Deakin University

Object Oriented Development

OnTrack Submission

An Enhanced Reaction-Timer Controller

Submitted By:
Peter Stacey
pstacey
2020/05/14 22:03

Tutor:
Dipto Pratyaksa

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

As an extension of 5.3D, this task involves evaluating both the requirements of the task and our existing code in order to design the additional features. The design component involves adding to our existing design and producing a new state transition diagram as evidence of the design updates.This also involves writing addition code to implement the design and developing a suite of tests to ensure the design is correct. For this task, I have used the C# unit testing framework and developed a set of tests that are demonstrated in my video.My video and the attached files provide evidence of meeting the outcomes, which aligns with the last outcome.

May 14, 2020

```
using System.Windows.Forms;

namespace SimpleReactionMachine
{
    public class EnhancedReactionController : IController
    {
        // Settings for the game times
        private const int MAX_READY_TIME = 1000; // Maximum time in ready without pressing GoStop
        private const int MIN_WAIT_TIME = 100;   // Minimum wait time, 1 sec in ticks
        private const int MAX_WAIT_TIME = 250;   // Maximum wait time, 2.5 sec in ticks
        private const int MAX_GAME_TIME = 200;   // Maximum of 2 sec to react, in ticks
        private const int GAMEOVER_TIME = 300;   // Display result for 3 sec, in ticks
        private const int RESULTS_TIME = 500;    // Display average time for 5 sec, in ticks
        private const double TICKS_PER_SECOND = 100.0; // Based on 10ms ticks

        // Instance variables and properties
        private State _state;
        private IGui Gui { get; set; }
        private IRandom Rng { get; set; }
        private int Ticks { get; set; }
        private int Games { get; set; }
        private int TotalReactionTime { get; set; }

        /// <summary>
        /// Connects the controller to the Gui and Random Number Generator
        /// </summary>
        /// <param name="gui">IGui concrete implementation</param>
        /// <param name="rng">IRandom concreate implementation</param>
        public void Connect(IGui gui, IRandom rng)
        {
            Gui = gui;
            Rng = rng;
            Init();
        }

        /// <summary>
        /// Initialises the state of the controller at the start of the program
        /// </summary>
        public void Init() => _state = new OnState(this);

        /// <summary>
        /// Coin inserted event handler
        /// </summary>
        public void CoinInserted() => _state.CoinInserted();

        /// <summary>
        /// Go/Stop pressed event handler
```

```
48          /// </summary>
49          public void GoStopPressed() => _state.GoStopPressed();
50
51          /// <summary>
52          /// Tick event handler
53          /// </summary>
54          public void Tick() => _state.Tick();
55
56          /// <summary>
57          /// Sets the state of the controller to the desired state
58          /// </summary>
59          /// <param name="state">The new state to transition to</param>
60          void SetState(State state) => _state = state;
61
62          /// <summary>
63          /// Base class for concrete State classes
64          /// </summary>
65          abstract class State
66          {
67              protected EnhancedReactionController controller;
68              public State(EnhancedReactionController con) => controller = con;
69              public abstract void CoinInserted();
70              public abstract void GoStopPressed();
71              public abstract void Tick();
72          }
73
74          /// <summary>
75          /// State of the game when it is waiting for a coin to be inserted
76          /// </summary>
77          class OnState : State
78          {
79              public OnState(EnhancedReactionController con) : base(con)
80              {
81                  controller.Games = 0;
82                  controller.TotalReactionTime = 0;
83                  controller.Gui.SetDisplay("Insert coin");
84              }
85              public override void CoinInserted() => controller.SetState(new
                    ↪ ReadyState(controller));
86              public override void GoStopPressed() { }
87              public override void Tick() { }
88          }
89
90          /// <summary>
91          /// State of the game when a coin has been inserted, but the game is not yet
92          /// started
93          /// </summary>
94          class ReadyState : State
95          {
96              public ReadyState(EnhancedReactionController con) : base(con)
97              {
98                  controller.Gui.SetDisplay("Press Go!");
99                  controller.Ticks = 0;
```

```
100             }
101             public override void CoinInserted() { }
