```
using System;
using System.Collections.Generic;
using System.Text;
using System.Security.Cryptography;

#region No Warranty
/*================================================================================
 * NO WARRANTY
 * Mediacrypt/IT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
 * INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OR MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THE SOFTWARE
 * AND THE ACCOMPANYING WRITTEN MATERIALS. NO LIABILITY FOR CONSEQUENTIAL DAMAGES.
 * IN NO EVENT SHALL Mediacrypt/IT OR ITS SUPPLIERS BE LIABLE
 * FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION,
 * DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION,
 * LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS)
 * ARISING OUT OF THE USE OF OR INABILITY TO USE THIS Mediacrypt/IT PRODUCT,
 * EVEN IF Mediacrypt/IT HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
 *================================================================================*/
#endregion No Warranty

namespace NXTBlocks.Blockchain.Demo05
{
    public class Block
    {
        public int Index { get; set; }
        public DateTime TimeStamp { get; set; }
        public string PreviousHash { get; set; }
        public string Hash { get; set; }
        public string Data { get; set; }
        public int Nonce { get; set; } = 0;

        public Block(DateTime timeStamp, string previousHash, string data)
        {
            Index = 0;
            TimeStamp = timeStamp;
            PreviousHash = previousHash;
            Data = data;
```

```
            Hash = CalculateHash();
        }

        public string CalculateHash()
        {
            string result = string.Empty;

            SHA256 sha256 = SHA256.Create();

            Encoding utf8 = Encoding.UTF8;

            string inputString = string.Empty;
            string outputString = string.Empty;

            try
            {
                inputString = $"{TimeStamp}-{PreviousHash ?? ""}-{Data}-{Nonce}";

                byte[] inputBytes = utf8.GetBytes(inputString);

                byte[] outputBytes = sha256.ComputeHash(inputBytes);

                result = Convert.ToBase64String(outputBytes);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Exception: \n" + ex.Message);
            }

            return result;
        }

        public void Mine(int difficulty)
        {
            var leadingZeros = new string('0', difficulty);
            while (this.Hash == null || this.Hash.Substring(0, difficulty) != leadingZeros)
            {
                this.Nonce++;
```

```
                this.Hash = this.CalculateHash();
            }
        }
    }
}
```