School of Science, Computing and Engineering Technologies

Object Oriented Programming

Task 12.1: Clock with Counters in Another Language

Overview

When learning a new language it is always best to create a small program that you are familiar with. In the last task of COS20007, you will recreate the Clock class from the previous task in a new programming language.

Purpose:	See that the principles you have learnt apply equally to other object oriented programming languages.
Task (2 marks)	Implement your Clock class and supporting Counter class in a different object oriented programming language.

Instructions

Review your design for the clock from the task 2.1 (Week 2) - Counter class with C#, and use this to implement following tasks.

1. The Clock application in C# (**see provided partial coding solution**) and
2. The Clock application in another different OO programming language. You can use any OO programming language except for C#, such as C++, Java, or Python.
3. Utilise existing libraries in C# and your chosen language to benchmark the memory usage and execution time of the above two applications **(see the hint next page)**.

Clock class description.

You can reuse the Counter class to define the behavior of a 24-hour clock. Start by creating a new *Clock* class that maintains three *Counter* objects (as instance variables). When the clock *ticks*, the value of one, two, or all three *Counter* objects changes. The collaboration between the Counter objects drives these changes and, when defined correctly, gives your *Clock* class the behavior of a 24-hour clock.

- The clock must keep track of an hours, minutes, and seconds value, 12-hour clock or 24-hour clock. **If your last student ID is smaller than or equal 5 takes 12-hour format. Otherwise, take the 24-hour clock format.**
- The clock can be told to "*tick*" to advance its overall value by one second.
- You need to be able to read the clock's time as a string in the format "hh:mm:ss" in 24-hour time.
- You should be able to reset the clock to 00:00:00.
- The clock does not have to run in real time. It should just *emulate* the behavior of a clock (e.g., if it has the value 00:00:59 and the clock is told to "*tick*", then its new value becomes 00:01:00). **This must comply with the format of 12 or 24-hour format.**

- **Hints. Below is the hint how to measure memory usage and execution time with C#, Java, and Python.**
- We do not require unit test implementation for this task. You could work out how to do unit testing in the other language at a later stage at your own study.

With C#, you can add the following code at the end of your Program.cs to measure the physical memory usage of your current process.

```
//Get the current process
System.Diagnostics.Process proc =
System.Diagnostics.Process.GetCurrentProcess();
Console.WriteLine("Current process: {0}", proc.ToString());
//Display the total physical memory size allocated for the current
process Console.WriteLine("Physical memory usage: {0} bytes",
proc.WorkingSet64);
// Display peak memory statistics for the process.
Console.WriteLine("Peak physical memory usage {0} bytes",
proc.PeakWorkingSet64);
```

- Reference for the memory usage with C#: https://learn.microsoft.com/en-us/dotnet/api/system.diagnostics.process.totalprocessortime?view=net-8.0

To measure the elapsed execution time, you can use the Stopwatch class in .Net framework. Please refer to the following link for an example.
https://learn.microsoft.com/en-us/dotnet/api/system.diagnostics.stopwatch?view=net-6.0

With Java, you can use the Runtime class to measure the memory usage. The following code will help you measure the used memory in bytes.

```
// Get the total memory available to the JVM in bytes
long totalMemory = runtime.totalMemory();
// Get the free memory available to the JVM in bytes
long freeMemory = runtime.freeMemory();
// Calculate the used memory in bytes
long usedMemory = totalMemory - freeMemory;
```

- Reference for the memory usage with Java: https://www.geeksforgeeks.org/java-runtime-totalmemory-method/
- Reference for the Stopwatch class to measure the elapsed time in Java. https://introcs.cs.princeton.edu/java/stdlib/Stopwatch.java.html

With Python, you can use either Tracemalloc or Psutil library to monitor the memory usage. Reference is provided below.

https://www.geeksforgeeks.org/monitoring-memory-usage-of-a-running-python-program/