102             public override void GoStopPressed()
103             {
104                 controller.SetState(new WaitState(controller));
105             }
106             public override void Tick()
107             {
108                 controller.Ticks++;
109                 if (controller.Ticks == MAX_READY_TIME)
110                     controller.SetState(new OnState(controller));
111             }
112         }
113 
114         /// <summary>
115         /// State of the game when the game has started and it is waiting for the
116         /// random time
117         /// </summary>
118         class WaitState : State
119         {
120             private int _waitTime;
121             public WaitState(EnhancedReactionController con) : base(con)
122             {
123                 controller.Gui.SetDisplay("Wait...");
124                 controller.Ticks = 0;
125                 _waitTime = controller.Rng.GetRandom(MIN_WAIT_TIME, MAX_WAIT_TIME);
126             }
127             public override void CoinInserted() { }
128             public override void GoStopPressed() => controller.SetState(new
                ↪ OnState(controller));
129             public override void Tick()
130             {
131                 controller.Ticks++;
132                 if (controller.Ticks == _waitTime)
133                 {
134                     controller.Games++;
135                     controller.SetState(new RunningState(controller));
136                 }
137             }
138         }
139 
140         /// <summary>
141         /// State of the game when the timer is counting and it is waiting for the
142         /// user to react by pressing the Go/Stop button
143         /// </summary>
144         class RunningState : State
145         {
146             public RunningState(EnhancedReactionController con) : base(con)
147             {
148                 controller.Gui.SetDisplay("0.00");
149                 controller.Ticks = 0;
150             }
151             public override void CoinInserted() { }
```

```
152             public override void GoStopPressed()
153             {
154                 controller.TotalReactionTime += controller.Ticks;
155                 controller.SetState(new GameOverState(controller));
156             }
157             public override void Tick()
158             {
159                 controller.Ticks++;
160                 controller.Gui.SetDisplay(
161                     (controller.Ticks / TICKS_PER_SECOND).ToString("0.00"));
162                 if (controller.Ticks == MAX_GAME_TIME)
163                     controller.SetState(new GameOverState(controller));
164             }
165         }
166 
167         /// <summary>
168         /// State of the game when the time has expired, or the user reacted.
169         /// If 3 games not yet played, sets the state to Wait, otherwise to
170         /// Results
171         /// </summary>
172         class GameOverState : State
173         {
174             public GameOverState(EnhancedReactionController con) : base(con)
175             {
176                 controller.Ticks = 0;
177             }
178             public override void CoinInserted() { }
179             public override void GoStopPressed() => CheckGames();
180             public override void Tick()
181             {
182                 controller.Ticks++;
183                 if (controller.Ticks == GAMEOVER_TIME)
184                     CheckGames();
185             }
186             private void CheckGames()
187             {
188                 if (controller.Games == 3)
189                 {
190                     controller.SetState(new ResultsState(controller));
191                     return;
192                 }
193                 controller.SetState(new WaitState(controller));
194             }
195         }
196 
197         /// <summary>
198         /// Shows the average reaction time for the 3 games played, for
199         /// 5 seconds, or until GoStop is pressed
200         /// </summary>
201         class ResultsState : State
202         {
203             public ResultsState(EnhancedReactionController con) : base(con)
204             {
```

```
205                 controller.Gui.SetDisplay("Average: "
206                     + ((double)controller.TotalReactionTime / controller.Games *
                        ↪ 0.01)
207                     .ToString("0.00"));
208                 controller.Ticks = 0;
209             }
210             public override void CoinInserted() { }
211             public override void GoStopPressed() => controller.SetState(new
                ↪ OnState(controller));
212             public override void Tick()
213             {
214                 controller.Ticks++;
215                 if (controller.Ticks == RESULTS_TIME)
216                     controller.SetState(new OnState(controller));
217             }
218         }
219     }
220 }
```

