Swinburne University of Technology Sarawak Campus

Faculty of Engineering, Computing and Science

LABORATORY COVER SHEET

Subject Code:	COS30008
Subject Title:	Data Structures and Patterns
Lab Number and Title:	1, Revisiting Visual Studio, C++ and Object-Oriented Programming
Lecturer:	Dr. Mark Tee

Summary

In this first lab session, you must quickly get up to speed with the following:

a) Install the latest Visual Studio Community Edition (at least 2019), build a C++ program and successfully run/debug it.

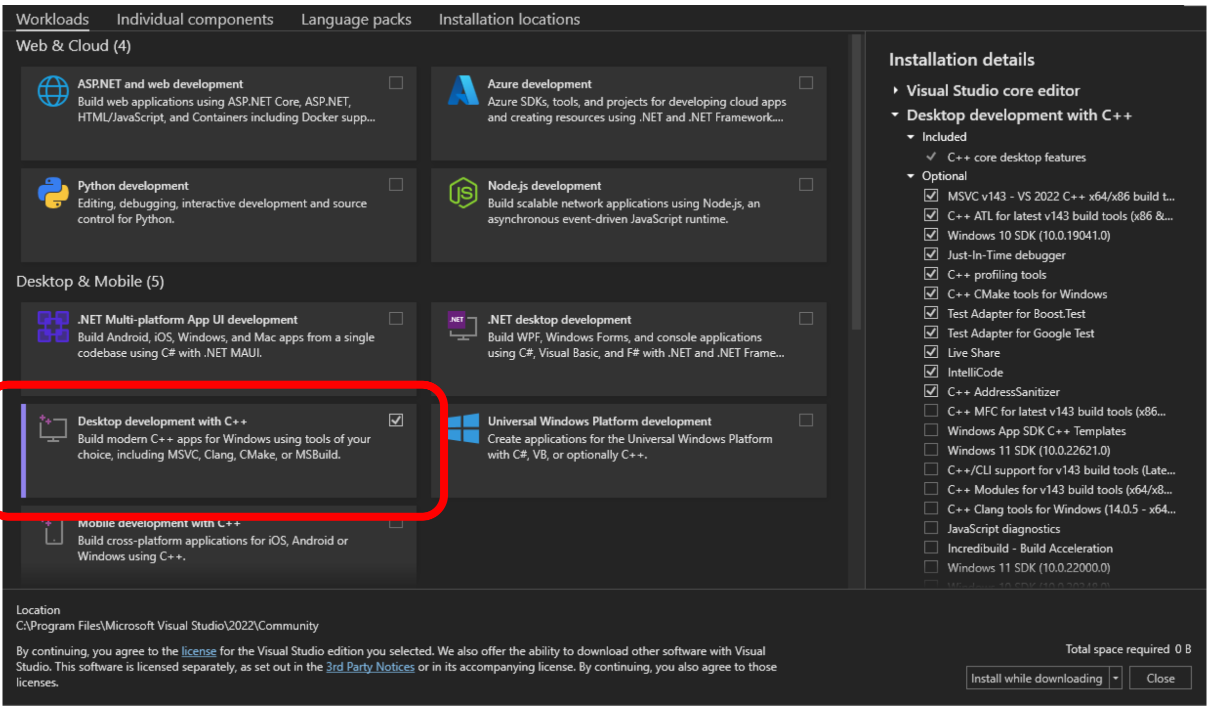
b) Revisit Object Oriented Programming in C++ (classes, objects and accessors)

c) Install and get started with an external library (SFML)

Part A: Beginning Visual Studio C++

Visual Studio is a comprehensive IDE for developing software in Windows environments through a variety of languages, including C++. If you are on Linux or MacOS, you may have to resort to VS Code or prepare a Windows virtual machine so you can follow the tutorial lessons with lesser issues. Follow the steps to get started:

1. Search online for the latest “Visual Studio Community Edition”. Download and install it onto your system. During the installation, don’t forget to include “Desktop development with C++”.



