Deakin University

Object Oriented Development

OnTrack Submission

The MyTime Class

Submitted By:
Peter Stacey
pstacey
2020/03/23 14:18

Tutor:
Dipto Pratyaksa

Outcome	**Weight**
Evaluate Code	♦♦♦◊◊
Principles	♦♦♦◊◊
Build Programs	♦♦♦◊◊
Design	♦♦♦◊◊
Justify	♦♦♦◊◊

This task involved broader reading of the C# reference and evaluation of different approaches to achieve an efficient implementation. Additionally, the task provided a set of specifications without providing any code, so the design needed to be developed as part of the task and then implemented in the time class and the associated testing. Defining a class for the time and implementing it in testing aligns well with the object oriented principles of the subject.

March 23, 2020

```
using System;

namespace Task_2._3C
{
    class TestMyTime
    {
        static void Main(string[] args)
        {
            int thrown = 0;

            Console.WriteLine("\n**************");
            Console.WriteLine("TESTING START");
            Console.WriteLine("**************");

            // ------------------------------------------------------------------
            // Class instantiation
            // ------------------------------------------------------------------
            MyTime time1 = new MyTime(); // Empty constructor
            MyTime time2 = new MyTime(10, 36, 45); // hour, min, sec

            try
            {
                MyTime time3 = new MyTime(100, 100, 100);
            }
            catch (ArgumentOutOfRangeException ex)
            {
                thrown++;
                Console.WriteLine("ERROR {0}: Constructor: {1}",
                    thrown, ex.Message);  // 1, expect to be thrown
            }

            /// ------------------------------------------------------------------
            // Property getters
            // ------------------------------------------------------------------
            Console.WriteLine("Property get time1 hour: {0}, min: {1}, sec {2}",
                time1.Hour, time1.Minute, time1.Second); // expect 0, 0 ,0
            Console.WriteLine("Property get time2 hour: {0}, min: {1}, sec {2}",
                time2.Hour, time2.Minute, time2.Second); // expect 10, 36, 45

            // ------------------------------------------------------------------
            // Property setters
            // ------------------------------------------------------------------

            // Reasonable values, no error expected
            try
            {
                time1.Hour = 10;
                time1.Minute = 10;
                time1.Second = 10;
            }
            catch (ArgumentOutOfRangeException ex)
            {
                thrown++;
```

```
54                  Console.WriteLine("Property Setters: Should not be thrown, {0}:
                    ↪ {1}",
55                      thrown, ex.Message);
56              }
57
58              // Invalid hour
59              try
60              {
61                  time1.Hour = 30;
62              }
63              catch (ArgumentOutOfRangeException ex)
64              {
65                  thrown++;
66                  Console.WriteLine("ERROR {0}: Hour Property: {1}",
67                      thrown, ex.Message); // 2, expect to be thrown
68              }
69
70              // Invalid minute
71              try
72              {
73                  time1.Minute = -15;
74              }
75              catch (ArgumentOutOfRangeException ex)
76              {
77                  thrown++;
78                  Console.WriteLine("ERROR {0}: Minute Property: {1}",
79                      thrown, ex.Message); // 3, expect to be thrown
80              }
81
82              // Invalid second
83              try
84              {
85                  // time1.Second = "thirty";  // syntax error
86                  // time1.Second = 13.5;      // syntax error
87                  time1.Second = 60;
88              }
89              catch (ArgumentOutOfRangeException ex)
90              {
91                  thrown++;
92                  Console.WriteLine("ERROR {0}: Second Property: {1}",
93                      thrown, ex.Message); // 4, expect to be thrown
94              }
95
96              // ------------------------------------------------------------------
97              // SetTime
98              // ------------------------------------------------------------------
99
100             // SetTime (reasonable values, no error expected)
101             try
102             {
103                 time2.SetTime(20, 15, 30);
104             }
105             catch (ArgumentOutOfRangeException ex)
```

```
106             {
107                 thrown++;
108                 Console.WriteLine("SetTime: Should not be thrown {0}: {1}",