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using SimpleReactionMachine;
using System;

namespace EnhancedSimpleReactionControllerTests
{
    [TestClass]
    public class EnhancedReactionControllerTests
    {
        private static IController controller;
        private static IGui gui;
        private static IRandom rng;
        private static string displayText;
        private static int RandomNumber { get; set; }

        [TestMethod]
        public void Create_Controller()
        {
            // Tests that the controller can be created and is not null
            // after creation
            controller = new EnhancedReactionController();
            Assert.IsNotNull(controller);
        }

        [TestMethod]
        public void Connect_And_Initialise_Controller()
        {
            controller = new EnhancedReactionController();
            gui = new DummyGui();
            gui.Connect(controller);
            controller.Connect(gui, new RndGenerator());

            // Controller Init() sets initial state to OnState
            // and display should be "Insert coin"
            controller.Init();
            Assert.AreEqual("Insert coin", displayText);
        }

        [TestMethod]
        public void Test_OnState_GoStopPressed()
        {
            controller = new EnhancedReactionController();
            gui = new DummyGui();
            rng = new RndGenerator();
            InitialiseToOnState(controller, gui, rng);

            // GoStopPressed has no effect in OnState
            Assert.AreEqual("Insert coin", displayText);
            controller.GoStopPressed();
            Assert.AreEqual("Insert coin", displayText);
        }

        [TestMethod]
```

```
54          public void Test_OnState_Tick()
55          {
56              controller = new EnhancedReactionController();
57              gui = new DummyGui();
58              rng = new RndGenerator();
59              InitialiseToOnState(controller, gui, rng);
60
61              // Tick has no effect in OnState
62              Assert.AreEqual("Insert coin", displayText);
63              controller.Tick();
64              Assert.AreEqual("Insert coin", displayText);
65          }
66
67          [TestMethod]
68          public void Test_OnState_CoinInserted()
69          {
70              controller = new EnhancedReactionController();
71              gui = new DummyGui();
72              rng = new RndGenerator();
73              InitialiseToOnState(controller, gui, rng);
74
75              // Inserting a coin sets the state to ReadyState
76              // Display should then be "Press Go!"
77              Assert.AreEqual("Insert coin", displayText);
78              controller.CoinInserted();
79              Assert.AreEqual("Press Go!", displayText);
80          }
81
82          [TestMethod]
83          public void Test_ReadyState_CoinInserted()
84          {
85              controller = new EnhancedReactionController();
86              gui = new DummyGui();
87              rng = new RndGenerator();
88              InitialiseToReadyState(controller, gui, rng);
89
90              // Inserting a coin has no effect in ReadyState
91              Assert.AreEqual("Press Go!", displayText);
92              controller.CoinInserted();
93              Assert.AreEqual("Press Go!", displayText);
94          }
95
96          [TestMethod]
97          public void Test_ReadyState_Tick()
98          {
99              controller = new EnhancedReactionController();
100             gui = new DummyGui();
101             rng = new RndGenerator();
102             InitialiseToReadyState(controller, gui, rng);
103
104             // Tick has no effect in ReadyState
105             Assert.AreEqual("Press Go!", displayText);
106             controller.Tick();
```

```
107             Assert.AreEqual("Press Go!", displayText);
108         }
109 
110         [TestMethod]
111         public void Test_ReadyState_GoStopPressed()
112         {
113             controller = new EnhancedReactionController();
114             gui = new DummyGui();
115             rng = new RndGenerator();
116             InitialiseToReadyState(controller, gui, rng);
117 
118             // Pressing Go/Stop sets the state to WaitState
119             // Display should then be "Wait..."
120             Assert.AreEqual("Press Go!", displayText);
121             controller.GoStopPressed();
122             Assert.AreEqual("Wait...", displayText);
123         }
124 
125         [TestMethod]
126         public void Test_ReadyState_Too_Long()
127         {
128             controller = new EnhancedReactionController();
129             gui = new DummyGui();
130             rng = new RndGenerator();
131             InitialiseToReadyState(controller, gui, rng);
132 
133             // Waiting for 10 seconds in WaitState resets the
134             // controller back to OnState
135             // Display should then be "Insert coin"
136             for (int t = 0; t < 999; t++) controller.Tick();
137             Assert.AreEqual("Press Go!", displayText);
138             controller.Tick();
139             Assert.AreEqual("Insert coin", displayText);
140         }
141 
142         [TestMethod]
143         public void Test_WaitState_CoinInserted()
144         {
145             controller = new EnhancedReactionController();
146             gui = new DummyGui();
147             rng = new RndGenerator();
148             InitialiseToWaitState(controller, gui, rng);
149 
150             // Inserting a coin has no effect in WaitState
151             Assert.AreEqual("Wait...", displayText);
152             controller.CoinInserted();
153             Assert.AreEqual("Wait...", displayText);
154         }
155 
156         [TestMethod]
157         public void Test_WaitState_GoStopPressed()
158         {
159             controller = new EnhancedReactionController();
```

