

```
1: #include <iostream>
2: #include <string>
3: using namespace std;
4:
5: class Television {
6:
7:     /* class-level attributes */
8:     static const int MIN_VOLUME = 0;
9:     static const int MAX_VOLUME = 10;
10:    static const int MIN_CHANNEL = 2;
11:    static const int MAX_CHANNEL = 99;
12:
13: private:
14:     // Data members of instance
15:
16:     /** Whether the power is on */
17:     bool    powerOn;
18:
19:     /** Whether the tv is muted */
20:     bool    muted;
21:
22:     /** The current volume level */
23:     int     volume;
24:
25:     /** The most recent previous channel number */
26:     int     prevChan;
27:
28: public:
29:
30:     /** Creates a new Television instance.
31:     *
32:     * The power is initially off. Upon the first time the TV is turned on,
33:     * it will be set to channel 2, and a volume level of 5.
34:     */
35:     Television() : powerOn(false), muted(false), volume(5),
36:                   channel(2), prevChan(2) { }
37:
38:     /** Toggles the power setting.
39:     *
40:     * If Television is off, turns it on.
41:     * If Television is on, turns it off.
42:     */
43:     togglePower() { powerOn = !powerOn; }
44:
45:     /** Toggles the setting for mute.
46:     *
47:     * If power is off, there is no effect.
48:     *
49:     * Otherwise, if television was unmuted, it becomes muted.
50:     * If television was muted, it becomes unmuted and the volume is
51:     * restored to its previous setting.
52:     */
53:     void toggleMute() {
54:         if (powerOn)
55:             muted = !muted;
56:     }
57:
58:     /** Increments the volume of the Television by one increment.
59:     *
60:     * If power is currently off, there is no effect (-1 returned).
61:     * Otherwise, updates the volume setting appropriately.
62:     *
63:     * If volume was at maximum level, it remains at maximum level.
64:     * If television is currently muted, it will be unmuted as a result.
65:     *
```

```
66:     * @return the resulting volume level
67:     */
68:     int volumeUp() {
69:         if (powerOn) {
70:             if (volume < MAX_VOLUME)
71:                 volume++;
72:             muted = false;
73:             return volume;
74:         } else
75:             return -1;
76:     }
77:
78:     /** Decrements the volume of the Television by one increment.
79:     *
80:     * If power is currently off, there is no effect (-1 returned).
81:     * Otherwise, updates the volume setting appropriately.
82:     *
83:     * If volume was at minimum level, it remains at minimum level.
84:     * If television is currently muted, it will be unmuted as a result.
85:     *
86:     * @return the resulting volume level
87:     */
88:     int volumeDown() {
89:         if (powerOn) {
90:             if (volume > MIN_VOLUME)
91:                 volume--;
92:             muted = false;
93:             return volume;
94:         } else
95:             return -1;
96:     }
97:
98:     /** Increments the channel.
99:     *
100:    * If power is off, there is no effect (-1 returned).
101:    * Otherwise, updates the channel setting appropriately.
102:    *
103:    * If channel had been set to the maximum of the valid range of
104:    * channels, the effect will be to 'wrap' around resulting in the
105:    * channel being set to the minimum channel.
106:    *
107:    * @return The resulting channel setting
108:    */
109:    int channelUp() {
110:        if (powerOn) {
111:            prevChan = channel;
112:            channel++;
113:            if (channel > MAX_CHANNEL)
114:                channel = MIN_CHANNEL;    // wrap around
115:            return channel;
116:        } else
117:            return -1;
118:    }
119:
120:    /** Decrements the channel.
121:    *
122:    * If power is off, there is no effect (-1 returned).
123:    * Otherwise, updates the channel setting appropriately.
124:    *
125:    * If channel had been set to the minimum of the valid range of
126:    * channels, the effect will be to 'wrap' around resulting in the
127:    * channel being set to the maximum channel.
