```
1  //-- *************************
2  //-- CLASS:     LoanCalculator
3  //-- AUTHOR:    Paul McKillop
4  //-- CREATED:   02 December 2018
5  //-  PURPOSE:   Method only to calculate loan costs as
6  //-                Total, Yearly, Monthly, Weekly
7  //-- *************************
8
9  using System;
10 using System.Collections.Generic;
11 using System.Linq;
12 using System.Text;
13 using System.Threading.Tasks;
14
15 namespace Motoring
16 {
17     public class LoanCalculator
18     {
19         //-- create a constant used multiple time in calculations
20         private const int MonthsPerYear = 12;
21
22         /// <summary>
23         /// Monthly payment
24         /// </summary>
25         /// <param name="myLoan"></param>
26         /// <returns></returns>
27         #region Monthly Payment
28         public static decimal LoanMonthlyPayment(Loan myLoan)
29         {
30             //-- create and initialise the return variable
31             double payment = 0;
32
33             //-- create local variables from the argument class
34             var loanTermMonths = myLoan.LoanTermYears * MonthsPerYear;
35             var interestRate = Convert.ToDouble(myLoan.LoanRate);
36             var loanAmount = myLoan.CarPrice - myLoan.CarDeposit;
37
38             if (myLoan.LoanTermYears > 0) //-- Check term years not 0
39             {
40                 if (myLoan.LoanRate != 0)
41                 {
42                     var rate = (double)((interestRate / MonthsPerYear) / 100);
43                     var factor = (rate + (rate / (Math.Pow(rate + 1,
                       loanTermMonths) - 1))); //-- use built-in library for
                       compound interest
44                     payment = (loanAmount * factor);
45                 }
46                 else
47                 {
48                     payment = (loanAmount / (double)loanTermMonths);
49                 }
50             }
51
52             return Math.Round((decimal)payment, 2);
53         }
54         #endregion
```

```
55
56          //-- Annual
57          /// <summary>
58          /// Annual
59          /// </summary>
60          /// <param name="myLoan"></param>
61          /// <returns></returns>
62          #region Annual
63          public static decimal LoanAnnualPayment(Loan myLoan)
64          {
65              return Math.Round(LoanMonthlyPayment(myLoan) * 12, 2);
66          }
67          #endregion
68
69          //-- Weekly
70
71          /// <summary>
72          /// Weekly
73          /// </summary>
74          /// <param name="myLoan"></param>
75          /// <returns></returns>
76          #region Weekly
77          public static decimal LoanWeeklyPayment(Loan myLoan)
78          {
79              return Math.Round(LoanAnnualPayment(myLoan) / 52, 2);
80          }
81          #endregion
82
83          //-- Total loan payment
84
85          /// <summary>
86          /// Total payment
87          /// </summary>
88          /// <param name="myLoan"></param>
89          /// <returns></returns>
90          #region Total payment
91          public static decimal LoanTotalPayment(Loan myLoan)
92          {
93              return Math.Round(LoanAnnualPayment(myLoan) *                    ↵
                  myLoan.LoanTermYears, 2);
94          }
95          #endregion
96
97
98
99      }
100 }
101
```