Workloads | Individual components | Language packs | Installation locations

Web & Cloud (4)

ASP.NET and web development ☐
Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...

Azure development ☐
Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET and .NET Framework....

Python development ☐
Editing, debugging, interactive development and source control for Python.

Node.js development ☐
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

.NET Multi-platform App UI development ☐
Build Android, iOS, Windows, and Mac apps from a single codebase using C# with .NET MAUI.

.NET desktop development ☐
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...

Desktop development with C++ ☑
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

Universal Windows Platform development ☐
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Mobile development with C++ ☐
Build cross-platform applications for iOS, Android or Windows using C++.

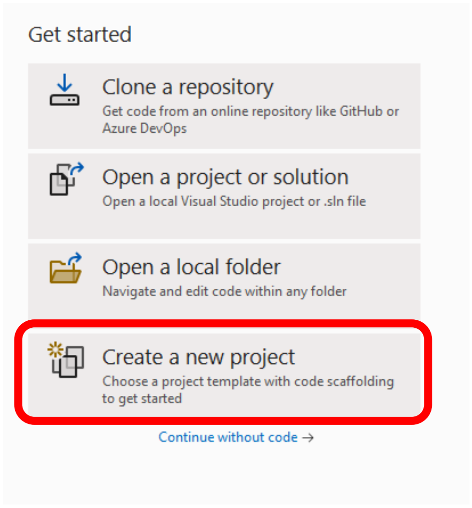
Installation details
- Visual Studio core editor
- Desktop development with C++
  - Included
    - ✓ C++ core desktop features
  - Optional
    - ☑ MSVC v143 - VS 2022 C++ x64/x86 build t...
    - ☑ C++ ATL for latest v143 build tools (x86 &...
    - ☑ Windows 10 SDK (10.0.19041.0)
    - ☑ Just-In-Time debugger
    - ☑ C++ profiling tools
    - ☑ C++ CMake tools for Windows
    - ☑ Test Adapter for Boost.Test
    - ☑ Test Adapter for Google Test
    - ☑ Live Share
    - ☑ IntelliCode
    - ☑ C++ AddressSanitizer
    - ☐ C++ MFC for latest v143 build tools (x86...
    - ☐ Windows App SDK C++ Templates
    - ☐ Windows 11 SDK (10.0.22621.0)
    - ☐ C++/CLI support for v143 build tools (Late...
    - ☐ C++ Modules for v143 build tools (x64/x8...
    - ☐ C++ Clang tools for Windows (14.0.5 - x64...
    - ☐ JavaScript diagnostics
    - ☐ Incredibuild - Build Acceleration
    - ☐ Windows 11 SDK (10.0.22000.0)

Location
C:\Program Files\Microsoft Visual Studio\2022\Community

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total space required 0 B

2. Once installed, run Visual Studio. It might ask you to log in or create an account. This is not compulsory (just choose “do it later”).
3. The Open File Menu may appear. Choose the appropriate option to “Create a new project”.



Get started

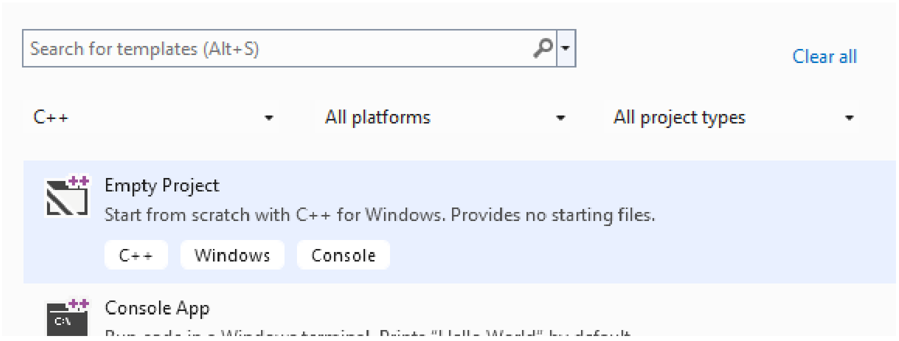
Clone a repository
Get code from an online repository like GitHub or Azure DevOps

Open a project or solution
Open a local Visual Studio project or .sln file

Open a local folder
Navigate and edit code within any folder

Create a new project
Choose a project template with code scaffolding to get started

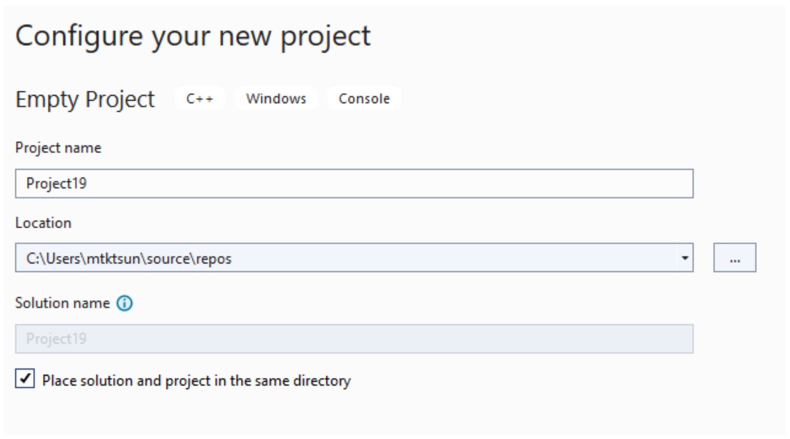
4. As a general practice, we opt for the Empty Project using the C++ language.



Empty Project
Start from scratch with C++ for Windows. Provides no starting files.
C++ | Windows | Console

Console App

5. Set an appropriate name for your project, determine a Working Directory (if you don't want to use the default one) and decide if you want to have a simpler folder arrangement or not.


