

//check all the filters and assign a lift

SPI.transfer(destination); // goes to selected slave  
enqueue(destination); // added

}

Keypad8(){

//take keypad input.

//check all the filters and assign a lift

SPI.transfer(destination); // goes to selected slave  
enqueue (destination); // added

}

**Slave code:**  
  
// declare array for the slave (queue is implemented using array)  
void setup(){  
 pinMode(MISO,OUTPUT);  
 SPCR |= \_BV(SPE); // turn on slave mode  
 SPCR |= \_BV(SPIE); // turn on interrupt  
}  
ISR(SPI\_STC\_vect){  
 // receive destination from master  
 // add to the queue  
}  
void loop(){  
 // always looks at the first element of the queue and goes to that floor.  
 // when the destination is reached, it is deleted from queue  
 // inform master about deletion so that it can update the queue for this slave  
}?

USE CASE 1

**A person is on a particular floor. Lets say, Ground Floor. She wants to go to 5th Floor. she clicks on the elevator button with up direction.**

* Lets call this the ExternalRequest.This Request will be having the direction and the floor on which the button has been pressed by the user i.e. source floor. The elevator will check the available requests if any and then process this request depending on some priority.
* The elevator reaches the source floor i.e. the 0th or the ground floor. The person enters the elevator.
* The person then presses the 5th floor button in the elevator to indicate the elevator to go to 5th floor.
* This will be the internal request. It will be having only the floor to which the person wants to go to i.e. the destination floor. The elevator moves to the fifth floor. And the person then exits the elevator.

**USE CASE 2**

* When the elevator is moving from the ground floor to the fifth floor and it reaches the first floor, suppose another person on the second floor wants to go in the UP direction. The elevator will stop for this request and the person on second floor will enter the elevator. Suppose he presses the 4th floor button. Then the elevator will first stop at 4th floor which was the destination of the person which entered on the second floor. Later the elevator stops on the fifth floor and the person from the ground floor exits.

**USE CASE 3**

* If suppose when the elevator is moving from the ground floor to the fifth floor and it reaches the first floor. At this moment suppose another person on the second floor wants to go in the DOWN direction. Then the elevator will not stop for this request immediately. Elevator will first go to the fifth floor where the person from the ground floor will exit. Elevator will then go to the second floor. The person will enter the elevator and press 0. The elevator will then move to the zeroth floor.

It is also at this step that the Programming language and Database to be used is determined. In my case I used Java and MySQL

**STEP 3: CODING**

**Classes**

* Enum Direction — This enum will have two values UP and DOWN.
* Class ExternalRequest —The request made by the person from the floor when he request for the elevator The request made by the person from the floor when he requests for the elevator by pressing either the UP or the DOWN button. It will have the field’s enum Direction and integer sourceFloor.
* Class InternalRequest — The Request made by person when he enters the elevator. The person presses the floor number to which he wants to go. This will be the integer destinationFloor.
* Class Request — this class will be the encapsulation for the ExternalRequest and InternalRequest. Pass this Request to the elevator to be processed. The class will has two fields — ExternalRequest and InternalRequest**.**
* Enum State — this enum will have three values MOVING, STOPPED and IDLE.
* Lift — this class will represent the Elevator. It will have the field’s currentFloor, currentState and currentDirection.

ITERATION 1

com.javastructures;class Lift {  
 private int currentFloor = 0;  
 private Direction currentDirection = Direction.UP;  
 private State currentState = State.IDLE;  
}enum State { MOVING, STOPPED, IDLE}enum Direction { UP, DOWN}class Request implements Comparable<Request> {  
 private InternalRequest internalRequest;  
 private ExternalRequest externalRequest; public Request(InternalRequest internalRequest, ExternalRequest externalRequest) {  
 this.internalRequest = internalRequest;  
 this.externalRequest = externalRequest;  
 } public InternalRequest getInternalRequest() {  
 return internalRequest;  
 } public void setInternalRequest(InternalRequest internalRequest) {  
 this.internalRequest = internalRequest;  
 } public ExternalRequest getExternalRequest() {  
 return externalRequest;  
 } public void setExternalRequest(ExternalRequest externalRequest) {  
 this.externalRequest = externalRequest;  
 } @Override  
 public int compareTo(Request req) {  
 if (this.getInternalRequest().getDestinationFloor() == req.getInternalRequest().getDestinationFloor())  
 return 0;  
 else if (this.getInternalRequest().getDestinationFloor() > req.getInternalRequest().getDestinationFloor())  
 return 1;  
 else  
 return -1;  
 }}class ExternalRequest { private Direction directionToGo;  
 private int sourceFloor; public ExternalRequest(Direction directionToGo, int sourceFloor) {  
 this.directionToGo = directionToGo;  
 this.sourceFloor = sourceFloor;  
 } public Direction getDirectionToGo() {  
 return directionToGo;  
 } public void setDirectionToGo(Direction directionToGo) {  
 this.directionToGo = directionToGo;  
 } public int getSourceFloor() {  
 return sourceFloor;  
 } public void setSourceFloor(int sourceFloor) {  
 this.sourceFloor = sourceFloor;  
 }}class InternalRequest {  
 private int destinationFloor; public InternalRequest(int destinationFloor) {  
 this.destinationFloor = destinationFloor;  
 } public int getDestinationFloor() {  
 return destinationFloor;  
 } public void setDestinationFloor(int destinationFloor) {  
 this.destinationFloor = destinationFloor;  
 }}public class TestElevator { public static void main(String args[]) { Elevator elevator = new Elevator();  
  
 //person wants to go in up direction from source floor 0  
 ExternalRequest er = new ExternalRequest(Direction.UP, 0);  
  
 //the destination floor is 5  
 InternalRequest ir = new InternalRequest(5);  
 Request request1 = new Request(ir, er); }}

Database Design and Development

DBMS: MySQL

Entity Relationship Diagrams: