File - C:\Users\mjhig\Desktop\cruise\src\Group.java

```java
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class Group {

    protected List<Person> people;


    // add person- adds a person to the group-return true, just for testing currently!!
    public void addPerson(Person p) {

        people.add(p);

    }

    // compute the average age of the dataset

    public double averageAge() {

        double sum = 0;

        for (Person p : people){sum += p.age;}

        return sum / people.size();
    }

    // initialize group of people
    public Group() {
        this.people = new LinkedList<>();

    }

    public int size() {

        return people.size();
    }
    public String display(){
        ArrayList<String> l = new ArrayList<>();
        for(Person p : people){

            l.add(p.display());

        }
        return "AVERAGE: "+averageAge()+" CURRENT MEMBERS: "+l.toString();
    }
}



/* attributes:
people - the collection of people in the group
services:
add person - adds a person to the group
average age - compute the average age of the people in the group */
```

File - C:\Users\mjhig\Desktop\cruise\src\Table.java

```java
import java.util.ArrayList;
import java.util.List;
public class Table {

    protected List<Person> occupants;

    protected int seatsAvailable;

    protected String tableName;


    // adds a person to the table, and updates occupants & seatsAvailable

    public void addPerson(Person p) {

        occupants.add(p);

        seatsAvailable -= 1;

    }

    //Compute a derived result, the average age of the table

    public boolean checkaverageAge(double currentAge) {

        int sum = 0;

        for (Person p : occupants) sum += p.age;

        int average = sum / occupants.size();

        if(average - currentAge <= 5 && average - currentAge >= -5){
            return true;
        }
        return false;
    }


    public Table(int seatsAvailable, String tableName) {

        this.tableName = tableName;

        this.seatsAvailable = seatsAvailable;

        occupants = new ArrayList<>();

    }
    public String display(){
        ArrayList<String> l = new ArrayList<>();

        for(Person p : occupants){

            l.add(p.display());

        }
        if(l.size()==0){
            return "TABLE NAME:"+tableName+ " CURRENT MEMBERS: N/A";
        }
        return "TABLE NAME: "+ tableName+" CURRENT MEMBERS: "+l.toString();
    }


```

File - C:\Users\mjhig\Desktop\cruise\src\Table.java

```
63
64
65
66
67      public boolean isEmpty(){
68
69          return occupants.isEmpty();
70
71      }
72
73
74
75      public boolean lookupPerson(Person lost){
76
77          for(Person p: occupants){
78
79              if(p.compareTo(lost)){
80
81                  return true;
82
83              }
84
85          }
86          return false;
87      }
88
89
90      public boolean spaceForGroup(int seatsAvailable){
91
92          return this.seatsAvailable >= seatsAvailable;
93
94      }
95
96  }
97  /*attributes:
98          [ not included: seats - the number of seats at this table -- it's a derived result]
99          people - people currently assigned to this table
100         seats available - the number of seats still available at this table
101         name - name (identifier) of this table
102         services:
103         add person - adds a person to this table (changes the average age and seats available)
104         average age - return the average age of the people currently seated at this table
105         seating chart - display the table's name then the people seated at the table
106         lookup person - isagiv en person sitting at this table
107         space for - given a need, check if there are at least that many seats available
108         is empty - true iff the table is empty*/
```

File - C:\Users\mjhig\Desktop\cruise\src\Person.java

```java
public class Person{

    protected double age;

    protected String name;


    public Person(String name, int age){

        this.name=name;

        this.age=age;

    }


    public String display(){

        return name+" : "+age;
    }


    public boolean compareTo(Person p){

        return name.equals(p.name);

    }

}