Configure your new project

Empty Project C++ Windows Console

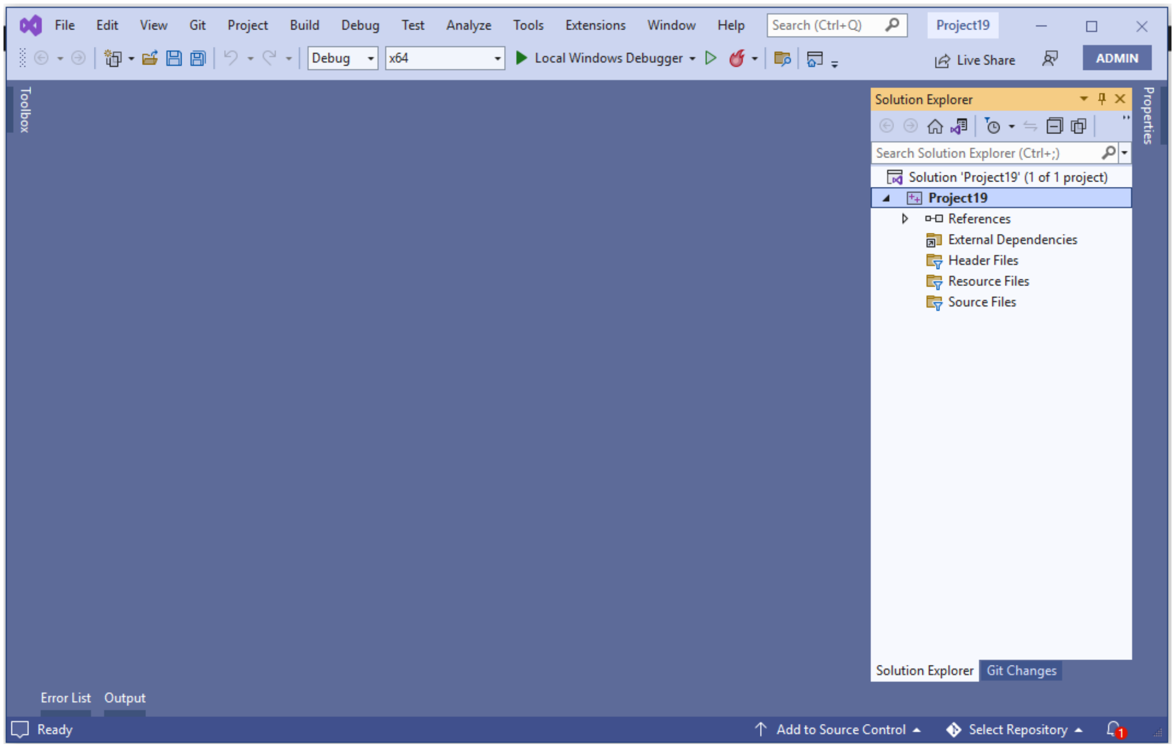
Project name: Project19

Location: C:\Users\mtktsun\source\repos

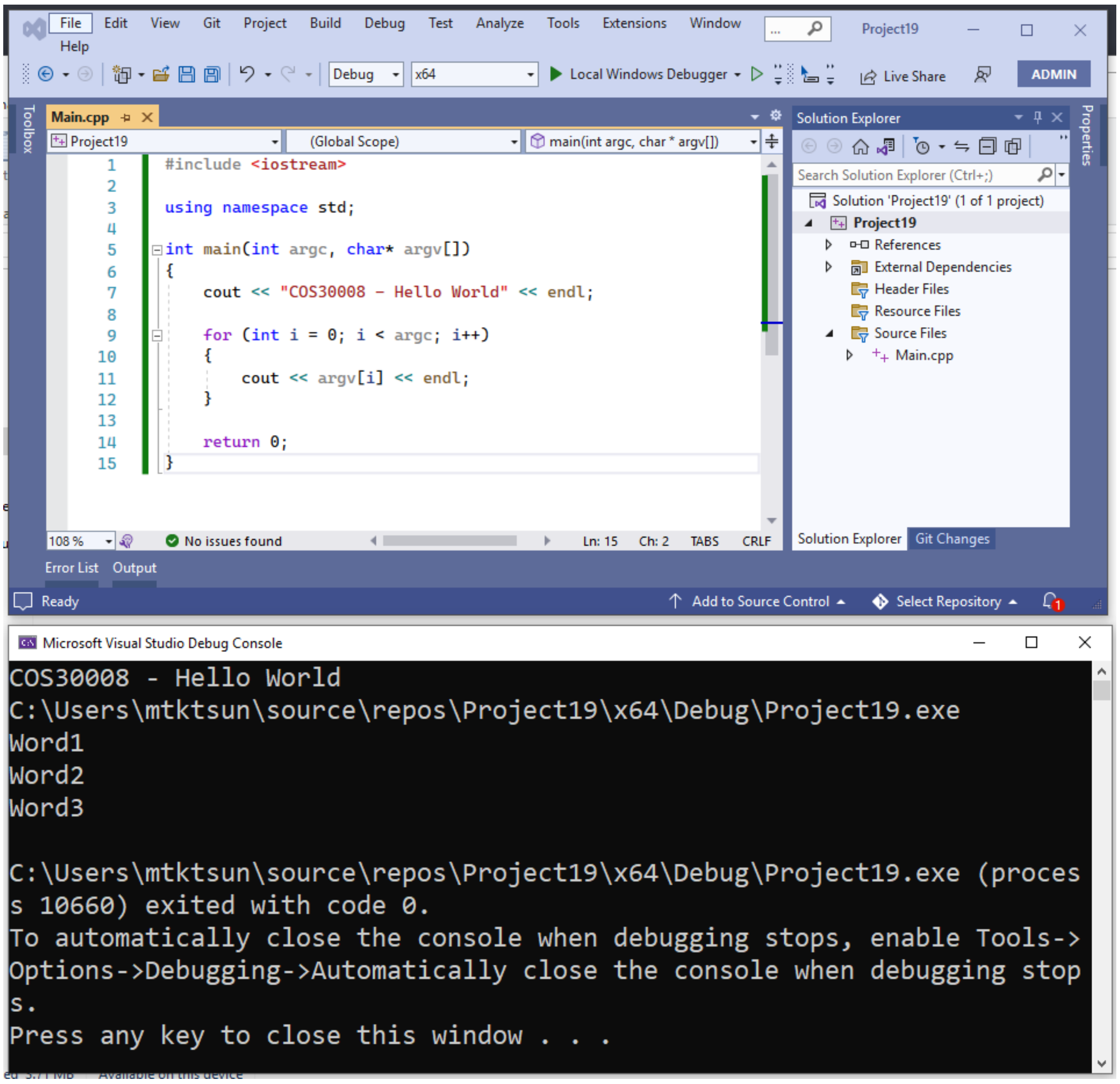
Solution name: Project19

☑ Place solution and project in the same directory

6. Click "Create" and this is what an empty project looks like in the IDE. Explore the different parts of the IDE, especially on how to:
    a. Add a New Source File
    b. Add an Existing Source File
    c. Open the File Location of your project
    d. Build (Compile & Run) a program
    e. Save the project and open a new one



7. What are you waiting for? Build your own Hello World.