```
160             gui = new DummyGui();
161             rng = new RndGenerator();
162             InitialiseToWaitState(controller, gui, rng);
163 
164             // GoStopPressed in the WaitState is considered
165             // cheating and it sets the game back to the OnState
166             // Display should then by "Insert coin"
167             Assert.AreEqual("Wait...", displayText);
168             controller.GoStopPressed();
169             Assert.AreEqual("Insert coin", displayText);
170         }
171 
172         [TestMethod]
173         public void Test_WaitState_Tick()
174         {
175             controller = new EnhancedReactionController();
176             gui = new DummyGui();
177             rng = new RndGenerator();
178             InitialiseToWaitState(controller, gui, rng);
179 
180             // After the rendom wait time, the controller should
181             // be set to the RunningState
182             // Display should then be "0.00"
183             for (int t = 0; t < RandomNumber - 1; t++) controller.Tick();
184             Assert.AreEqual(displayText, "Wait...");
185             controller.Tick();
186             Assert.AreEqual(displayText, "0.00");
187         }
188 
189         [TestMethod]
190         public void Test_RunningState_CoinInserted()
191         {
192             controller = new EnhancedReactionController();
193             gui = new DummyGui();
194             rng = new RndGenerator();
195             InitialiseToRunningState(controller, gui, rng);
196 
197             // CoinInserted has no effect in the RunningState
198             Assert.AreEqual("0.00", displayText);
199             controller.CoinInserted();
200             Assert.AreEqual("0.00", displayText);
201         }
202 
203         [TestMethod]
204         public void Test_RunningState_Tick()
205         {
206             controller = new EnhancedReactionController();
207             gui = new DummyGui();
208             rng = new RndGenerator();
209             InitialiseToRunningState(controller, gui, rng);
210 
211             // Ticks advance the time display in the RunningState
212             Assert.AreEqual("0.00", displayText);
```

```
213             controller.Tick();
214             Assert.AreEqual("0.01", displayText);
215 
216             for (int t = 0; t < 10; t++) controller.Tick();
217             Assert.AreEqual("0.11", displayText);
218 
219             for (int t = 0; t < 100; t++) controller.Tick();
220             Assert.AreEqual("1.11", displayText);
221 
222             // GoStopPressed should advance to the GameOverState
223             // and no further update to the display
224             controller.GoStopPressed();
225             Assert.AreEqual("1.11", displayText);
226         }
227 
228         [TestMethod]
229         public void Test_RunningState_GoStopPressed()
230         {
231             controller = new EnhancedReactionController();
232             gui = new DummyGui();
233             rng = new RndGenerator();
234             InitialiseToRunningState(controller, gui, rng);
235 
236             // GoStopPressed records the reaction time in the RunningState
237             // and advances the controller to the GameOverState
238             // Display should be the same as the reaction time when
239             // GoStop is pressed
240             for (int t = 0; t < 164; t++) controller.Tick();
241             Assert.AreEqual("1.64", displayText);
242             controller.GoStopPressed();
243             Assert.AreEqual("1.64", displayText);
244         }
245 
246         [TestMethod]
247         public void Test_RunningState_Tick_Two_Seconds()
248         {
249             controller = new EnhancedReactionController();
250             gui = new DummyGui();
251             rng = new RndGenerator();
252             InitialiseToRunningState(controller, gui, rng);
253 
254             // Not reacting in 2 seconds automatically ends the game
255             // Display should show 2.00 seconds
256             for (int t = 0; t < 199; t++) controller.Tick();
257             Assert.AreEqual("1.99", displayText);
258             controller.Tick();
259             Assert.AreEqual("2.00", displayText);
260             controller.Tick();
261             Assert.AreEqual("2.00", displayText);
262         }
263 
264         [TestMethod]
265         public void Test_GameOverState_CoinInserted()
```

```
266         {
267             controller = new EnhancedReactionController();
268             gui = new DummyGui();
269             rng = new RndGenerator();
270             InitialiseToRunningState(controller, gui, rng);
271 
272             // Inserting a coin has no effect in GameOverState
273             for (int t = 0; t < 22; t++) controller.Tick();
274             Assert.AreEqual("0.22", displayText);
275             controller.CoinInserted();
276             Assert.AreEqual("0.22", displayText);
277         }
278 
279         [TestMethod]
280         public void Test_GameOverState_Tick()
281         {
282             controller = new EnhancedReactionController();
283             gui = new DummyGui();
284             rng = new RndGenerator();
285             InitialiseToRunningState(controller, gui, rng);
286 
287             // Tick shows the reaction time and then sets the
288             // controller to the WaitState
289             // NOTE: This test does not test the transition
290             // to the ResultState. That is tested in
291             // Test_Play_Three_Games_And_Wait_Ticks
292             for (int t = 0; t < 50; t++) controller.Tick();
293             controller.GoStopPressed();
294             Assert.AreEqual("0.50", displayText);
295             for (int t = 0; t < 299; t++) controller.Tick();
296             Assert.AreEqual("0.50", displayText);
297             controller.Tick();
298             Assert.AreEqual("Wait...", displayText);
299         }
300 
301         [TestMethod]
302         public void Test_GameOver_GoStopPressed()
303         {
304             controller = new EnhancedReactionController();
305             gui = new DummyGui();
306             rng = new RndGenerator();
307             InitialiseToRunningState(controller, gui, rng);
308 
309             // GoStopPressed immediately ends the GameOverState
310             // and sets the state to WaitState
311             // NOTE: This does not test the state moving
312             // to the ResultState after 3 games. That is tested in
313             // Test_Play_Three_Games_And_GoStopPressed
314             for (int t = 0; t < 56; t++) controller.Tick();
315             controller.GoStopPressed();
316             Assert.AreEqual("0.56", displayText);
317             controller.GoStopPressed();
318             Assert.AreEqual("Wait...", displayText);
```