```

File - C:\Users\mjhig\Desktop\cruise\src\Tester.java

```
 1 import org.junit.Test;
 2 /*
 3         * This is Michaels code!  It's goal is to test the Dining Hall code
 4         * using jUnit.
 5         * CS 312 - Assignment 8
 6         * @author Michael Higgins
 7         * @version 1  11/22/2020
 8         */
 9
10         import static org.junit.Assert.*;
11 // for main
12         import org.junit.runner.JUnitCore;
13         import org.junit.runner.Result;
14         import org.junit.runner.notification.Failure;
15
16 // for listener
17         import org.junit.runner.Description;
18         import org.junit.runner.notification.RunListener;
19
20 public class Tester
21 {
22     public Tester() {}
23
24     @Test
25     public void personTest() // might go on to test a range of powers
26     {
27         Person p = new Person("Harry Warden",31);
28         assertEquals(p.display(),
29                 "Harry Warden : 31.0");
30     }
31
32     @Test
33     public void addGroupTest()
34     {
35         Person p1 = new Person("Sarah Mercer",21);
36         Person p2 = new Person("Axel Palmer", 22);
37         Group g1 = new Group();
38         g1.addPerson(p1);
39         g1.addPerson(p2);
40         assertEquals(g1.display(), "AVERAGE: 21.5 CURRENT MEMBERS: [Sarah Mercer : 21.0, Axel Palmer : 22.0]");
41     }
42     @Test
43     public void addPersonTable(){
44         Person p = new Person("Chief Jake Newby",62 );
45         Table t = new Table(5, "Table one");
46         t.addPerson(p);
47         assertEquals(t.display(),"TABLE NAME: Table one CURRENT MEMBERS: [Chief Jake Newby : 62.0]");
48     }
49     @Test
50     public void checkifTableEmpty(){
51         Table t = new Table(5, "Table TWO");
52         assertTrue(t.isEmpty());
53     }
54     @Test
55     public void AddTableToDininghall(){
56         DiningHall d = new DiningHall();
57         Table t = new Table(8, "Table THREE!");
58         d.addTable(t);
59         assertEquals(d.seatingChart(),"SEATING CHART: [TABLE NAME:Table THREE! CURRENT MEMBERS: N/A]");
60     }
61     @Test
62     public void lost(){
```

File - C:\Users\mjhig\Desktop\cruise\src\Tester.java

```
63
64          DiningHall d = new DiningHall();
65          Person p = new Person("Axel Palmer",26);
66
67          Table t1 = new Table(5,"Table One");
68          t1.addPerson(new Person("Tom Jesse",22));
69          t1.addPerson(new Person("Sarah Mercer",20));
70          t1.addPerson(new Person("Axel Palmer",26));
71
72          Table t2= new Table(5,"Table One");
73          t2.addPerson(new Person("Patty",52));
74          t2.addPerson(new Person("Chief Jake Newby",20));
75          t2.addPerson(new Person("Hollis",29));
76
77          Table t3 = new Table(5,"Table One");
78          t3.addPerson(new Person("Howard Landers",22));
79          t3.addPerson(new Person("Harriet",23));
80          t3.addPerson(new Person("Mike Stavinski",19));
81
82          d.addTable(t1);
83          d.addTable(t2);
84          d.addTable(t3);
85
86          assertEquals(d.findTable(p),t1);
87
88      }
89      @Test
90      public void placeGroup(){
91
92          DiningHall d = new DiningHall();
93
94          Table t1= new Table(8,"Table One");
95
96          t1.addPerson(new Person("Patty",52));
97
98          t1.addPerson(new Person("Chief Jake Newby",20));
99
100         t1.addPerson(new Person("Hollis",29));
101
102         Table t2 = new Table(8,"Table Two");
103
104         t2.addPerson(new Person("Howard Landers",22));
105
106         t2.addPerson(new Person("Harriet",23));
107
108         t2.addPerson(new Person("Mike Stavinski",19));
109
110         Group g1 = new Group();
111
112         g1.addPerson(new Person("Tom Jesse",22));
113
114         g1.addPerson(new Person("Sarah Mercer",20));
115
116         g1.addPerson(new Person("Axel Palmer",26));
117
118         d.addTable(t1);
119
120         d.addTable(t2);
121
122         d.placeGroup(g1);
123
124         assertEquals(t2.display(), "TABLE NAME: Table Two CURRENT MEMBERS: [Howard Landers : 22.0, Harriet : 23.0, Mike Stavinski : 19.0, Tom Jesse : 22.0, Sarah Mercer : 20.0, Axel Palmer
```

