**Classes and Methods Used for The Elevator Program:**

1. PeopleInElevator Class  
   a. enterElevator () method (Synchronized)

b. leaveElevator () method (Synchronized)

1. PeopleInFloor extending Thread Class
2. run () method
3. LeavingTheElevator extending Thread Class
4. run () method
5. ElevatorTest Main class
6. main () method

**Design Description of the Elevator/ Algorithm:**

1. Initially, we would want to have a data structure which can be treated as an elevator, in which we would insert values, when people would enter the lift and insert values when people leave the lift. We would use a Linked List.
2. We would make a Class PeopleInElevator which would have this linked list PeoplePresent, and a Boolean variable elevatorAvailable which would be set to false. In this class, we will have two methods enterElevator () and leaveElevator (). Both these methods should be synchronized so race condition won’t happen.
3. The constructor of this class will set the maxNoOfPpl which is the maximum number of people allowed in a floor, this is provided by user input.
4. The method enterElevator () would be a synchronized method, the first thing we do here is, we check for the value of the Boolean variable elevatorAvailable. We go in while loop until the value is false, inside the while loop we make this thread wait as the elevator might be busy while taking the people out of the elevator. When exiting of people from the elevator is done, elevatorAvailable becomes true, we can move further to let people in queue enter the elevator. In method enterElevator (), first we will find the number of people present in a floor (Randomly but between 0 and maxNoOfPpl), then we will enter that many number of weights (randomly between 40 and 100) in an array of same length. Now the queue of people to enter the elevator is ready, based on a sumWeight variable, which is the sum of people/weights inside the elevator/linked list, if the sumWeight is less than or equal to 300, the weights/people would be entered inside the elevator/linked list. If the sumWeight become more than that we exit the method dropping a message. The last thing we do in the method is, calling the notifyAll () method to notify all the threads which are waiting for the entering of people in elevator to finish. We also make the elevatorAvailable to false for the leaveElevator () method to allow people to leave the elevator.
5. The method leaveElevator () would be a synchronized method, the first thing we do here is, we check for the value of the Boolean variable elevatorAvailable. We go in while loop until the value is true, inside the while loop we make this thread wait as the elevator might be busy while allowing to enter the people inside the elevator. When entering of people in the elevator is done, elevatorAvailable becomes false, we can move further to the next floor. In method leaveElevator (), first we will find the number of people leaving the elevator (Randomly but between 0 and Number of people present in the elevator/linked list), then we will can remove that many people/weights from the linked list/elevator, meanwhile subtracting their weights from the weight the elevator is currently holding, i.e, sumWeight. The last thing we do in the method is, calling the notifyAll () method to notify all the threads which are waiting for the exiting of people out of the elevator to finish. We also make the elevatorAvailable to true for the enterElevator () method to allow people to enter the elevator.
6. Now we make two thread classes PeopleInFloor and LeavingTheElevator, which would be used to call the synchronized methods enterElevator () and leaveElevator () respectively, in for loops based on the number of floors. In each method we would make 2 for loops, one for the elevator to go up and one for the elevator to traverse down. When the threads for both the classes are started, it goes inside the run method which would have the 2 for loops mentioned above. Each method in the two different thread classes would be called in a synchronized way, since both the methods would be synchronized, and would use the wait and notify () methods accordingly as discussed above in step 4 and 5. These methods should be synchronized, otherwise the entering and exiting of people from the elevator would not be in synch.
7. Lastly, we make a ElevatorTest class, which would take user inputs for maximum number of people in a floor, Number of floors present and the number of times the elevator would traverse up and down (We can even make it infinite number of times). Then, we make an object of the PeopleInElevator class. We make the objects of the thread classes PeopleInFloor and LeavingTheElevator, passing the user inputs taken, and the PeopleInElevator class object to its constructor. We than start both the threads for both the elevator and the people. This class is the main class, which handles if an user input is valid or not, and prints appropriate messages accordingly.