Lab 8: OOP continued

- Realize the concept of polymorphism.
- Recognize how to declare a virtual function.
- Observe how virtual functions relate to inheritance.
- Learn how to implement pure virtual functions.
- Understand the concept of abstract classes and interfaces

A virtual function is declared in a ***base class*** of a program and can then be redefined in each derived class. The declaration in the base class acts as a kind of template which can be enhanced by each derived class.

- In the base class, you must begin the function declaration with the keyword ***virtual***.
- The keyword ***virtual*** is ***not*** used in the derived functions.
- The number and type of parameters in the derived function **must** match the number and type in the initial declaration of the virtual function.

A virtual function that is declared but not defined in a base class is referred to as a pure virtual function. When the base class uses a PURE virtual function, each descendent **must** override that function, or you will get a compile error. This makes sense, because the function has to be defined *somewhere*.

Here is the general syntax for declaring a pure virtual function:

```
virtual type function_name(parameter_list)=0;
```

If a class contains a pure virtual function, that class is called an abstract class.

Because there is no definition for the function, you cannot create an object of that class. You can however, declare pointers to it.

Think of an abstract class as a general class that lays the foundation for descendent classes that define their own methods. This is the heart of polymorphism - let each class define its own operation.

Submission:

- **Answer all questions in Tasks 1.1, 1.2 and 1.3**
- **Write your answers in AnswersLab8.txt, and include it in your zip file**
- **Submit the Zip file.**

Task1.1:

1. Create a class named base_conversion that has:
   - 2 private data members of type double (m_initialValue, m_convertedValue).
   - Public const getter methods: get_initial, get_converted
   - Public setter methods: set_initial, set_converted
   - A parameterized constructor that is passed a double value to initialize m_initialValue.
   - A virtual function convert_it() that prints "Base conversion by super class", multiplies m_initialValue by 2, and sets result to m_convertedValue.
2. Create a class named kilos_miles that extends base_conversion and also:
   - Overrides convert_it() to convert m_initialValue from kilometers to miles, sets result to m_convertedValue and prints "Conversion by kilos_miles class". Formula: miles = kilos * 0.6
3. Create a class named miles_kilos that extends base_conversion and also:
   - Overrides convert_it() to convert m_initialValue from miles to kilometers, sets result to m_convertedValue, and prints "Conversion by miles_kilos class". Formula: kilos = miles / 0.6
4. Create a class named yard_meter that extends base_conversion.
   - This class does not override convert_it() function.
   - Has a methods newConvert() that prints "Not able to convert yet"

Task1.2:

In main():

1. Create baseOBJ an instance object of base_conversion with initial value = 100.0.
2. Make baseOBJ invoke convert_it().
   Now add the following statements:
```
      cout << "Initial value is: " << baseOBJ.get_initial()<<endl;
      cout << "Converted value is: " << baseOBJ.get_converted()<<endl;
```
3. Create kmOBJ an instance object of kilos_miles speacifying an initial value of 100.0.
4. Make kmOBJ invoke convert_it().
   Now add the following statements:
```
   cout << "Distance in kilometers is: " << kmOBJ.get_initial()<<endl;
   cout << "Distance in miles is: " << kmOBJ.get_converted()<<endl;
```
   Which convert_it() function is invoked? Explain why?
5. Create mkOBJ an instance of miles_kilos class, specifying an initial value of 60.
6. Make mkOBJ invoke convert_it().
7. Print out the initial and converted values.
8. Create ymOBJ an instance of yard_meter class, specifying an initial value of 50.
9. Make ymOBJ invoke convert_it().
   Print out the initial and converted values.
   Which convert_it() function is invoked? Explain why?
10. Was it mendatory for derived class yards_meter to override inherited virtual function of base_conversion class? Explain?

Task1.3:

1. Comment out all instructions in main().
2. In base_conversion class, turn convert_it() into a **pure virtual** function.
3. Create baseOBJ an instance object of base_conversion with initial value = 100.0. Was this instruction successful? What is the cause of the error if any?
4. Comment out the previous instruction.
5. Create ymOBJ an instance of yard_meter class, specifying an initial value of 50. Was this instruction successful? What is the cause of the error if any?
6. Make the necessary changes to yard_meter class to fix this error.