File - C:\Users\mjhig\Desktop\cruise\src\Tester.java

```
124  : 26.0]");
125
126      }
127      @Test
128      public void checkAgeMatch(){
129
130          DiningHall d = new DiningHall();
131
132          Table t1= new Table(8,"Table One");
133
134          t1.addPerson(new Person("Patty",52));
135
136          t1.addPerson(new Person("Chief Jake Newby",20));
137
138          t1.addPerson(new Person("Hollis",29));
139
140          Table t2 = new Table(8,"Table Two");
141
142          t2.addPerson(new Person("Howard Landers",22));
143
144          t2.addPerson(new Person("Harriet",23));
145
146          t2.addPerson(new Person("Mike Stavinski",19));
147
148          Group g1 = new Group();
149
150          g1.addPerson(new Person("Tom Jesse",22));
151
152          g1.addPerson(new Person("Sarah Mercer",20));
153
154          g1.addPerson(new Person("Axel Palmer",26));
155
156          d.addTable(t1);
157
158          d.addTable(t2);
159
160          d.ageMatch(22.6,5);
161
162          assertEquals( d.ageMatch(22.6,5),t2);
163
164      }
165
166      @Test
167      public void findTableEmpty(){
168
169          DiningHall d = new DiningHall();
170
171          Table t1= new Table(8,"Table One");
172
173          t1.addPerson(new Person("Patty",52));
174
175          t1.addPerson(new Person("Chief Jake Newby",20));
176
177          t1.addPerson(new Person("Hollis",29));
178
179          Table t2 = new Table(8,"Table Two");
180
181          t2.addPerson(new Person("Howard Landers",22));
182
183          t2.addPerson(new Person("Harriet",23));
184
185          t2.addPerson(new Person("Mike Stavinski",19));
```

File - C:\Users\mjhig\Desktop\cruise\src\Tester.java

```
186         Table t3 = new Table(4,"Table Three");
187 
188         d.addTable(t1);
189 
190         d.addTable(t2);
191 
192         d.addTable(t3);
193 
194         assertEquals(d.findTableEmpty().tableName,"Table Three");
195     }
196     @Test
197     public void findEnoughSpace(){
198 
199     DiningHall d = new DiningHall();
200 
201     Table t1= new Table(8,"Table One");
202 
203     t1.addPerson(new Person("Patty",52));
204 
205     t1.addPerson(new Person("Chief Jake Newby",20));
206 
207     t1.addPerson(new Person("Hollis",29));
208 
209     Table t2 = new Table(8,"Table Two");
210 
211     t2.addPerson(new Person("Howard Landers",22));
212 
213     t2.addPerson(new Person("Harriet",23));
214 
215     t2.addPerson(new Person("Mike Stavinski",19));
216     Table t3 = new Table(4,"Table Three");
217 
218     d.addTable(t1);
219 
220     d.addTable(t2);
221 
222     d.addTable(t3);
223 
224     assertEquals(d.findTableWithFreeSeats(3).tableName,"Table One");
225 }
226 @Test
227     public void displayChart(){
228 
229     DiningHall d = new DiningHall();
230 
231     Table t1= new Table(8,"Table One");
232 
233     t1.addPerson(new Person("Patty",52));
234 
235     t1.addPerson(new Person("Chief Jake Newby",60));
236 
237     t1.addPerson(new Person("Hollis",29));
238 
239     Table t2 = new Table(8,"Table Two");
240 
241     t2.addPerson(new Person("Howard Landers",22));
242 
243     t2.addPerson(new Person("Harriet",23));
244 
245     t2.addPerson(new Person("Mike Stavinski",19));
246 
247     Table t3 = new Table(4,"Table Three");
```

