```
1  package caltool.schedule;
2
3  import caltool.caldb.*;
4  import mvp.*;
5  import java.util.*;
6
7  /****
8   *
9   * Class ScheduleTest is the companion testing class for class <a href=
10  * Schedule.html> Schedule </a>.  It implements the following module test plan:
11  *                                                                      <pre>
12  *     Phase 1: Unit test the constructor.
13  *
14  *     Phase 2: Unit test the simple access method getCategories.
15  *
16  *     Phase 3: Unit test the constructive methods scheduleAppointment,
17  *              scheduleTask, and scheduleEvent.
18  *
19  *     Phase 4: Unit test the constructive methods scheduleMeeting and
20  *              confirmMeeting.
21  *
22  *     Phase 5: Unit test the changeItem and deleteItem methods.
23  *
24  *     Phase 6: Repeat phases 1 through 5.
25  *
26  *     Phase 7: Stress test by scheduling and deleting 100000 items.
27  *                                                                    </pre>
28  */
29 public class ScheduleTest extends Schedule {
30
31     /*-*
32      * Public methods
33      */
34
35     /**
36      * Construct by calling the parent constructor with a null View and the
37      * given stubbed CalendarDB.
38      */
39     public ScheduleTest(CalendarDB calDB) {
40         super(null, calDB);
41     }
42
43     /**
44      * Run all the phases of this.
45      */
46     public void run() {
47         phase1();
48         phase2();
49         phase3();
50         phase4();
51         phase5();
52         phase6();
53         phase7();
54     }
55
56
57     /*-*
58      * PhaseX methods are for each module testing phase.
59      */
60
61     /**
62      * Execute test phase 1 by calling testSchedule.
63      */
64     protected void phase1() {
65         dumpPhaseHeader(1);
66         testSchedule();
67         dumpPhaseEndSpacing();
68     }
69
70     /**
71      * Execute test phase 2 by calling testGetCategories.
72      */
73     protected void phase2() {
74         dumpPhaseHeader(2);
75         testGetCategories();
76         dumpPhaseEndSpacing();
77     }
78
79     /**
80      * Execute test phase 3 by calling testScheduleAppointment,
81      * testScheduleTask, and testScheduleEvent on the schedule built in phase
82      * 1.  These three unit tests in turn exercise the testAlreadyScheduled
83      * and testValidateInputs methods.
84      */
85     protected void phase3() {
86         dumpPhaseHeader(3);
87         testScheduleAppointment();
88         testScheduleTask();
89         testScheduleEvent();
90         dumpPhaseEndSpacing();
91     }
92
93     /**
94      * Execute test phase 4 by ...
95      */
96     protected void phase4() {
97         dumpPhaseHeader(4);
98         /* ... */
99         dumpPhaseEndSpacing();
100    }
101
102    /**
103     * Execute test phase 5 by ...
104     */
105    protected void phase5() {
106        dumpPhaseHeader(5);
107        /* ... */
108        dumpPhaseEndSpacing();
109    }
110
111    /**
112     * Execute test phase 6 by ...
```

```
113      */
114     protected void phase6() {
115         dumpPhaseHeader(6);
116         /* ... */
117         dumpPhaseEndSpacing();
118     }
119
120
121     /**
122      * Execute test phase 7 by ...
123      */
124     protected void phase7() {
125         dumpPhaseHeader(7);
126         /* ... */
127         dumpPhaseEndSpacing();
128     }
129
130     /*-*
131      * Individual unit testing methods for member methods
132      */
133
134     /**
135      * Unit test the constructor by building one Schedule object.  No further
136      * constructor testing is necessary since only one Schedule object is every
137      * constructed in the CalendarTool system.
138      */
139     protected void testSchedule() {
140         dumpUnitTestHeader("Constructor");
141         schedule = new Schedule(null, calDB);
142         dumpUnitTestEndSpacing();
143     }
144
145     /**
146      * Unit test getCategories by calling getCategories on a schedule with a
147      * null and non-null categories field.  The Categories model is tested
148      * fully in its own class test.
149      *                                                                    <pre>
150      *  Test
151      *  Case    Input                   Output          Remarks
152      * ========================================================================
153      *   1      schedule.categories     null            Null case
154      *            = null
155      *
156      *   2      schedule.categories     same non-null   Non-null case
157      *            = non-null value      value
158      *                                                                   </pre>
159      */
160     protected void testGetCategories() {
161
162         /*
163          * Dump a unit test header.
