Selected files

6 printable files

```
Week_11\11.1\Clock\Clock.cs
Week_11\11.1\Clock\Counter.cs
Week_11\11.1\Clock\Program.cs
Week_11\11.1\ClockPython\clock.py
Week_11\11.1\ClockPython\counter.py
Week_11\11.1\ClockPython\main.py
```

Week_11\11.1\Clock\Clock.cs

```csharp
using System;

namespace ClockProgram
{
    public class Clock
    {
        private Counter _seconds;
        private Counter _minutes;
        private Counter _hours;

        public Clock()
        {
            _seconds = new Counter("Seconds");
            _minutes = new Counter("Minutes");
            _hours = new Counter("Hours");
        }

        public void Tick()
        {
            _seconds.Increment();
            if (_seconds.Ticks != 60) return;

            _seconds.Reset();
            _minutes.Increment();
            if (_minutes.Ticks != 60) return;

            _minutes.Reset();
            _hours.Increment();
            if (_hours.Ticks != 12) return;

            _hours.Reset();
        }

        public void Reset()
        {
            _seconds.Reset();
            _minutes.Reset();
            _hours.Reset();
        }

        public string Time
        {
            get
            {
                return $"{_hours.Ticks:D2}:{_minutes.Ticks:D2}:{_seconds.Ticks:D2}";
            }
        }
    }
}
```

Week_11\11.1\Clock\Counter.cs

```
public class Counter
{
    private int _count;
    private string _name;

    public Counter(string name)
    {
        _name = name;
        _count = 0;
    }

    public void Increment()
    {
        _count++;
    }

    public void Reset()
    {
        _count = 0;
    }

    public string Name
    {
        get
        {
            return _name;
        }

        set
        {
            _name = value;
        }
    }

    public int Ticks
    {
        get
        {
            return _count;
        }
    }
}
```

Week_11\11.1\Clock\Program.cs

```
using System;
using System.Diagnostics;

namespace ClockProgram
{
    public static class Program
    {
        public static void RunClock(int seconds)
        {
            Clock clock = new Clock();

            for (int i = 0; i < seconds; i++)
            {
                clock.Tick();
            }
        }

        public static void Main()
        {
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();

            RunClock(104844794);

            stopwatch.Stop();

            TimeSpan ts = stopwatch.Elapsed;
            Console.WriteLine("C# Clock - Minh An Nguyen - 104844794\n");
            Console.WriteLine($"Time elapsed: {ts.Microseconds:n0} microseconds");

            //Get the current process
            Process proc = Process.GetCurrentProcess();

            //Display the total physical memory size allocated for the current process
            Console.WriteLine($"Current physical memory usage: {proc.WorkingSet64:n0} bytes");

            // Display peak memory statistics for the process.
            Console.WriteLine($"Peak physical memory usage {proc.PeakWorkingSet64:n0} bytes");
        }
    }
}
```

Week_11\11.1\ClockPython\clock.py

```python
from counter import Counter


class Clock:
    def __init__(self):
        self._seconds = Counter("seconds")
        self._minutes = Counter("minutes")
        self._hours = Counter("hours")

    def tick(self):
        self._seconds.increment()
        if self._seconds.ticks != 60:
            return

        self._seconds.reset()
        self._minutes.increment()
        if self._minutes.ticks != 60:
            return

        self._minutes.reset()
        self._hours.increment()
        if self._hours.ticks != 24:
            return

        self._hours.reset()

    def reset(self):
        self._seconds.reset()
        self._minutes.reset()
        self._hours.reset()

    @property
    def time(self):
        return (
            f"{self._hours.ticks:02}:{self._minutes.ticks:02}:{self._seconds.ticks:02}"
        )

```

Week_11\11.1\ClockPython\counter.py

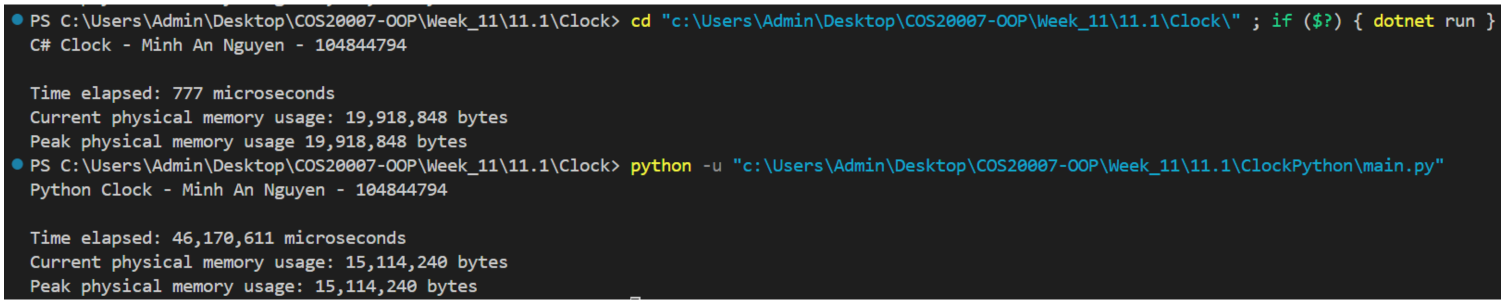
```python
class Counter:
    def __init__(self, name):
        self._name = name
        self._count = 0

    def increment(self):
        self._count += 1

    def reset(self):
        self._count = 0

    @property
    def name(self):
        return self._name

    @name.setter
    def name(self, value):
        self._name = value

    @property
    def ticks(self):
        return self._count

```

Week_11\11.1\ClockPython\main.py

```python
import time
import os
import psutil
from clock import Clock


def run_clock(seconds):
    clock = Clock()
    for _ in range(seconds):
        clock.tick()


def main():
    # Start the stopwatch
    start_time = time.time()

    # Run the clock for 104844794 ticks
    run_clock(104844794)

    # Stop the stopwatch
    end_time = time.time()

    print("Python Clock - Minh An Nguyen - 104844794\n")
    # Calculate elapsed time in microseconds
    elapsed_time = (end_time - start_time) * 1_000_000
    print(f"Time elapsed: {elapsed_time:,.0f} microseconds")

    # Get the current process
    process = psutil.Process(os.getpid())

    # Display the total physical memory size allocated for the current process
    print(f"Current physical memory usage: {process.memory_info().rss:,} bytes")

    # Display peak memory statistics for the process
    print(f"Peak physical memory usage: {process.memory_info().peak_wset:,} bytes")

```

```python

if __name__ == "__main__":
    main()

```

Screenshot of 2 programs (C# and Python) running:



```
PS C:\Users\Admin\Desktop\COS20007-OOP\Week_11\11.1\Clock> cd "c:\Users\Admin\Desktop\COS20007-OOP\Week_11\11.1\Clock\" ; if ($?) { dotnet run }
C# Clock - Minh An Nguyen - 104844794

Time elapsed: 777 microseconds
Current physical memory usage: 19,918,848 bytes
Peak physical memory usage 19,918,848 bytes
PS C:\Users\Admin\Desktop\COS20007-OOP\Week_11\11.1\Clock> python -u "c:\Users\Admin\Desktop\COS20007-OOP\Week_11\11.1\ClockPython\main.py"
Python Clock - Minh An Nguyen - 104844794

Time elapsed: 46,170,611 microseconds
Current physical memory usage: 15,114,240 bytes
Peak physical memory usage: 15,114,240 bytes
```