

time.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  
  
namespace System  
{  
    public class TimeError: Exception  
    {  
        public TimeError(const string& operation, const string& reason):  
            base(operation + ": " + reason)  
        {  
        }  
    }  
  
    public class TimePoint  
    {  
        public nothrow TimePoint(): nanosecs(0)  
        {  
        }  
        public explicit nothrow TimePoint(long nanosecs_): nanosecs(  
            nanosecs_)  
        {  
        }  
        public nothrow inline long Rep() const  
        {  
            return nanosecs;  
        }  
        private long nanosecs;  
    }  
  
    public class Duration  
    {  
        public nothrow Duration(): nanosecs(0)  
        {  
        }  
    }  
}
```

```

public explicit nothrow Duration(long nanosecs_): nanosecs(
    nanosecs_)
{
}
public nothrow long Hours() const
{
    return nanosecs / (3600 * long(1000000000));
}
public nothrow long Minutes() const
{
    return nanosecs / (60 * long(1000000000));
}
public nothrow long Seconds() const
{
    return nanosecs / long(1000000000);
}
public nothrow long Milliseconds() const
{
    return nanosecs / long(1000000);
}
public nothrow long Microseconds() const
{
    return nanosecs / long(1000);
}
public nothrow long Nanoseconds() const
{
    return nanosecs;
}
public static nothrow Duration FromHours(long hours)
{
    return Duration(3600 * long(1000000000) * hours);
}
public static nothrow Duration FromMinutes(long minutes)
{
    return Duration(60 * long(1000000000) * minutes);
}
public static nothrow Duration FromSeconds(long seconds)
{
    return Duration(long(1000000000) * seconds);
}
public static nothrow Duration FromMilliseconds(long milliseconds
)
{
    return Duration(long(1000000) * milliseconds);
}
public static nothrow Duration FromMicroseconds(long microseconds
)
{
    return Duration(long(1000) * microseconds);
}
public static nothrow Duration FromNanoseconds(long nanoseconds)
{
    return Duration(nanoseconds);
}

```

```

    }
    public nothrow inline long Rep() const
    {
        return nanosecs;
    }
    private long nanosecs;
}

public nothrow inline bool operator==(Duration left , Duration right)
{
    return left.Rep() == right.Rep();
}

public nothrow inline bool operator<(Duration left , Duration right)
{
    return left.Rep() < right.Rep();
}

public nothrow inline bool operator==(TimePoint left , TimePoint right)
{
    return left.Rep() == right.Rep();
}

public nothrow inline bool operator<(TimePoint left , TimePoint right)
{
    return left.Rep() < right.Rep();
}

public nothrow inline Duration operator+(Duration left , Duration
    right)
{
    return Duration(left.Rep() + right.Rep());
}

public nothrow inline Duration operator-(Duration left , Duration
    right)
{
    return Duration(left.Rep() - right.Rep());
}

public nothrow inline Duration operator*(Duration left , Duration
    right)
{
    return Duration(left.Rep() * right.Rep());
}

public nothrow inline Duration operator/(Duration left , Duration
    right)
{
    return Duration(left.Rep() / right.Rep());
}

```

```

public nothrow inline Duration operator%(Duration left , Duration
    right)
{
    return left.Rep() % right.Rep();
}

public nothrow inline Duration operator-(TimePoint left , TimePoint
    right)
{
    long diff = left.Rep() - right.Rep();
    return Duration(diff);
}

public nothrow inline TimePoint operator+(TimePoint tp, Duration d)
{
    return TimePoint(tp.Rep() + d.Rep());
}

public nothrow inline TimePoint operator+(Duration d, TimePoint tp)
{
    return TimePoint(tp.Rep() + d.Rep());
}

public nothrow inline TimePoint operator-(TimePoint tp, Duration d)
{
    return TimePoint(tp.Rep() - d.Rep());
}

public TimePoint Now()
{
    long secs = 0;
    int nanosecs = 0;
    int result = time_nanosecs(secs , nanosecs);
    if (result != 0)
    {
        string reason = strerror(result);
        throw TimeError("could not get current time", reason);
    }
    long now = 1000000000 * secs + nanosecs;
    return TimePoint(now);
}
}

```