164          */
165         dumpUnitTestHeader("getCategories");
166
167         /*
168          * Do case 1 and dump the results.
169          */
170         schedule.categories = null;
171         dump("Case 1:");
172         dump("Should be null: " + schedule.getCategories());
173
174         /*
175          * Do case 2 and dump the results.
176          */
177         schedule.categories = new Categories();
178         dump("Case 2:");
179         dump("Should be non-null: " + schedule.getCategories());
180     }
181
182     /**
183      * Unit test scheduleAppointment by ...
184      */
185     protected void testScheduleAppointment() {
186         /* ... */
187     }
188
189     /**
190      * Unit test scheduleAppointment by ...
191      */
192     protected void testScheduleTask() {
193         /* ... */
194     }
195
196     /**
197      * Unit test scheduleEvent by supplying inputs that test the value range of
198      * each Event data field and exercise each precondition clause.  The cases
199      * are as follows:
200      *                                                                    <pre>
201      *  Test
202      *  Case    Input                   Output          Remarks
203      * ========================================================================
204      *   1      {"Event 0",             Input event     Lower end of possible
205      *           {"Sunday",1,           added to items  event data range
206      *            "January",1},         no junk, no
207      *            null,                 confusion
208      *            null,
209      *            null}
210      *
211      *   2      {10000-char string,     Input event     Upper end of possible
212      *           {"Sunday",1,           added to items  event data range
213      *            "January",1},         no junk,
214      *           {"Saturday",31,        no confusion
215      *            "December", 9999},
216      *           10000-char string
217      *             reversed,
218      *           10000-char string}
219      *
220      *   3      events with start and   Input event     Events with all combos
221      *  thru    end date ranging thru   added to items  of days, dates, and
222      *  2564    each legal day, each    no junk, no     months
223      *          legal date, and each    confusion
224      *          legal month, with
```

```
225       *          arbitrary year,                                                  
226       *          category, and security                                           
227       *                                                                           
228       *  2565    32 cases to exercise    Schedule        Events that violate      
229       *  thru    possible boolean        Event           precond logic.           
230       *  2597    values for 5 precond    Precond                                  
231       *          clauses                 Exception                                
232       *                                   thrown                                  
233       *                                                                  </pre>   
234       */                                                                         
235      protected void testScheduleEvent() {                                        
236                                                                                  
237          /*                                                                      
238           * Dump a unit test header.                                             
239           */                                                                     
240          dumpUnitTestHeader("scheduleEvent");                                    
241                                                                                  
242          /*                                                                      
243           * Do the cases.                                                        
244           */                                                                     
245          try {                                                                   
246              testScheduleEventCase1();                                           
247              testScheduleEventCase2();                                           
248              testScheduleEventCases3Thru2564();                                  
249              testScheduleEventCases2565Thru2597();                               
250          }                                                                       
251          catch (ScheduleEventPrecondViolation e) {                               
252              /* This will never happen */                                        
253          }                                                                       
254                                                                                  
255          /*                                                                      
256           * Dump the test case results.                                          
257           */                                                                     
258          dump("");                                                               
259          dumpUnitTestEndSpacing();                                               
260                                                                                  
261      }                                                                           
262                                                                                  
263      /**                                                                         
264       * Run testScheduleEvent case 1.                                            
265       */                                                                         
266      protected void testScheduleEventCase1()                                     
267              throws ScheduleEventPrecondViolation {                              
268                                                                                  
269          dumpUnitTestCaseHeader("scheduleEvent", 1);                             
270          schedule.scheduleEvent(new Event(                                       
271              "Event 0",                          // title                        
272              new Date(new DayName("Sunday"),     // start date                   
273                  1,                                                              
274                  new MonthName("January"),                                       
275                  1),                                                             
276              new Date(new DayName("Tuesday"),    // start date                   
277                  2,                                                              
278                  new MonthName("January"),                                       
279                  1),                                                             
280              new Category("Category 1"),         // category                     
281              new SimpleSecurity("public")                // security             
282          ));                                                                     
283      }                                                                           
284                                                                                  
285      /**                                                                         
286       * Run testScheduleEvent case 2.                                            
287       */                                                                         
288      protected void testScheduleEventCase2()                                     
289              throws ScheduleEventPrecondViolation {                              
290                                                                                  
291          int i;                                                                  
292          String longString = "";                                                 
293          String longStringReversed = "";                                         
294          for (i = 0; i < 10000; i++) {                                           
295              longString += (char) i % 256;                                       
296              longStringReversed = (char) i + longStringReversed;                 
297          }                                                                       
298                                                                                  
299          dumpUnitTestCaseHeader("scheduleEvent", 2);                             
300                                                                                  
301          schedule.scheduleEvent(new Event(                                       
302              longString,                         // title                        
303              new Date(new DayName("Sunday"),     // start date                   
304                  1,                                                              
305                  new MonthName("January"),                                       
306                  1),                                                             
307              new Date(new DayName("Tuesday"),    // start date                   
308                  2,                                                              
309                  new MonthName("January"),                                       
310                  1),                                                             
311              new Category(longStringReversed),   // category                     
312              new SimpleSecurity("private")       // security                     
313          ));                                                                     
314      }                                                                           
315                                                                                  
316      /**                                                                         
317       * Run testScheduleEvent cases 3 through 2564 by looping through various    
318       * combinations of days, weeks, and months.                                 