```
319         }
320
321         [TestMethod]
322         public void Test_Play_Three_Games_And_Wait_Ticks()
323         {
324             controller = new EnhancedReactionController();
325             gui = new DummyGui();
326             rng = new RndGenerator();
327             InitialiseToRunningState(controller, gui, rng);
328
329             // Run three games and then wait the final 3 seconds
330             // State should advance to ResultState
331             // Display should then show the average reaction time
332             for (int t = 0; t < 20; t++) controller.Tick();
333             controller.GoStopPressed();
334             Assert.AreEqual("0.20", displayText);
335             for (int t = 0; t < 299; t++) controller.Tick();
336             Assert.AreEqual("0.20", displayText);
337             controller.Tick();
338             Assert.AreEqual("Wait...", displayText);
339
340             for (int t = 0; t < RandomNumber + 30; t++) controller.Tick();
341             controller.GoStopPressed();
342             Assert.AreEqual("0.30", displayText);
343             for (int t = 0; t < 299; t++) controller.Tick();
344             Assert.AreEqual("0.30", displayText);
345             controller.Tick();
346             Assert.AreEqual("Wait...", displayText);
347
348             for (int t = 0; t < RandomNumber + 40; t++) controller.Tick();
349             controller.GoStopPressed();
350             Assert.AreEqual("0.40", displayText);
351             for (int t = 0; t < 299; t++) controller.Tick();
352             controller.Tick();
353             Assert.AreEqual("Average: 0.30", displayText);
354
355         }
356         [TestMethod]
357         public void Test_Play_Three_Games_And_GoStopPressed()
358         {
359             controller = new EnhancedReactionController();
360             gui = new DummyGui();
361             rng = new RndGenerator();
362             InitialiseToRunningState(controller, gui, rng);
363
364             // Run three games and then press GoStop
365             // State should advance to ResultState immediately
366             // Display should then show the average reaction time
367             for (int t = 0; t < 155; t++) controller.Tick();
368             controller.GoStopPressed();
369             Assert.AreEqual("1.55", displayText);
370             controller.GoStopPressed();
371             Assert.AreEqual("Wait...", displayText);
```

```
372
373             for (int t = 0; t < RandomNumber + 160; t++) controller.Tick();
374             controller.GoStopPressed();
375             Assert.AreEqual("1.60", displayText);
376             controller.GoStopPressed();
377             Assert.AreEqual("Wait...", displayText);
378
379             for (int t = 0; t < RandomNumber + 165; t++) controller.Tick();
380             controller.GoStopPressed();
381             Assert.AreEqual("1.65", displayText);
382             controller.GoStopPressed();
383             Assert.AreEqual("Average: 1.60", displayText);
384         }
385
386         [TestMethod]
387         public void Test_ResultState_CoinInserted()
388         {
389             controller = new EnhancedReactionController();
390             gui = new DummyGui();
391             rng = new RndGenerator();
392             InitialiseToRunningState(controller, gui, rng);
393
394             // Play 3 games
395             for (int t = 0; t < 10; t++) controller.Tick();
396             controller.GoStopPressed();
397             controller.GoStopPressed();
398
399             for (int t = 0; t < RandomNumber + 15; t++) controller.Tick();
400             controller.GoStopPressed();
401             controller.GoStopPressed();
402
403             for (int t = 0; t < RandomNumber + 20; t++) controller.Tick();
404             controller.GoStopPressed();
405             controller.GoStopPressed();
406
407             // Inserting a coin in the ResultState has no effect
408             Assert.AreEqual("Average: 0.15", displayText);
409             controller.CoinInserted();
410             Assert.AreEqual("Average: 0.15", displayText);
411         }
412
413         [TestMethod]
414         public void Test_ResultState_Ticks()
415         {
416             controller = new EnhancedReactionController();
417             gui = new DummyGui();
418             rng = new RndGenerator();
419             InitialiseToRunningState(controller, gui, rng);
420
421             // Play 3 games
422             for (int t = 0; t < 10; t++) controller.Tick();
423             controller.GoStopPressed();
424             controller.GoStopPressed();
```