109                     thrown, ex.Message);
110             }
111 
112             // SetTime (bad hour)
113             try
114             {
115                 time2.SetTime(24, 15, 30);
116             }
117             catch (ArgumentOutOfRangeException ex)
118             {
119                 thrown++;
120                 Console.WriteLine("ERROR {0}: SetTime: {1}", thrown, ex.Message);
                    ↪ // 5, expect to be thrown
121             }
122 
123             // SetTime (bad minute)
124             try
125             {
126                 time2.SetTime(20, 60, 30);
127             }
128             catch (ArgumentOutOfRangeException ex)
129             {
130                 thrown++;
131                 Console.WriteLine("ERROR {0}: SetTime: {1}", thrown, ex.Message);
                    ↪ // 6, expect to be thrown
132             }
133 
134             // SetTime (bad second)
135             try
136             {
137                 time2.SetTime(20, 15, 100);
138             }
139             catch (ArgumentOutOfRangeException ex)
140             {
141                 thrown++;
142                 Console.WriteLine("ERROR {0}: SetTime: {1}", thrown, ex.Message);
                    ↪ // 7, expect to be thrown
143             }
144 
145             // ------------------------------------------------------------------
146             // SetHour
147             // ------------------------------------------------------------------
148 
149             // Reasonable value, no error expected
150             try
151             {
152                 time1.SetHour(20);
153             }
154             catch (ArgumentOutOfRangeException ex)
155             {
```

```
156                 thrown++;
157                 Console.WriteLine("SetHour: Should not be thrown {0}: {1}",
158                     thrown, ex.Message);
159             }
160 
161             // SetTime (bad hour)
162             try
163             {
164                 time1.SetHour(70);
165             }
166             catch (ArgumentOutOfRangeException ex)
167             {
168                 thrown++;
169                 Console.WriteLine("ERROR {0}: SetHour: {1}", thrown, ex.Message);
                    ↪ // 8, expect to be thrown
170             }
171 
172             // ------------------------------------------------------------
173             // SetMinute
174             // ------------------------------------------------------------
175 
176             // Reasonable value, no error expected
177             try
178             {
179                 time1.SetMinute(20);
180             }
181             catch (ArgumentOutOfRangeException ex)
182             {
183                 thrown++;
184                 Console.WriteLine("SetMinute: Should not be thrown {0}: {1}",
185                     thrown, ex.Message);
186             }
187 
188             // SetTime (bad minute)
189             try
190             {
191                 time1.SetMinute(70);
192             }
193             catch (ArgumentOutOfRangeException ex)
194             {
195                 thrown++;
196                 Console.WriteLine("ERROR {0}: SetMinute: {1}", thrown, ex.Message);
                    ↪ // 9, expect to be thrown
197             }
198 
199             // ------------------------------------------------------------
200             // SetSecond
201             // ------------------------------------------------------------
202 
203             // Reasonable value, no error expected
204             try
205             {
206                 time1.SetSecond(20);
```

```
207             }
208             catch (ArgumentOutOfRangeException ex)
209             {
210                 thrown++;
211                 Console.WriteLine("SetSecond: Should not be thrown {0}: {1}",
212                     thrown, ex.Message);
213             }
214 
215             // SetTime (bad second)
216             try
217             {
218                 time1.SetSecond(70);
219             }
220             catch (ArgumentOutOfRangeException ex)
221             {
222                 thrown++;
223                 Console.WriteLine("ERROR {0}: SetSecond: {1}", thrown, ex.Message);
                    // 10, expect to be thrown
224             }
225 
226             // ------------------------------------------------------------
227             // GetHour, GetMinute, GetSecond
228             // ------------------------------------------------------------
229             time2.SetTime(10, 20, 30);
230 
231             int hour = time2.GetHour();
232             int minute = time2.GetMinute();
233             int second = time2.GetSecond();
234 
235             Console.WriteLine("GetHour, GetMinute, GetSecond: Expect 10:20:30 -
                {0}:{1}:{2}",