319       */                                                                         
320      protected void testScheduleEventCases3Thru2564()                            
321              throws ScheduleEventPrecondViolation {                              
322                                                                                  
323          int day, number, month, year, i, n;                                     
324          Date date;                                                              
325                                                                                  
326          for (i = 3, day = 0; day < 7; day++) {                                  
327              for (number = 1; number <= 31; number++) {                          
328                  for (month = 0; month < 12; month++) {                          
329                                                                                  
330                      date = new Date(                                            
331                          new DayName(day),                                       
332                          number,                                                 
333                          new MonthName(month),                                   
334                          year = number * 100);                                   
335                                                                                  
336                      if (date.isValid()) {                                       
```

```
337                         dumpUnitTestCaseHeader("scheduleEvent", i++);
338
339                         schedule.scheduleEvent(new Event(
340                             "Event " +                                  // title
341                                 String.valueOf(n = year + month + day),
342                             date,                                       // start
343                             date,                                       // end
344                             new Category(String.valueOf(n*10)),         // category
345                             ((number % 2) == 0) ?                       // security
346                                 new SimpleSecurity("public") :
347                                 new SimpleSecurity("private")
348                         ));
349                     }
350                 }
351             }
352         }
353
354     }
355
356     /**
357      * Run testScheduleEvent cases 2565 through 2597 to fully exercise the
358      * precond logic.
359      */
360     protected void testScheduleEventCases2565Thru2597() {
361         /* ... */
362     }
363
364
365     /*-*
366      * Testing utility methods.
367      */
368
369     /**
370      * Output a header message to stdout identifying the test phase.
371      */
372     protected void dumpPhaseHeader(int phasenum) {
373         System.out.print("**** Schedule Testing Phase ");
374         System.out.print(phasenum);
375         System.out.println(" ****");
376     }
377
378     /**
379      * Output a header message to stdout for the unit test of the given
380      * testname.
381      */
382     protected void dumpUnitTestHeader(String testname) {
383         System.out.print("** Unit Test " + testname + " **\n");
384     }
385
386     /**
387      * Output a header message to stdout for a unit test case.
388      */
389     protected void dumpUnitTestCaseHeader(String testname, int caseNumber) {
390         System.out.print("* Unit Test Case " +
391             testname + String.valueOf(caseNumber)  + " **\n");
392     }
393
394
395     /**
396      * Dump three blank lines following a test phase.
397      */
398     protected void dumpPhaseEndSpacing() {
399         System.out.print("\n\n\n");
400     }
401
402     /**
403      * Dump a couple blank lines following a unit test.
404      */
405     protected void dumpUnitTestEndSpacing() {
406         System.out.print("\n\n");
407     }
408
409     /**
410      * Dump the data values of schedule to stdout.  For test validation
411      * purposes, include a print of the number of elements in the dumped items.
412      * Precede the dump with the given message, if the message is non-null.
413      */
414     protected void dump(String message) {
415         int l;                  // Temp
416
417         if (message != null) {
418             System.out.print("* " + message + " *" + "\n");
419         }
420
421         System.out.print("Schedule contains\n" +
422             "Categories: " + schedule.categories + "\n" +
423             (l = schedule.calDB.getCurrentCalendar().numItems()) + " " +
424             (l == 1 ? " item" : " items") + "\n" +
425             schedule.toString() + "\n"
426         );
427     }
428
429     /**
430      * Schedule data object to support the tests.
431      */
432     protected Schedule schedule;
433
434 }
```