```cpp
#include <iostream>

using namespace std;

int main(int argc, char* argv[])
{
    cout << "COS30008 - Hello World" << endl;

    for (int i = 0; i < argc; i++)
    {
        cout << argv[i] << endl;
    }

    return 0;
}
```

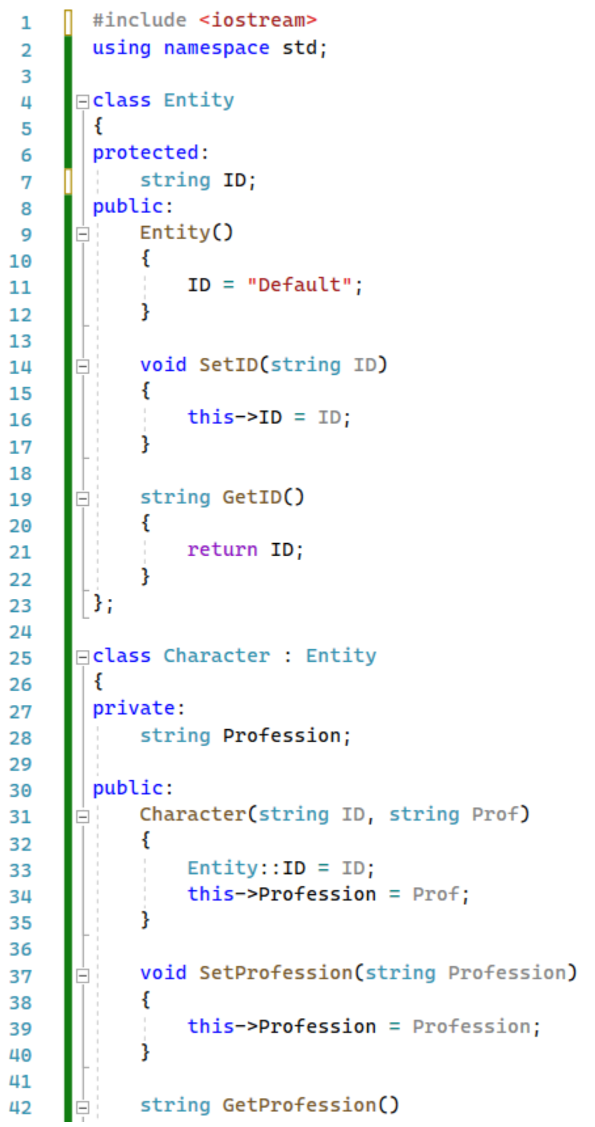
```
COS30008 - Hello World
C:\Users\mtktsun\source\repos\Project19\x64\Debug\Project19.exe
Word1
Word2
Word3

C:\Users\mtktsun\source\repos\Project19\x64\Debug\Project19.exe (proces
s 10660) exited with code 0.
To automatically close the console when debugging stops, enable Tools->
Options->Debugging->Automatically close the console when debugging stop
s.
Press any key to close this window . . .
```

