

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 /* TITLE:      RunningCostsCalculator
8  * AUTHOR:     Paul McKillop
9  * DATE:       October 2022
10 * PURPOSE:    Calculate the costs other than the Loan costs
11 */
12
13 namespace McKillopMotoring
14 {
15     public class RunningCostsCalculator
16     {
17
18         #region Insurance
19         // - Insurance Costs
20
21         // --- Annual costs
22         public static double InsuranceCostAnnual(RunningCost myRunningCost)
23         {
24             //-- return value
25             double cost = 0;
26             string period = myRunningCost.InsurancePeriod;
27
28             switch (period)
29             {
30                 case "Annual":
31                     cost = Convert.ToDouble(myRunningCost.Insurance);
32                     break;
33                 case "Monthly":
34                     cost = Convert.ToDouble(myRunningCost.Insurance *
35                                             12);
36                     break;
37                 case "Weekly":
38                     cost = Convert.ToDouble(myRunningCost.Insurance *
39                                             52);
40                     break;
41                 default:
42                     cost = 0;
43                     break;
44             }
45
46             // --now return the value from the method based on the
47             // switch
48             return Math.Round(cost, 2);
49         }
50     }
51 }
```

```
49 // --- Monthly costs
50 public static double InsuranceCostMonthly(RunningCost myRunningCost)
51 {
52     double cost = 0;
53     string period = myRunningCost.InsurancePeriod;
54
55     switch (period)
56     {
57         case "Annual":
58             cost = Convert.ToDouble(myRunningCost.Insurance / 12);
59             break;
60         case "Monthly":
61             cost = Convert.ToDouble(myRunningCost.Insurance);
62             break;
63         case "Weekly":
64             cost = Convert.ToDouble(myRunningCost.Insurance * 52 / 12);
65             break;
66         default:
67             cost = 0;
68             break;
69     }
70
71     return Math.Round(cost, 2);
72 }
73
74 // --- Weekly costs
75 public static double InsuranceCostWeekly(RunningCost myRunningCost)
76 {
77     double cost = 0;
78     string period = myRunningCost.InsurancePeriod;
79
80     switch (period)
81     {
82         case "Annual":
83             cost = Convert.ToDouble(myRunningCost.Insurance / 52);
84             break;
85         case "Monthly":
86             cost = Convert.ToDouble(myRunningCost.Insurance * 12 / 52);
87             break;
88         case "Weekly":
89             cost = Convert.ToDouble(myRunningCost.Insurance);
90             break;
91         default:
92             cost = 0;
93             break;
94     }
95 }
```

```
96
97         return Math.Round(cost, 2);
98     }
99     #endregion
100
101     //-- Fuel
102     //-- Annual
103     #region Fuel costs
104
105     public static double FuelCostAnnual(RunningCost myRunningCost)
106     {
107         double cost = 0;
108         string period = myRunningCost.FuelPeriod;
109
110         switch (period)
111         {
112             case "Annual":
113                 cost = Convert.ToDouble(myRunningCost.Fuel);
114                 break;
115             case "Monthly":
116                 cost = Convert.ToDouble(myRunningCost.Fuel * 12);
117                 break;
118             case "Weekly":
119                 cost = Convert.ToDouble(myRunningCost.Fuel * 52);
120                 break;
121             default:
122                 cost = 0;
123                 break;
124         }
125
126         return Math.Round(cost, 2);
127     }
128
129     //-- Monthly
130     public static double FuelCostMonthly(RunningCost myRunningCost)
131     {
132         double cost = 0;
133         string period = myRunningCost.FuelPeriod;
134
135         switch (period)
136         {
137             case "Annual":
138                 cost = Convert.ToDouble(myRunningCost.Fuel / 12);
139                 break;
140             case "Monthly":
141                 cost = Convert.ToDouble(myRunningCost.Fuel);
142                 break;
143             case "Weekly":
144                 cost = Convert.ToDouble(myRunningCost.Fuel * 52 /
145                 12);
146                 break;
147             default:
```

```
148         cost = 0;
149         break;
150
151     }
152
153     return Math.Round(cost, 2);
154 }
155
156 //-- Weekly
157 public static double FuelCostWeekly(RunningCost myRunningCost)
158 {
159     double cost = 0;
160     string period = myRunningCost.FuelPeriod;
161
162     switch (period)
163     {
164         case "Annual":
165             cost = Convert.ToDouble(myRunningCost.Fuel / 52);
166             break;
167         case "Monthly":
168             cost = Convert.ToDouble(myRunningCost.Fuel * 12 / 52);
169             break;
170         case "Weekly":
171             cost = Convert.ToDouble(myRunningCost.Fuel);
172             break;
173         default:
174             cost = 0;
175             break;
176     }
177
178     return Math.Round(cost, 2);
179 }
180
181 #endregion
182
183
184 //-- Servicing
185 #region Servicing costs
186 public static double ServicingCostAnnual(RunningCost
187     myRunningCost)
188 {
189     double cost = 0;
190     string period = myRunningCost.ServicingPeriod;
191
192     switch (period)
193     {
194         case "Annual":
195             cost = Convert.ToDouble(myRunningCost.Servicing);
196             break;
197         case "Monthly":
198             cost = Convert.ToDouble(myRunningCost.Servicing *
199                 12);
```

