**LIFT SIMULATION**

**Lift Simulation**  
  
 Lift Simulation implemented using ArrayList in java  
  
**Design :**  
  
 This application will capture the number of floors and number of elevators in the building. I am using ArrayList one for Events which stores list of events and another for list of requests coming from person. In this I implemented First Come First Serve based request processing. As of now this supports only two elevators. Adding events, removing processed events/requests from the ArrayList.  
  
**Implementation:** In this application Engine.java is the main class where the execution starts. Application will capture the Number of floors and number of elevators in the Building. As of now this application supports only two elevators. user should enter 1 or 2 as the value for the numberofElevators. if user enters other than these two values will throw InvalidArgumentException.  
 Elevator starts from the first floor in the beginning. First person starts it from 1st floor. So, when person comes will add personEvent into eventList. As this is the first time, no waiting request as of now. So, user gives the destination floor number.  
Here I implemented destination using random number. So, this request will be added into requestList.  
 Event.java is the class which maintains the event is of person or elevator status.  
 EventList.java contains List of events of Event type. It provides addEvents, get next event methods.  
 Floor.java class contains floorId and list of people present on the particular floor.  
 Person.java class provides the details of in which floor the user present and to which floor he wants to travel, i.e., destination.  
 Also the time he arrived to the elevator.  
 Request.java is the class maintains time when the request the elevator received and the floor the request it got from.  
 RequestList.java contains all the requests came to elevator.  
  
  
 Now Elevator starts execution based on eventList and requestList. It travels from source floor to destination floor based on input.

As of now it process the requests based on First Come First Serve and source floor to destination floor.

**JUnit test cases:**  
  
 I have written test cases for all classes and the coverage is 100% except for Engine.java class due to time constraint.

