



DEPARTMENT OF
SOFTWARE TECHNOLOGY

CSOPESY Seatwork

Instructor: Neil Patrick Del Gallego, Ph.D.

Use Calibri Font Size 11 for texts.

NAME:	Daniel Gavrie Clemente
SECTION:	S18
DATE OF SEATWORK	May 17, 2025

1.

Description of Attributes:

- pid: Unique process ID
- name: Optional process name
- instructions: A list of instruction lines

- `currentInstructionIndex`: Tracks the line being executed

Key Functions:

- Constructor initializes the process with an ID and its instructions.
- `step()` advances the process to the next instruction.
- `isFinished()` checks if the process completed all instructions
- `printCurrentInstruction()` prints the instruction currently being executed.

2.

```

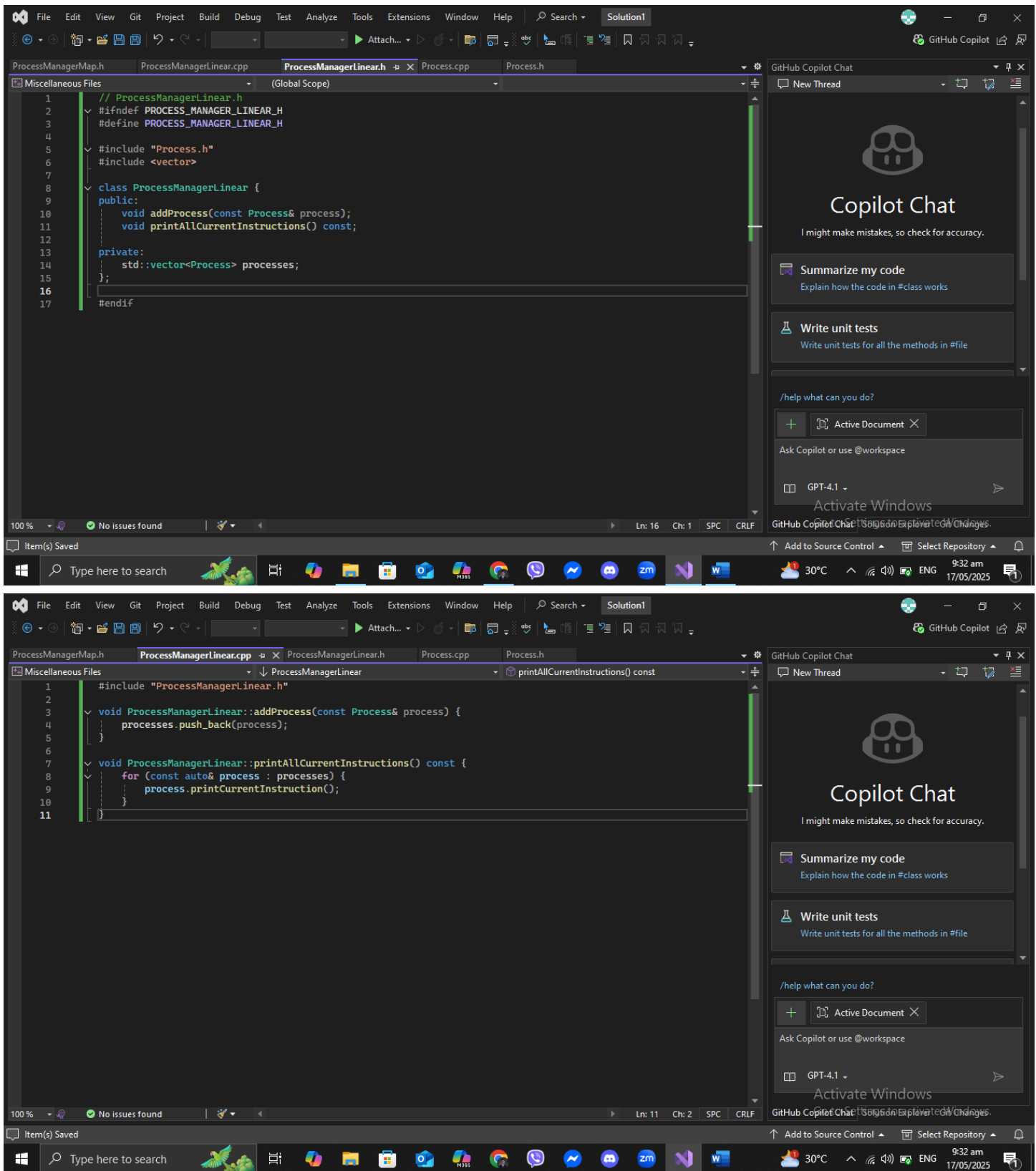
1  #include "Process.h"
2  #include <iostream>
3
4  Process::Process(int pid, const std::vector<std::string>& instructions)
5      : pid(pid), instructions(instructions), currentInstructionIndex(0) {
6  }
7
8  int Process::getPID() const {
9      return pid;
10 }
11
12 int Process::getCurrentInstructionIndex() const {
13     return currentInstructionIndex;
14 }
15
16 void Process::step() {
17     if (!isFinished()) {
18         currentInstructionIndex++;
19     }
20 }
21
22 bool Process::isFinished() const {
23     return currentInstructionIndex >= instructions.size();
24 }
25
26 void Process::printCurrentInstruction() const {
27     if (!isFinished()) {
28         std::cout << "PID " << pid << " executing line " << currentInstructionIndex
29             << " : " << instructions[currentInstructionIndex] << std::endl;
30     }
31     else {
32         std::cout << "PID " << pid << " has finished execution." << std::endl;
33     }
34 }

