We watched three videos on applying object-oriented programming to C++ using Visual Studio 2015. I looked through the first four videos but did not see anything I had not already covered in my previous C++ class so I decided to focus mainly on the last three videos as that is where I am weakest. In the first two videos, they developed a program to calculate the BMI of patients. We learned about classes, which is a data type that contains variables and functions, and how to work with them.

In the first video, we created the main source file (Main.cpp), a separate cpp file (BMI.cpp) to store functions, and a header file (BMI.h) to create the class and declare functions inside it.

Inside the BMI.h file, we created the code:

#ifndef BMI\_h

#define BMI\_h

#endif

#ifndef stands for “if not defined” and checks whether BMI\_h has already been defined or not. If it has not been defined, then it is then defined by the #define line. Afterward, the class is created by the class BMI {} line. This creates a class and calls it BMI. Inside of the BMI class, we have public: and private: variables and functions (I am not sure if a function can be private). Under public, we declare the default constructor BMI();, the overload constructor BMI(string, int, double); and, under private, the member variables, string newName, int newHeight, and double newWeight. The class is then ended with }; (be sure to remember the unconventional ; here as it will throw errors otherwise).

Inside the BMI.cpp file, we defined our two constructor functions. The line #include "BMI.h" tells the file that it may read and access the BMI.h header file. The default constructor BMI(); is used to create the object and assign the default values of 0 to the variables within the object. In the overload constructor BMI(string, int, double); we are taking the user input values of name, height, and weight, and assigning them to the object variables newName, newHeight, and newWeight. We must remember that in order to define a function declared in the header file, we must use the :: scope resolution operator to “attach” it to the class, for instance BMI::BMI().

Inside the main.cpp file we are also including the #include "BMI.h" line so that it may read the header file. Otherwise, we are declaring the variables name, height, weight, asking the user to enter values for them and then taking their input and assigning them to the variables. Afterwards, we are using the BMI Student\_1(name, height, weight); to create the object Student\_1 and assign the variables to it.

For the second video, we declared and defined the destructor, accessor, and mutator functions, as well as the BMI calculation function. We showed different ways to set the variables in an object as well as how to work with separate objects.

Inside the BMI.h file, declared the Destructor with the line ~BMI();. We then declared the accessor and mutator functions, for example string getName() const; and void setName(string); respectively. We learned that const is to be used at the end of the function declaration if the function is not going to be changing any variables it calls.

Inside the BMI.cpp file, we defined what was previously declared in the BMI.h file. The get functions were just a return of their corresponding new\* variables, such as newName. The set functions simply set the new variables to be equal to the user input variables, such as newName = name;. Lastly, we calculate the BMI by defining the BMI::calculateBMI() functions, which was the weight multiplied by 703 divided by the height squared.

Afterwards, we added to the Main.cpp file. We created another object which was Student\_2 and then called the set functions to show that we can use the mutator functions in order to set the variables in a different way than we did in the first video by just calling the BMI class function.

The third video was the most difficult for me and I hope by writing this it will help me to understand and work through it. Our goal here was to create a vector of objects so that we may dynamically make many new objects without having to write code for each one. We created a class, functions, and header file the same as the previous two videos.

Once we started working on the Source.cpp file was where it started to get interesting. There we used #include <vector> and vector<Student> myClass; in order to create a vector called myClass. Then we created the functions void fillVector(vector<Student>& newMyClass); and void printVector(const vector<Student>& newMyClass);. If I understand correctly, we are creating a variable newMyClass as a reference for the student vector that we have created (myClass). Inside the fillVector function we are asking for the size of the class and using a for loop to ask for a student name and grade for each member of the class. Then, we are creating a new object, adding those variables to it, and using newMyClass.push\_back(newStudent); we are adding that object (newStudent) onto the end of the vector. Lastly, in printVector, we are using a for loop for the size of the class and pulling the Student Name and Grade using the accessor functions in the form of newMyClass[i].getName(). This is using the vector reference along with the i value to reference the location on the vector of the object we are performing getName on.

That last one took a little while to wrap my mind around!