```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Crash
{
   class theCrash
   {

      // Properties ----------------------------------------------------------
      /// <summary>
      /// Speed [mph]
      /// </summary>
      public double Speed { get; set; }
      /// <summary>
      /// Distance [ft]
      /// </summary>
      public double Distance { get; set; }

      /// <summary>
      /// Input value [0.0 - 1.0]
      /// </summary>
      public double Friction
      {
         get
         {
            return this._friction;
         }
         set
         {
            if (value < 0.0 || value > 1.0)
               throw new Exception("What are you, some kind of idiot?\n\n" +
                  "Friction input value out of range\n" +
               "It needs to be between 0.0 - 1.0!");
            else
               _friction = value;
         }
      }//end Friction get set

      /// <summary>
      /// Velocity [ft/s]
      /// </summary>
      public double Velocity { get; set; }

      /// <summary>
      /// Time [s]
      /// </summary>
      public double Time { get; set; }

      /// <summary>
      /// Radius [ft]
      /// </summary>
      public double Radius { get; set; }

      /// <summary>
      /// Chord of circle [ft}
      /// </summary>
```

```
public double Chord { get; set; }

/// <summary>
/// Middle Ordinate of circle [ft]
/// </summary>
public double MiddleOrdinate { get; set; }

// Fields --------------------------------------------------
public const double C = 30.0;
private double _friction;

// Methods -------------------------------------------------

/// <summary>
/// Calculates speed based on distance and friction
/// </summary>
/// <param name="d">Distance</param>
/// <param name="f">Friction</param>
/// <returns>Speed</returns>
public double calcSpeed1(double d, double f)
{
   Distance = d;
   Friction = f;

   return Math.Sqrt(d * f * C);
}

/// <summary>
/// Calculates the speed based on radius and friction
/// </summary>
/// <param name="r">Radius</param>
/// <param name="f">Friction</param>
/// <returns>Speed</returns>
public double calcSpeed2(double r, double f)
{
   Radius = r;
   Friction = f;

   return 3.86 * Math.Sqrt(Radius * Friction);
}

/// <summary>
/// Calculates time based on distance and velocity
/// </summary>
/// <param name="d">Distance</param>
/// <param name="v">Velocity</param>
/// <returns>Time</returns>
public double calcTime(double d, double v)
{
   Distance = d;
   Velocity = v;

   return Distance / Velocity;
}

/// <summary>
/// Calculates distance with speed and friction as inputs
/// </summary>
```

```
        /// <param name="s">Speed</param>
        /// <param name="f">Friction</param>
        /// <returns>Distance</returns>
        public double calcDistance1(double s, double f)
        {
           Speed = s;
           Friction = f;

           return (Speed * Speed) / (C * Friction);
        }

        /// <summary>
        /// Calculates distance with friction and time as inputs
        /// </summary>
        /// <param name="f">Friction</param>
        /// <param name="t">Time</param>
        /// <returns>Distance</returns>
        public double calcDistance2(double t,double f)
        {
           Friction = f;
           Time = t;

           return 16.1 * Friction * Time * Time;
        }

        /// <summary>
        /// Converts speed to velocity
        /// </summary>
        /// <param name="s">Speed</param>
        /// <returns>Velocity</returns>
        public double calcVelocity(double s)
        {
           Speed = s;

           return Speed * 1.466;
        }

        /// <summary>
        /// Calculates the radiusu given chord and middle ordinate of circle
        /// </summary>
        /// <param name="c">Chord</param>
        /// <param name="m">Middle Ordinate</param>
        /// <returns>Radius</returns>
        public double calcRadius(double c, double m)
        {
           Chord = c;
           MiddleOrdinate = m;

           return ((Chord * Chord) / (8 * MiddleOrdinate)) + (MiddleOrdinate / 2);
        }



     }//end class theCrash
  }//end namespace
```