```

Explanation:

- **step()**: Moves the process forward by incrementing the instruction index, simulating execution of the next line.
- **isFinished()**: Returns true if the process has executed all instructions.
- **printCurrentInstruction()**: Prints the current instruction being executed, along with the process ID and line number. If all instructions have been executed, it prints a finished message.

3.

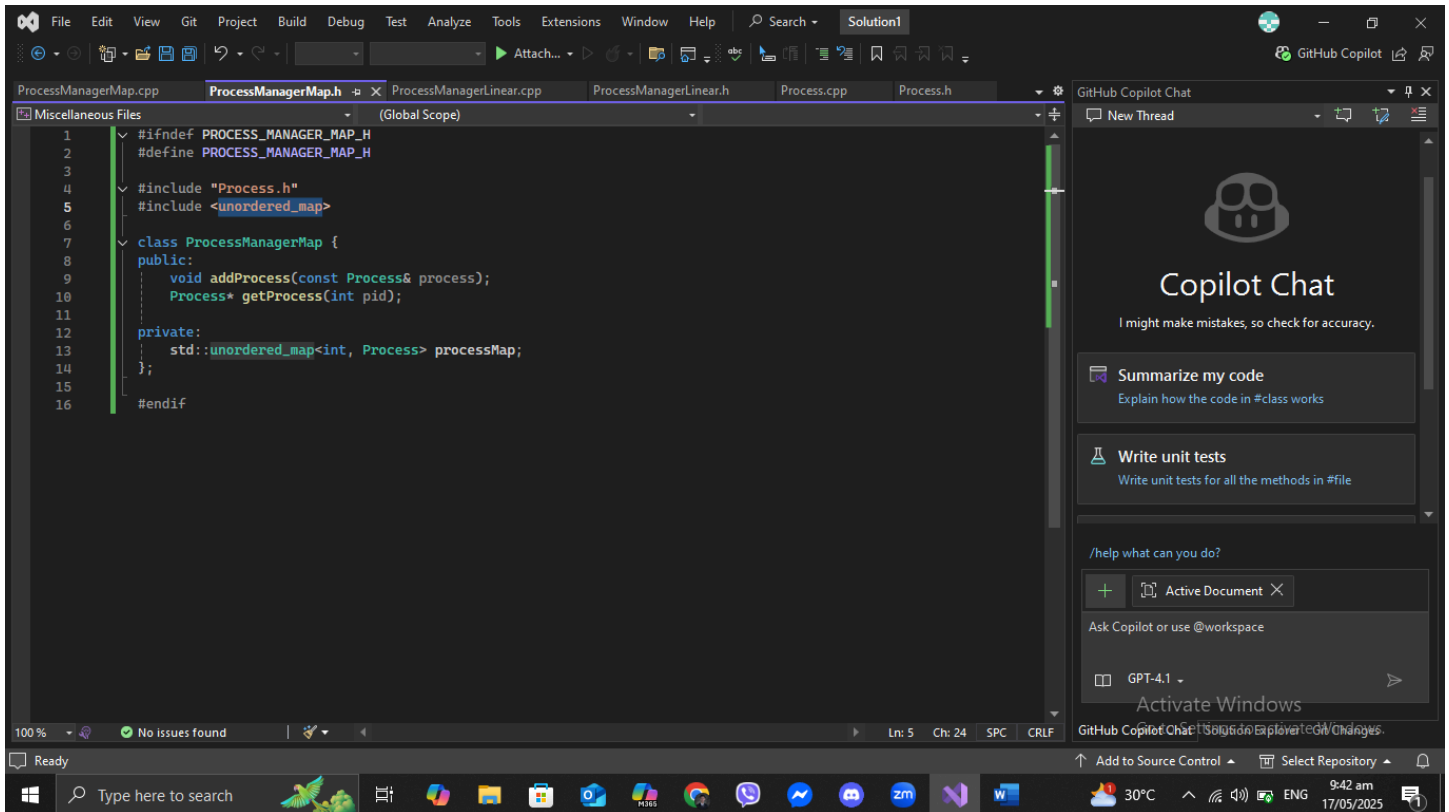


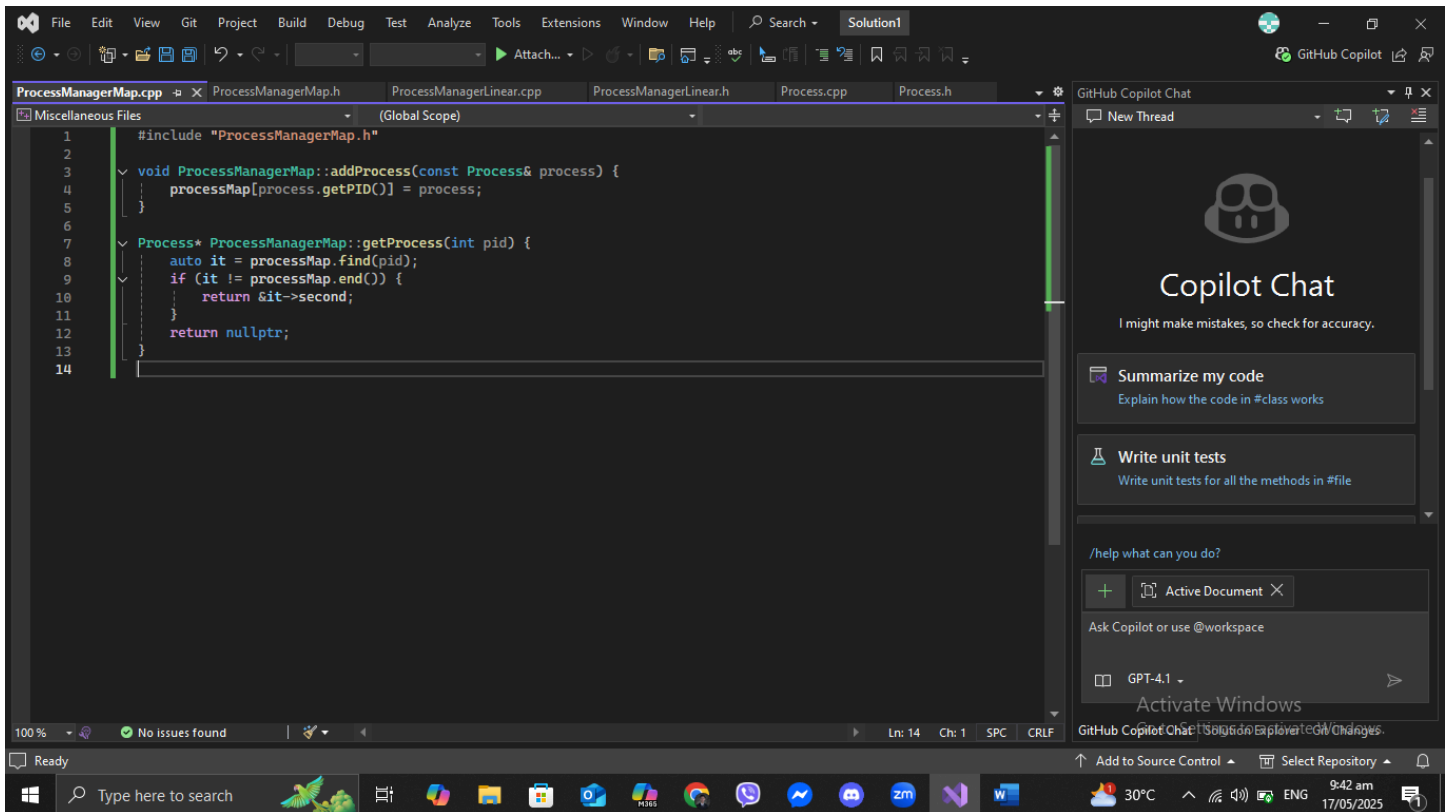
Explanation:

- **Purpose:** Manage multiple processes and allow iterating over them sequentially.
- **Data Structure:** Uses a `std::vector<Process>` to store processes.

- **addProcess():** Adds a new process to the vector.
- **printAllCurrentInstructions():** Iterates over all processes and prints each process's current instruction.
- **Traversal:** Since it uses a vector, iterating through all processes is done in linear time $O(N)$, where N is the number of processes.

4.





Explanation:

- **Purpose:** Store processes in a hash map for fast retrieval by pid.
- **Data Structure:** `std::unordered_map<int, Process>` maps pid (key) to Process object (value).
- **addProcess():** Inserts or updates the process associated with the given PID.
- **getProcess():** Retrieves a pointer to the process with the given PID in **constant time $O(1)$** . Returns `nullptr` if no such process exists.
- **Benefit:** This is more efficient when you want to directly access a process without iterating over the entire list.

References:

<https://en.cppreference.com/w/>
<https://www.w3schools.com/cpp/default.asp>
<https://cplusplus.com/reference/>