```
198         break;
199         case "Weekly":
200             cost = Convert.ToDouble(myRunningCost.Servicing * 52);
201             break;
202         default:
203             cost = 0;
204             break;
205     }
206 }
207
208     return Math.Round(cost, 2);
209 }
210
211 public static double ServicingCostMonthly(RunningCost myRunningCost)
212 {
213     double cost = 0;
214     string period = myRunningCost.ServicingPeriod;
215
216     switch (period)
217     {
218         case "Annual":
219             cost = Convert.ToDouble(myRunningCost.Servicing / 12);
220             break;
221         case "Monthly":
222             cost = Convert.ToDouble(myRunningCost.Servicing);
223             break;
224         case "Weekly":
225             cost = Convert.ToDouble(myRunningCost.Servicing * 52 / 12);
226             break;
227         default:
228             cost = 0;
229             break;
230     }
231
232     return Math.Round(cost, 2);
233 }
234
235
236 public static double ServicingCostWeekly(RunningCost myRunningCost)
237 {
238     double cost = 0;
239     string period = myRunningCost.ServicingPeriod;
240
241     switch (period)
242     {
243         case "Annual":
244             cost = Convert.ToDouble(myRunningCost.Servicing / 52);
```

```
245         break;
246     case "Monthly":
247         cost = Convert.ToDouble(value:
myRunningCost.Servicing * 12 / 52);
248         break;
249     case "Weekly":
250         cost = Convert.ToDouble(value:
myRunningCost.Servicing);
251         break;
252     default:
253         cost = 0;
254         break;
255
256     }
257
258     return Math.Round(value: cost, digits: 2);
259 }
260 #endregion
261
262 //-- RoadTax
263 #region Road Tax Costs
264 public static double RoadTaxAnnual(RunningCost myRunningCost)
265 {
266     double cost = 0;
267     string period = myRunningCost.RoadTaxPeriod;
268
269     switch (period)
270     {
271     case "Annual":
272         cost = Convert.ToDouble(value:
myRunningCost.RoadTax);
273         break;
274     case "Monthly":
275         cost = Convert.ToDouble(value: myRunningCost.RoadTax
* 12);
276         break;
277     case "Weekly":
278         cost = Convert.ToDouble(value: myRunningCost.RoadTax
* 52);
279         break;
280     default:
281         cost = 0;
282         break;
283
284     }
285
286     return Math.Round(value: cost, digits: 2);
287 }
288
289 public static double RoadTaxMonthly(RunningCost myRunningCost)
290 {
291     double cost = 0;
292     string period = myRunningCost.RoadTaxPeriod;
```

```
295     {
296         case "Annual":
297             cost = Convert.ToDouble(value:
myRunningCost.RoadTax / 12);
298             break;
299         case "Monthly":
300             cost = Convert.ToDouble(value:
myRunningCost.RoadTax);
301             break;
302         case "Weekly":
303             cost = Convert.ToDouble(value: myRunningCost.RoadTax
* 52 / 12);
304             break;
305         default:
306             cost = 0;
307             break;
308     }
309
310     return Math.Round(value: cost, digits: 2);
311 }
312
313 public static double RoadTaxWeekly(RunningCost myRunningCost)
314 {
315     double cost = 0;
316     string period = myRunningCost.RoadTaxPeriod;
317
318     switch (period)
319     {
320     case "Annual":
321         cost = Convert.ToDouble(value:
myRunningCost.RoadTax / 52);
322         break;
323     case "Monthly":
324         cost = Convert.ToDouble(value: myRunningCost.RoadTax
* 12 / 52);
325         break;
326     case "Weekly":
327         cost = Convert.ToDouble(value:
myRunningCost.RoadTax);
328         break;
329     default:
330         cost = 0;
331         break;
332     }
333
334     return Math.Round(value: cost, digits: 2);
335 }
336
337 #endregion
338
339 //--- Total Weekly Running costs
340 public static double TotalWeeklyRunningCost(RunningCost
```

```
343         double totalWeeklyCost = 0;
344         totalWeeklyCost += InsuranceCostWeekly(myRunningCost);
345         totalWeeklyCost += FuelCostWeekly(myRunningCost);
346         totalWeeklyCost += ServicingCostWeekly(myRunningCost);
347         totalWeeklyCost += RoadTaxWeekly(myRunningCost);
348
349         return totalWeeklyCost;
350     }
351
352
353
354     #region Method Template
355     //-- Template
356     public static double CalculatorTemplate(RunningCost myRunningCost) ↗
357     {
358         double cost = 0;
359         string period = "Z";
360
361         switch (period)
362         {
363             case "Annual":
364                 cost = Convert.ToDouble(value: 0);
365                 break;
366             case "Monthly":
367                 cost = Convert.ToDouble(value: 0);
368                 break;
369             case "Weekly":
370                 cost = Convert.ToDouble(value: 0);
371                 break;
372             default:
373                 cost = 0;
374                 break;
375         }
376
377         return Math.Round(value: cost, digits: 2);
378     }
379
380     #endregion
381
382 }
383 }
384
```