Project 2: Elevator System

Objective

Q. Design and code an Elevator. You can use any JDK, preferably JDK 8 and use collection framework, OOP principles to design a single thread elevator programme. This will test your knowledge of CORE Java.

Remember to use all the standard and good programming habits while programming.

Source Code

Direction.java

```
package com.java.assingment.elevator;

public enum Direction {
         UP, DOWN, IDLE
}
```

Elevator.java

```
package com.java.assingment.elevator;

import java.util.*;

//An elevator goes up , it continues to go up until there are no *dropoffs or *pickup requests
in that direction
public class Elevator {
         private static final int MIN_FLOOR = 0;
         private static final int MAX_FLOOR = 10;
         private static int processingTime = 500;// ms
         private int currentFloor;
         private Direction currentDirection;
//keeps track of people waiting K(starting floor) : V(List of All the destination floor from
that floor)
         private Map<Integer, List<Integer>> requestedPathsMap;

// Once the people at a given floor have boarded the elevator,
// *The currentFloorDestinations array -> (keeps track of the floors the elevator will visit by
setting the value at the appropriate index to true)
// Your job is to implement the processFloor(), callElevator() and moveElevator() functions

         private Boolean[] currentFloorDestinations;

         public Elevator() {
                  this.currentFloor = 0;// assumption the lift is starting from Ground
                  this.currentDirection = Direction.UP;// If at bottom , the lift will go up
                  this.requestedPathsMap = new HashMap<>();
                  this.currentFloorDestinations = new Boolean[MAX_FLOOR + 1];
                  Arrays.fill(this.currentFloorDestinations, Boolean.FALSE);
         }

         public void setProcessingTime(int processingTime) {
                  Elevator.processingTime = processingTime;
```

```
          }

          public int getCurrentFloor() {
                    return this.currentFloor;
          }

          public Map<Integer, List<Integer>> getRequestedPathsMap() {
                    return this.requestedPathsMap;
          }

          public Boolean[] getCurrentFloorDestinations() {
                    return this.currentFloorDestinations;
          }

          public void start() throws InterruptedException {
                    currentDirection = Direction.UP;// Assumption the lift is on ground floor
initially
                    do {
                              System.out.println("--------");
                              processFloor(currentFloor);
                              System.out.println("--------");
                    } while (currentDirection != Direction.IDLE);
                    System.out.println("No one is waiting and " + "no one is looking to go
anywhere");
                    System.out.println("Chilling for now");
          }

          public void lunchtimeElevatorRush() {
                    Random random = new Random();
                    for (int i = 0; i < 30; i++) {
                              callElevator(random.nextInt(11), random.nextInt(10) + 1);
                    }
          }

// TODO #1
          public void callElevator(int start, int destination) {
                    if (isInvalidFloor(start) || isInvalidFloor(destination) || start ==
destination) {
                              System.out.println("INVALID FLOORS. Try again");
                              return;
                    }
                    if (requestedPathsMap.containsKey(start))// if already START is in map, add
the destination in the list
                              requestedPathsMap.get(start).add(destination);
                    else {// else add the new key aas START with the list containing our
DESTINATION
                              requestedPathsMap.put(start, new ArrayList<Integer>() {
                                        {
                                                  add(destination);
                                        }
                              });
                    }
          }

// TODO #2
          private void processFloor(int floor) throws InterruptedException {
                    if (currentFloorDestinations[floor])
                              System.out.println("UN-BOARDING at Floor : " + floor);
                    if (requestedPathsMap.containsKey(floor)) {
          System.out.println("BOARDING at Floor : " + floor);
requestedPathsMap.get(floor).forEach(destinationFloor ->
currentFloorDestinations[destinationFloor] = true);
requestedPathsMap.remove(floor);// removing the entry from map as we have marked all the
destination
                    }
                    currentFloorDestinations[floor] = false;// Marked false as we are just
arrived in the current floor
                    moveElevator();
          }
```

```
//TODO #3
          private void moveElevator() throws InterruptedException {
//SETIING OF DIRECTION
//IDELING the elevator
                    if (!Arrays.asList(currentFloorDestinations).contains(true) &&
requestedPathsMap.isEmpty())
                              currentDirection = Direction.IDLE;// this will break the while loop
in our initial start() method
                              return;
                    } else if (isInvalidFloor(currentFloor + 1)) {// SWITCH TO DOWN direction
when reached top floor
                              currentDirection = Direction.DOWN;
                    } else if (isInvalidFloor(currentFloor - 1)) {// SWITCH TO UP direction when
reached bottom floor
                              currentDirection = Direction.UP;
                    }
                    switch (currentDirection) {// Move the elevator
// Enhanced switch available only in JDK14 onwards
// case UP-> moveUp();
                    case UP: {
                              moveUp();
                              break;
                    }
                    case DOWN: {
                              moveDown();
                              break;
                    }
                    default: {
                              System.out.println("Elevator Malfunctioned");
                    }
                    }
          }

          private void moveUp() throws InterruptedException {
                    currentFloor++;
                    System.out.println("GOING UP TO " + currentFloor);
                    Thread.sleep(processingTime);
          }

          private void moveDown() throws InterruptedException {
                    currentFloor--;
                    System.out.println("GOING DOWN TO " + currentFloor);
                    Thread.sleep(processingTime);
          }

          private boolean isInvalidFloor(int floor) {
                    return floor < MIN_FLOOR || floor > MAX_FLOOR;
          }
}
```

Main.java

```
package com.java.assingment.elevator;

import java.util.Scanner;

public class Main {

          static void automaticElevator() throws InterruptedException {
                    Elevator elevator = new Elevator();
                    elevator.lunchtimeElevatorRush();
                    elevator.start();
          }
          static void manualElevator() throws InterruptedException {
                    Elevator elevator = new Elevator();
                    Scanner sc = new Scanner(System.in);
                    System.out.println("Enter a starting floor 0 - 10");
                    int start = sc.nextInt();
                    System.out.println("Enter a destination floor 0 - 10");
                    int end = sc.nextInt();
                    elevator.callElevator(start, end);// calling the elevator to pick us up
                    elevator.start();
                    sc.close();
          }

          public static void main(String[] args) throws InterruptedException {
                    manualElevator();
//                  automaticElevator();
          }
}
```

Output

```
Enter a starting floor 0 - 10
2
Enter a destination floor 0 - 10
3
---------
GOING UP TO 1
---------
---------
GOING UP TO 2
---------
---------
BOARDING at Floor : 2
GOING UP TO 3
---------
---------
UN-BOARDING at Floor : 3
---------
No one is waiting and no one is looking to go anywhere
Chilling for now
```