236                 hour, minute, second);
237 
238             // ------------------------------------------------------------
239             // ToString
240             // Also tests Format
241             // ------------------------------------------------------------
242             time1.SetTime(1, 2, 3);
243             Console.WriteLine("ToString: Expect 01:02:03 - Result {0}",
                time1.ToString());
244 
245             time2.SetTime(10, 5, 35);
246             Console.WriteLine("ToString: Expect 10:05:35 - Result {0}",
                time2.ToString());
247 
248             // ------------------------------------------------------------
249             // NextSecond, NextMinute, NextHour
250             // Also tests Format
251             // ------------------------------------------------------------
252             time1.SetTime(23, 59, 55);
253 
254             Console.WriteLine();
255             for (int i = 0; i < 10; i++)
```

```
256             {
257                 time1.NextSecond();
258                 Console.WriteLine(time1.ToString());
259             }
260 
261             // ------------------------------------------------------------------
262             // PreviousSecond, PreviousMinute, PreviousHour
263             // Also tests Format
264             // ------------------------------------------------------------------
265             Console.WriteLine();
266             for (int i = 0; i < 10; i++)
267             {
268                 time1.PreviousSecond();
269                 Console.WriteLine(time1.ToString());
270             }
271 
272             Console.WriteLine("\n***********");
273             Console.WriteLine("TESTING END");
274             Console.WriteLine("***********\n");
275         }
276     }
277 }
```

```cs
using System;

namespace Task_2._3C
{
    class MyTime
    {
        // Instance variables
        private int _hour;
        private int _minute;
        private int _second;

        /// Reference for this approach, from:
        /// https://stackoverflow.com/questions/56197825
        public int Hour
        {
            get => _hour;
            set => _hour = (value >= 0) && (value <= 23)
                ? value
                : throw new ArgumentOutOfRangeException("Invalid hour. Must be
                ↪  0-23");
        }

        public int Minute
        {
            get => _minute;
            set => _minute = (value >= 0) && (value <= 59)
                ? value
                : throw new ArgumentOutOfRangeException("Invalid minute. Must be
                ↪  0-59");
        }

        public int Second
        {
            get => _second;
            set => _second = (value >= 0) && (value <= 59)
                ? value
                : throw new ArgumentOutOfRangeException("Invalid second. Must be
                ↪  0-59");
        }

        /// <summary>
        /// Constructor to create new time with 0 values
        /// </summary>
        public MyTime() { }

        /// <summary>
        /// Constructor to create a time with hour, minute and second
        /// </summary>
        /// <param name="hour">Hour in the range 0-23</param>
        /// <param name="minute">Minute in the range 0-59</param>
        /// <param name="second">Second in the range 0-59</param>
        /// <exception cref="System.ArgumentOutOfRangeException">Thrown
        /// when one of the parameters is outside the range</exception>
```

```
51          public MyTime(int hour, int minute, int second)
52          {
53              Hour = hour;
54              Minute = minute;
55              Second = second;
56          }
57
58          /// <summary>
59          /// Sets a time with hour, minute and second
60          /// </summary>
61          /// <param name="hour">Hour in the range 0-23</param>
62          /// <param name="minute">Minute in the range 0-59</param>
63          /// <param name="second">Second in the range 0-59</param>
64          /// <exception cref="System.ArgumentOutOfRangeException">Thrown
65          /// when one of the parameters is outside the range</exception>
66          public void SetTime(int hour, int minute, int second)
67          {
68              Hour = hour;
69              Minute = minute;
70              Second = second;
71          }
72
73          /// <summary>
74          /// Sets the hour of a time
75          /// </summary>
76          /// <param name="hour">Hour in the range 0-23</param>
77          /// <exception cref="System.ArgumentOutOfRangeException">Thrown
78          /// when hour is outside the range 0-23</exception>
79          public void SetHour(int hour)
80          {
81              Hour = hour;
82          }
83
84          /// <summary>
85          /// CSets the minute of a time
86          /// </summary>
