BLG252 ASSIGNMENT 2 REPORT

Mehmet Ali Balıkçı

150200059

April 2023

Contents

Note: All of the existing association types wrote and marked with a yellow marker.

1- Introduction

This report analyzes a code for designing a computer, which includes classes and objects for, GPU, CPU, ALU, CUDA, and Computer. It demonstrates how the messages sent to these objects enable the application to run effectively. For further details, please refer to the code provided in the assignment.

2- Method

To analyze the code, the report will utilize the Debug feature of Visual Studio Code and repeatedly use the step info feature to ensure that no level of abstraction is overlooked.

3- Implementation

As a first, the methods and attributes of five classes existing on the code respectively are:

GPU Class:

- Attributes:
  - m_cuda: This is an object of CUDA class. It provides this class (owner) to build a composition relationship with CUDA (part) class.
- Methods:
  - GPU() (constructor): It takes an argument that keeps the number of cuda and assigns it to the m_cuda attribute.
  - execute(): It takes a string argument that indicates which operation will be executed and returns a call for the related method of the CUDA object.

CPU Class:

- Attributes:
  - m_alu: This is an object of the ALU class. It provides this class (owner) to build a composition relationship with the ALU class.
- Methods:

- CPU() (constructor) : It takes an argument which keeps the number of alu and assign it to m_alu attribute.
- execute(): It takes a string argument that indicates which operation will be executed. Two variables are defined in this method. The values of these variables will be entered by users. Finally, this method returns a call for the related method of the ALU object according to entered operation name and passed two variables defined here.

ALU Class:

- Attributes:
  - numPerCores: It keeps the number of ALU per core.
- Methods:
  - ALU() (constructor): It takes a parameter named cores and assigns it to the numPerCores variable.
  - add(), subtract(), sultiply() methods: These take two variables as parameters and do the operations that their name indicates, and return the result.
  - getNumPerCores (): It returns the value of numPerCores.
  - setNumPerCores (): It takes a value as a parameter and assigns it the variable numPerCores.

CUDA Class:

- Attributes:
  - numCores: It keeps the number of Cuda cores.
- Methods:
  - CUDA() (constructor): It takes a parameter named cores and assigns it to the numCores variable.
  - render(): It returns the string “Video is rendered”.
  - trainmodel(): It returns the string “AI model is trained”.
  - getNumCores(): It returns the value of numCores.
  - setNumCores(): It takes a value as a parameter and assigns it the variable numCores.

Computer Class:

- Attributes:
  - cpu_counter: It keeps the number of cpu in a computer.

- gpu_counter: It keeps the number of gpu in a computer.
- attachedCPU: It is a pointer to the object CPU. It consists of an aggregation relationship between the computer class (whole) and cpu class (part).
- attachedGPU: It is a pointer to the object GPU. It consists of an aggregation relationship between the computer class (whole) and cpu class (part).

- Methods:
  - Computer() (constructor): It prints the message “Computer is ready”.
  - operator+(): This method overloads the operator + to add a cpu to the computer. It takes an object in terms of CPU sent with the way of call by reference.
  - operator+(): This method overloads the operator + to add a gpu to the computer. It takes an object in terms of GPU sent with the way of call by reference.
  - execute(): It takes a string variable keeping the name of the operation that will be executed. According to the operation name, this method performs this operation the execution methods of GPU and CPU through the object pointers that it has.

4- Explanation for parameter visibility:

The Computer class has two pointers to the object of the CPU and GPU, and uses a method of these objects. But, these methods are not sent a reference to these objects. Therefore, there is no parameter visibility.

5- UML Diagram