Random testers:

All tests were based off the initial provided code and built upon from there. For the first file, "Appt" both methods that needed to be covered could be accomplished through a single pass of randomization for each of the values that go into the "Appt" constructor. There was an odd dilemma surfacing where the "setDescription" and "setStartMinute" calls did not receive full coverage but was immediately remedied by requesting them to perform their operation twice for every call. I assume this was somehow the result of very rare gaps in the random testing. The next file for coverage was "CalDay" and its method simply incorporated what "Appt" did but then adds each new set into the calendar. Slight alterations had to be made to ensure that a full appointment was submitted by removing the if-statements to each random generation resulting in a complete set being submitted. Lastly, "TimeTable" with its "deleteAppt" method required a slightly more involved resolution. After adding what "CalDay" test coverage did there was also the need to add an if-statement that will randomly test for variations of null being introduced into the linked list.

All methods mentioned in the assignment have made full coverage.

Images of coverage:

```
74                public void addAppt(Appt appt) {
75  270419949         if (appt.getValid()) {
76   45378978             for (int i = 0; i < getAppts().size(); i++) {
77                            //Put the appointment in the correct order - finish this
78   32832904                 if (((Appt)getAppts().get(i)).getStartHour() >
79   32832904                                                         appt.getStartHour()) {
80
81    5358560                     getAppts().add(i, appt);
82    5358560                     return;
83                            }
84                        }
85                        //The appointment hasn't been added yet, so add it
86   12546074             getAppts().add(appt);
87                    }
88  265061389         }
89
```

```
125                   public LinkedList<Appt> deleteAppt(LinkedList<Appt> appts,Appt appt) {
126                       //Do not do anything to appts equals to null
127  219641900           if(appts==null||appt==null)
128   48820100               return null;
129                       //Do not do anything to invalid appointments
130  170821800           if (!appt.getValid()) {
131   81252702               return null;
132                       }
133
134                       //Remove the appointment from the list appts if applicable
135
136   98823498           for(int i=0;i<appts.size();i++){
137   14760698               Appt tempAppt=appts.get(i);
138   14760698               if(tempAppt.equals(appt)){
139    5506298                   appts.remove(i);
140    5506298                   return appts;
141                           }
142
143                       }
144   84062800           return null;
145                   }
146               }
```

```
 79              private void isValid() {
 80 2725902269          if(startHour<0 || startHour>23)
 81  963689802                  this.valid=false;
 82                       else
 83 1762212467              if(startMinute<0 || startMinute>59)
 84  864380920                      this.valid=false;
 85                           else
 86  897831547                  if(startDay<1 || startDay>31)
 87  195462292                          this.valid=false;
 88                               else
 89  702369255                      if(startMonth<1 || startMonth>12)
 90   84414832                              this.valid=false;
 91                                   else
 92  617954423                              this.valid=true;
 93 2725902269      }
 94

135              /** Sets description */
136              public void setDescription(String description) {
137  874760430          if (description == null)
138   12755866              this.description = "";
139                   else
140  862004564              this.description = description;
141  874760430      }
142
```