File - C:\Users\mjhig\Desktop\cruise\src\Tester.java

```
248     t3.addPerson(new Person("Tom Jesse",22));
249
250     t3.addPerson(new Person("Sarah Mercer",20));
251
252     t3.addPerson(new Person("Axel Palmer",26));
253
254     d.addTable(t1);
255
256     d.addTable(t2);
257
258     d.addTable(t3);
259
260     assertEquals(d.seatingChart(),"SEATING CHART: [TABLE NAME: Table One CURRENT MEMBERS: [Patty : 52.0, Chief Jake Newby : 60.0, Hollis : 29.0], TABLE NAME: Table Two CURRENT MEMBERS: [
    Howard Landers : 22.0, Harriet : 23.0, Mike Stavinski : 19.0], TABLE NAME: Table Three CURRENT MEMBERS: [Tom Jesse : 22.0, Sarah Mercer : 20.0, Axel Palmer : 26.0]]");
261 }
262     public static void main(String[] args)
263     {
264         JUnitCore runner = new JUnitCore();
265
266         runner.addListener(new TestListener());
267
268         Result result = runner.run(Tester.class);
269
270     }
271 }
272
273
274 class TestListener extends RunListener
275 {
276     public void testStarted(Description description) { }
277
278     public void testFinished(Description description)
279     {
280         // System.out.println("Finished "+ description.getMethodName());
281     }
282
283     public void testRunFinished(Result result)
284     {
285         int ran = result.getRunCount();
286
287         int failed = result.getFailureCount();
288
289
290         System.out.println("Ran " + ran + " SUCCESSES: " + (ran-failed) + (result.wasSuccessful() ? " passed ":" "));
291
292         for (Failure failure : result.getFailures())
293         {
294             System.out.println(failure.toString());
295         }
296
297     }
298 }
299
```

File - C:\Users\mjhig\Desktop\cruise\src\DiningHall.java

```java
import java.util.ArrayList;
import java.util.LinkedList;

public class DiningHall {

    LinkedList<Table> tables;


    public void addTable(Table t){

        tables.add(t);

    }



    public DiningHall(){

    tables =new LinkedList<>();
    }

    public Table findTable(Person p) {

        for (Table t : tables) {

            if (t.lookupPerson(p))

                return t;

        }
        return null;
    }


    public void placeGroup(Group g) {

        Table t = ageMatch(g.averageAge(), g.size());

        if (t == null){

            t = findTableEmpty();
        }

        if (t == null){

            t = findTableWithFreeSeats(g.size());}

        if (t == null){

            return;

        }

        for( Person p : g.people){

            t.addPerson(p);
        }

    }
    public String seatingChart(){

        ArrayList<String> l = new ArrayList<>();
```

File - C:\Users\mjhig\Desktop\cruise\src\DiningHall.java

```
63
64          for(Table t : tables){
65
66              l.add(t.display());
67
68          }
69          return "SEATING CHART: " + l.toString();
70      }
71
72      public Table ageMatch(double targetAge, int seatsNeeded){
73
74          for(Table current : tables){
75
76              if(current.checkaverageAge(targetAge) && current.spaceForGroup(seatsNeeded)) return current;
77
78          }
79          return null;
80
81      }
82
83      /*spaceFor(seatsNeeded) and
84
85      t.averageAge=targetAge ± 5
86      zeroOccupants -(private)*/
87
88      public Table findTableEmpty(){
89
90          for (Table t :tables) {
91              if (t.isEmpty()) {
92                  return t;
93              }
94          }
95          return null;
96      }
97
98      public Table findTableWithFreeSeats(int seatsNeeded) {
99
100             return tables.stream().filter(t -> t.spaceForGroup(seatsNeeded)).findFirst().orElse(null);
101     }
102 }
```