128:    *
129:    * @return The resulting channel setting
130:    */
```

```
131:     int channelDown() {
132:         if powerOn {
133:             prevChan = channel;
134:             channel--;
135:             if (channel < MIN_CHANNEL)
136:                 channel = MAX_CHANNEL;    // wrap around
137:             return channel;
138:         } else
139:             return -1;
140:     }
141:
142:     /** Sets the channel to given number (if valid).
143:     *
144:     * If power is off, there is no effect.
145:     * If given number is illegal channel, no effect.
146:     *
147:     * @param number the desired channel number
148:     * @return true if change was enacted; false otherwise.
149:     */
150:     bool setChannel(number) {
151:         if ((powerOn) && (MIN_CHANNEL <= number) && (number <= MAX_CHANNEL)) {
152:             prevChan = channel;    // must record this before it is lost
153:             channel = number;
154:             return true;
155:         } else
156:             return false;
157:     }
158:
159:     /** Changes the channel to most recent, previously viewed.
160:     *
161:     * If power is off, there is no effect.
162:     *
163:     * @return the resulting channel setting
164:     */
165:     int jumpPrevChannel() const {
166:         if (powerOn) {
167:             int temp;
168:             temp = channel;
169:             channel = prevChan;
170:             prevChan = temp;
171:             return channel;
172:         } else
173:             return -1;
174:     }
175:
176:     /* allows private access to external function */
177:     friend ostream& operator<<(ostream&, const Television&);
178: };
179:
180: /*
181:  * Overloading the output operator.
182:  */
183: ostream& operator<<(ostream& out, const Television& tv) {
184:     out << "Power setting is currently      "
185:         << (tv.powerOn ? "true" : "false") << endl
186:         << "Channel setting is currently      "
187:         << tv.channel << endl
188:         << "(previous channel) is currently  "
189:         << tv.prevChan << endl
190:         << "Volume Setting is currently      "
191:         << tv.volume << endl
192:         << "Mute is currently                "
193:         << (tv.muted ? "true" : "false") << endl;
194:     return out;
195: }
```

```
196:
197: /** Sample unit test. */
198: int main() {
199:
200:     Television sony;    // uses the DEFAULT constructor
201:     cout << "Newly created television:" << endl;
202:     cout << sony << endl << endl;
203:
204:     sony.channelUp();
205:     cout << "After call to channelUp():" << endl;
206:     cout << sony << endl << endl;
207:
208:     sony.togglePower();
209:     cout << "After call to togglePower():" << endl;
210:     cout << sony << endl << endl;
211:
212:     sony.setChannel(22);
213:     cout << "After call to setChannel(22):" << endl;
214:     cout << sony << endl << endl;
215:
216:     sony.jumpPrevChannel();
217:     cout << "After call to jumpPrevChannel():" << endl;
218:     cout << sony << endl << endl;
219:
220:     sony.jumpPrevChannel();
221:     cout << "After another call to jumpPrevChannel():" << endl;
222:     cout << sony << endl << endl;
223:
224:     sony.channelUp();
225:     cout << "After call to channelUp():" << endl;
226:     cout << sony << endl << endl;
227:
228:     sony.jumpPrevChannel();
229:     cout << "After call to jumpPrevChannel():" << endl;
230:     cout << sony << endl << endl;
231:
232:     sony.toggleMute();
233:     cout << "After call to toggleMute():" << endl;
234:     cout << sony << endl << endl;
235:
236:     sony.volumeUp();
237:     cout << "After call to volumeUp():" << endl;
238:     cout << sony << endl << endl;
239:
240:     // try to max-out the volume
241:     for (int i=0; i<250; i++)
242:         sony.volumeUp();
243:     cout << "After 250 calls to volumeUp():" << endl;
244:     cout << sony << endl << endl;
245:
246:     // try to wrap-around the channel
247:     for (int i=0; i<250; i++)
248:         sony.channelDown();
249:     cout << "After 250 calls to channelDown():" << endl;
250:     cout << sony << endl << endl;
251: }
```