```
425
426             for (int t = 0; t < RandomNumber + 15; t++) controller.Tick();
427             controller.GoStopPressed();
428             controller.GoStopPressed();
429
430             for (int t = 0; t < RandomNumber + 20; t++) controller.Tick();
431             controller.GoStopPressed();
432             controller.GoStopPressed();
433
434             // Ticks displays the average reaction time for 5 seconds
435             // and then the controller is set to OnState
436             // Display should then be "Insert coin"
437             Assert.AreEqual("Average: 0.15", displayText);
438             for (int i = 0; i < 499; i++) controller.Tick();
439             Assert.AreEqual("Average: 0.15", displayText);
440             controller.Tick();
441             Assert.AreEqual("Insert coin", displayText);
442         }
443
444         [TestMethod]
445         public void Test_ResultState_GoStopPressed()
446         {
447             controller = new EnhancedReactionController();
448             gui = new DummyGui();
449             rng = new RndGenerator();
450             InitialiseToRunningState(controller, gui, rng);
451
452             // Play 3 games
453             for (int t = 0; t < 10; t++) controller.Tick();
454             controller.GoStopPressed();
455             controller.GoStopPressed();
456
457             for (int t = 0; t < RandomNumber + 15; t++) controller.Tick();
458             controller.GoStopPressed();
459             controller.GoStopPressed();
460
461             for (int t = 0; t < RandomNumber + 20; t++) controller.Tick();
462             controller.GoStopPressed();
463             controller.GoStopPressed();
464
465             // GoStopPressed displays the average reaction time for 5 seconds
466             // and then the controller is set to OnState
467             // Display should then be "Insert coin"
468             Assert.AreEqual("Average: 0.15", displayText);
469             controller.GoStopPressed();
470             Assert.AreEqual("Insert coin", displayText);
471         }
472
473         // sets the controller to the OnState
474         private void InitialiseToOnState(IController controller, IGui gui, IRandom
            ↪ rng)
475         {
476             gui.Connect(controller);
```

```
477             controller.Connect(gui, rng);
478             gui.Init();
479             controller.Init();
480         }
481 
482         // sets the controller to the ReadyState
483         private void InitialiseToReadyState(IController controller, IGui gui,
            ↪ IRandom rng)
484         {
485             InitialiseToOnState(controller, gui, rng);
486             controller.CoinInserted();
487         }
488 
489         // sets the controller to the WaitState
490         private void InitialiseToWaitState(IController controller, IGui gui,
            ↪ IRandom rng)
491         {
492             InitialiseToReadyState(controller, gui, rng);
493             controller.GoStopPressed();
494         }
495 
496         // sets the controller to the RunningState
497         private void InitialiseToRunningState(IController controller, IGui gui,
            ↪ IRandom rng)
498         {
499             InitialiseToWaitState(controller, gui, rng);
500             for (int t = 0; t < RandomNumber; t++)
501                 controller.Tick();
502         }
503 
504         // Mock Gui, as implementation of the IGui, to allow the controller
505         // to Connect and SetDisplay
506         private class DummyGui : IGui
507         {
508             private IController _controller;
509             public void Connect(IController controller) => _controller = controller;
510             public void Init() => displayText = "?reset?";
511 
512             public void SetDisplay(string msg)
513             {
514                 displayText = msg;
515             }
516         }
517 
518         // IRandom implementation that also sets the RandomNumber property
519         // to allow the test to access it, as well as the values being
520         // passed to the controller
521         private class RndGenerator : IRandom
522         {
523             Random rnd = new Random(42);
524 
525             public int GetRandom(int from, int to)
526             {
```

```
527                     RandomNumber = rnd.Next(from) + to;
528                     return RandomNumber;
529                 }
530             }
531         }
532 }
```

SIT232 - Task 5.4HD
State Transition Diagram