/\*

Student: Marina Fiamskaia

Student ID: 025-698-051

email: mfiamska@myseneca.ca

Date: June 14, 2022

Subject: OOP345 NAA

Project: Workshop 4 part 2 reflect

\*\*\*\*I have done all the coding by myself and only copied the code that my professor provided to complete my workshops and assignments.\*\*\*\*

\*/

In this workshop four I have learned how to design and code composition and aggregation class relationships; how to use member functions of the string class to parse a string into token based on simple rules; and how to design and code a class that manages a dynamically allocated array of pointers to objects.

For both classes Child and ConfrimOrder I have added 6 more public members: default no argument constructor, copy constructor, copy assignment operator, move constructor, move assignment operator and destructor. I have done so because both classes Child and ConfirmOrder are classes that have resources outside the class, that is a pointer to the array of pointers of the Toy objects. Therefore, both classes need to follow the rule of five.

I had two main challenges completing this workshop. My first challenge was coming up with logic for trimming of the read string using std::string functions. After reading the cpprefence.com on the std::functions I have decided to copy original string into working string using std::string.substr(beginning, end). To find the end of the string to be retrieved I have used std::string.find(delimeter) function. That allowed me to retrieve from the working string the necessary strings to the delimiter. In order to retrieve next string I have used std::string.erase(beginning, end)function. In the same manner, I have used std::string.find(delimeter) to locate the last character to be erased. After discovering std::stoi and std::stod functions that convert string into integers and doubles respectively and discard any whitespace characters in the string, the only issue was to discard beginning and ending white space characters in the retrieved name string. It took quite some time to figure out logic and which functions to use because std::string.find no longer will be suitable since name string consist of two words and have one space between them. I have decide to use std::string.erase(0, first not “ “ character), and std::string.erase(last not “ “ character + 1). In order to find first not “ “ character I have used std::string.find\_first\_not\_of(' ') and in order to find find the last character before the end of string whitespace I have used std::string.find\_last\_not\_of(' ').

I find that std::string have a lot of useful functions and they have a lot of overloads to them. Surprisingly, it doesn’t have a trim function designed specifically for trimming out the “garbage”.

My next, the bigger challenge was to implement the composition design class relationship specifically using double pointer. The modules Child and Toy has a composition relationship. Child is a composing class and contains the pointer to the array of pointers to the Toy objects. Essentially, the Child object has an array of pointers that point to the Toy objects. Child completely owns the array of pointers and the objects it is pointing to. It controls the lifetime of the Toy objects contained within. Child is responsible for maintaining and managing the array pointers and the Toy objects. That is the Child should have a destructor that will not only delete resource of a pointer to the array, but also resources of an array of pointers, the objects itself. Child is responsible for destroying the component Toy objects before its own destruction. Also, when resizing the array, Child is responsible for pointer to the array, the array of pointers and the objects it is pointing to. When applying the rule of five, Child should manage the Toy objects contained within array of pointers.

On the other hand, ConfirmOrder class have an aggregation relationship with class Toy. ConfirmOrder has a pointer to the array of pointers to the objects Toy. ConfirmOrder owns the pointer to the array but not the objects itself. Toy objects exist independently of the array of pointers. The ConfirmOrder does not manage the creation or destruction of the objects in the pointer array. ConfirmOrder is not responsible for destroying Toy objects before its own destruction. ConfirmOrder class is complete despite the existence of Toy objects that is pointed to by the array of pointers. When resizing the array of pointers, ConfirmOrder is not responsible for the objects it is pointing to, only for the pointer to the array and array of pointers itself. When applying the rule of five, the ConfirmOrder is not responsible for the objects contained within pointer array and should not manage the Toy objects.

My personal issue was the syntax for managing the array of pointers and the Toy objects contained in the Child class. After I have showed my code to Khoi Vu and Jenny Lee on June 14, I was made aware that even though my intentions are correct, my syntax is not for the line 34 in Child.cpp file. It was thanks to them that I was able to continue with my workshop and essentially complete it.

Using the pointer to the array of pointers seems to be adding another layer during management and maintenance. It does not look too complicated, but something that needs to be visualized and practiced more often to be mastered.