hashmap.cm

```
/*
    ========================================================================

    Copyright (c) 2012-2016 Seppo Laakko
    http://sourceforge.net/projects/cmajor/

    Distributed under the GNU General Public License, version 3 (GPLv3).
    (See accompanying LICENSE.txt or http://www.gnu.org/licenses/gpl.html
        )

    ========================================================================
    */

// Copyright (c) 1994
// Hewlett-Packard Company
// Copyright (c) 1996
// Silicon Graphics Computer Systems, Inc.
// Copyright (c) 2009 Alexander Stepanov and Paul McJones

using System;
using System.Concepts;

namespace System.Collections
{
    public class HashMap<K, T, H = Hasher<K>, C = EqualTo<K>> where K is
        Semiregular and T is Semiregular and HashFunction<H, K> and C is
        Relation and C.Domain is K
    {
        public typedef K KeyType;
        public typedef T MappedType;
        public typedef Pair<KeyType, MappedType> ValueType;
        public typedef H HashFun;
        public typedef C Compare;
        public typedef HashMap<KeyType, MappedType, HashFun, Compare>
            Self;
        public typedef Hashtable<KeyType, ValueType, SelectFirst<KeyType,
            MappedType>, HashFun, Compare> TableType;
        public typedef TableType.ConstIterator ConstIterator;
        public typedef TableType.Iterator Iterator;

        public nothrow Iterator Begin()
        {
            return table.Begin();
        }
        public nothrow ConstIterator Begin() const
        {
            return table.CBegin();
        }
        public nothrow ConstIterator CBegin() const
        {
            return table.CBegin();
```

```
        }
        public nothrow Iterator End()
        {
            return table.End();
        }
        public nothrow ConstIterator End() const
        {
            return table.CEnd();
        }
        public nothrow ConstIterator CEnd() const
        {
            return table.CEnd();
        }
        public nothrow inline int Count() const
        {
            return table.Count();
        }
        public nothrow inline bool IsEmpty() const
        {
            return table.IsEmpty();
        }
        public nothrow void Clear()
        {
            table.Clear();
        }
        public nothrow Iterator Find(const KeyType& key)
        {
            return table.Find(key);
        }
        public nothrow ConstIterator Find(const KeyType& key) const
        {
            return table.CFind(key);
        }
        public nothrow ConstIterator CFind(const KeyType& key) const
        {
            return table.CFind(key);
        }
        public MappedType& operator[](const KeyType& key)
        {
            Pair<Iterator, bool> ib = Insert(ValueType(key, MappedType())
                );
            Iterator i = ib.first;
            return i->second;
        }
        public Pair<Iterator, bool> Insert(const ValueType& value)
        {
            return table.Insert(value);
        }
        public nothrow void Remove(const KeyType& key)
        {
            table.Remove(key);
        }
        public nothrow void Remove(Iterator pos)
```

```
        {
            table.Remove(pos);
        }
        private TableType table;
    }

    public nothrow bool operator==<K, T, H, C>(const HashMap<K, T, H, C>&
        left, const HashMap<K, T, H, C>& right) where K is Semiregular
        and T is Semiregular and HashFunction<H, K> and C is Relation and
        C.Domain is K
    {
        if (left.Count() != right.Count()) return false;
        for (const Pair<K, T>& p : left)
        {
            HashMap<K, T, H, C>.ConstIterator i = right.Find(p.first);
            if (i == right.End()) return false;
            if (i->second != p.second) return false;
        }
        return true;
    }
}
```