School of Science, Computing and Engineering Technologies

Object Oriented Programming

Pass Task 11.1: Clock in Another Language

Overview

When learning a new language it is always best to create a small program that you are familiar with. In this task 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:	Implement your Clock class and supporting Counter class in a different object oriented programming language.
Time:	End of Week 12, Friday 1 August 2025, 23:59:00 (Firmed)

Submission Details

All students have access to the Adobe Acrobat tools. Please print your solution to PDF and combine it with the screenshots taken for this task.

- Program source code
- Screenshot of program execution
- Screenshots comparing the memory usage and execution time between your clock application implemented in C# and your newly selected programming language.

Instructions

Review your design for the clock from the previous pass task, and use this to implement the Clock in a different OO programming language. You can use any OO programming language except for C#, and you must recreate the design from the previous task.

Hints. See next page for references regarding how to measure memory usage and execution time with C#, Java, and Python.

Note: We do not require unit test implementation for this task. You could work out how to do unit test-ing in the other language at a later stage at your own study.

Assessment Criteria

Make sure that your task has the following in your submission:

- The program is implemented correctly based on the original clock design.
- Code must mostly follow the coding conventions of your chosen language.
- The "Universal Task Requirements" (see Canvas) have been met.

Hints.

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/