Random Testers

TimeTable

TimeTable was the most difficult to write as it utilized features from the previous two classes. I created a for loop to create a random amount of appointments between 0 and 100. These appointments were created with randomly generated details and added to a linked list. Passed into the deleteAppt method was one of three randomly selected possibilities: An appointment in the linked list, an appointment not in the list, or null. This provided better coverage and was required to fully cover the branches in the method. All branches were covered.

```
125
126
127  7225927    public LinkedList<Appt> deleteAppt(LinkedList<Appt> appts,Appt appt) {
128  1665633            //Do not do anything to appts equals to null
129                 if(appts==null||appt==null)
130  5560294                return null;
131  5508511            //Do not do anything to invalid appointments
132                 if (!appt.getValid()) {
133                     return null;
134                 }
135
136   235342        //Remove the appointment from the list appts if applicable
137   226047
138   226047        for(int i=0;i<appts.size();i++){
139    42488            Appt tempAppt=appts.get(i);
140    42488            if(tempAppt.equals(appt)){
141                         appts.remove(i);
142                         return appts;
143                     }
144     9295
145
                    }
                    return null;
                }
```

CalDay

CalDay was tested by generating random appointments for the time limit and calling the addAppt method. This was a fairly easy method to create a random test for and I was able to achieve complete branch coverage.

```
74
75    759109    public void addAppt(Appt appt) {
76   7057962        if (appt.getValid()) {
77                     for (int i = 0; i < getAppts().size(); i++) {
78   7056683               //Put the appointment in the correct order - finish this
79   7056683               if (((Appt)getAppts().get(i)).getStartHour() >
80                                                                   appt.getStartHour()) {
81      6912
82      6912                       getAppts().add(i, appt);
83                                 return;
84                             }
85                         }
86      1279               //The appointment hasn't been added yet, so add it
87                         getAppts().add(appt);
88    752197           }
                }
```

Appt

The two methods in Appt were the easiest to create a random test for, and I achieved full branch coverage. For full coverage of setDescription, the loop generated a random string with the possibility of it being null. For isValid, a for loop ran for the length of the time limit generating random values ranging from lower than the conditional checks for to higher than it checks for.

```
135                 /** Sets description */
136                 public void setDescription(String description) {
137 179956651           if (description == null)
138   9100221               this.description = "";
139                     else
140 170856430               this.description = description;
141 179956651         }
142
```

```
 79
 80  12201866   private void isValid() {
 81   5475651           if(startHour<0 || startHour>23)
 82                             this.valid=false;
 83   6726215           else
 84   3098671               if(startMinute<0 || startMinute>59)
 85                                 this.valid=false;
 86   3627544               else
 87   2511638                   if(startDay<1 || startDay>31)
 88                                     this.valid=false;
 89   1115906                   else
 90    972594                       if(startMonth<1 || startMonth>12)
 91                                         this.valid=false;
 92    143312                       else
 93  12201866                             this.valid=true;
                     }
```