87          /// <param name="minute">Minute in the range 0-59</param>
88          /// <exception cref="System.ArgumentOutOfRangeException">Thrown
89          /// when minute is outside the range 0-59</exception>
90          public void SetMinute(int minute)
91          {
92              Minute = minute;
93          }
94
95          /// <summary>
96          /// Sets the second of a time
97          /// </summary>
98          /// <param name="second">Second in the range 0-59</param>
99          /// <exception cref="System.ArgumentOutOfRangeException">Thrown
100         /// when second is outside the range 0-59</exception>
101         public void SetSecond(int second)
102         {
103             Second = second;
```

```
104         }
105
106         /// <summary>
107         /// Gets the hour of a time
108         /// </summary>
109         /// <returns>
110         /// The hour of the time
111         /// </returns>
112         public int GetHour()
113         {
114             return _hour;
115         }
116
117         /// <summary>
118         /// Gets the minute of a time
119         /// </summary>
120         /// <returns>
121         /// The minute of the time
122         /// </returns>
123         public int GetMinute()
124         {
125             return _minute;
126         }
127
128         /// <summary>
129         /// Gets the second of a time
130         /// </summary>
131         /// <returns>
132         /// The second of the time
133         /// </returns>
134         public int GetSecond()
135         {
136             return _second;
137         }
138
139         /// <summary>
140         /// Formats a number to two digits with leading 0 if needed
141         /// </summary>
142         /// <returns>
143         /// The formatted string
144         /// </returns>
145         /// <param name="value">The integer to format as a 2-digit string</param>
146         private String Format(int value)
147         {
148             return value < 10 ? "0" + value : value.ToString();
149         }
150
151         public override String ToString()
152         {
153             return Format(_hour) + ":" + Format(_minute) + ":" + Format(_second);
154         }
155
156         /// <summary>
```

```
157         /// Advances a time by 1 second
158         /// </summary>
159         public MyTime NextSecond()
160         {
161             try
162             {
163                 Second += 1;
164             }
165             catch (ArgumentOutOfRangeException)
166             {
167                 Second = 0;
168                 NextMinute();
169             }
170             return this;
171         }
172 
173         /// <summary>
174         /// Advances a time by 1 minute
175         /// </summary>
176         public MyTime NextMinute()
177         {
178             try
179             {
180                 Minute += 1;
181             }
182             catch (ArgumentOutOfRangeException)
183             {
184                 Minute = 0;
185                 NextHour();
186             }
187             return this;
188         }
189 
190         /// <summary>
191         /// Advances a time by 1 hour
192         /// </summary>
193         public MyTime NextHour()
194         {
195             try
196             {
197                 Hour += 1;
198             }
199             catch (ArgumentOutOfRangeException)
200             {
201                 Hour = 0;
202             }
203             return this;
204         }
205 
206         /// <summary>
207         /// Reduces a time by 1 second
208         /// </summary>
209         public MyTime PreviousSecond()
```

```
210         {
211             try
212             {
213                 Second -= 1;
214             }
215             catch (ArgumentOutOfRangeException)
216             {
217                 Second = 59;
218                 PreviousMinute();
219             }
220             return this;
221         }
222 
223         /// <summary>
224         /// Reduces a time by 1 minute
225         /// </summary>
226         public MyTime PreviousMinute()
227         {
228             try
229             {
230                 Minute -= 1;
231             }
232             catch (ArgumentOutOfRangeException)
233             {
234                 Minute = 59;
235                 PrevioustHour();
236             }
237             return this;
238         }
239 
240         /// <summary>
241         /// Reduces a time by 1 hour
242         /// </summary>
243         public MyTime PrevioustHour()
244         {
245             try
246             {
247                 Hour -= 1;
248             }
249             catch (ArgumentOutOfRangeException)
250             {
251                 Hour = 23;
252             }
253             return this;
254         }
255     }
256 }
```