Find out:

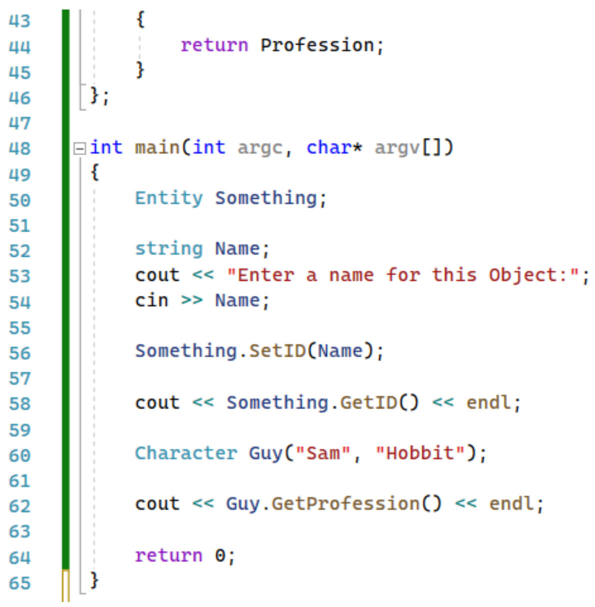
- What is "**#include <iostream>**" in C++?
- What is "**using namespace std;**" for?
- What is the "**(int argc, char* argv[])**" for?
- How to use Breakpoints for Debugging?

Part B: A Soft Revisit of OOP in C++

Create a new project in your solution, and code the following. It should execute properly.



```cpp
#include <iostream>
using namespace std;

class Entity
{
protected:
    string ID;
public:
    Entity()
    {
        ID = "Default";
    }

    void SetID(string ID)
    {
        this->ID = ID;
    }

    string GetID()
    {
        return ID;
    }
};

class Character : Entity
{
private:
    string Profession;

public:
    Character(string ID, string Prof)
    {
        Entity::ID = ID;
        this->Profession = Prof;
    }

    void SetProfession(string Profession)
    {
        this->Profession = Profession;
    }

    string GetProfession()
```



```cpp
    {
        return Profession;
    }
};

int main(int argc, char* argv[])
{
    Entity Something;

    string Name;
    cout << "Enter a name for this Object:";
    cin >> Name;

    Something.SetID(Name);

    cout << Something.GetID() << endl;

    Character Guy("Sam", "Hobbit");

    cout << Guy.GetProfession() << endl;

    return 0;
}
```

Find out:

- Which are Classes and Objects? What is the difference?
- Access Modifiers: What are **public**, **private** and **protected** for?
- Inheritance: Which is the Parent Class, and which is the Derived Class? What is the Difference?
- What is **this** for?
- Get this to work: "**cout<<Guy.GetID()<<endl**"
- Do you think "**cout**" and "**cin**" are objects? Where are they defined?
- Do you know that **<<, >>, ->, =,** and ***** are also functions? Where are they defined?

The Friend Operator Overload

The ability to extend existing operators (such as +, -, <<, etc.) so they can work with your own custom Classes is called "Friend Operator Overloading".

To demonstrate this, try adding this code to the **Main Function** and see if it works:

```
cin >> Guy;
cout << Guy;
```

This will fail, because the compiler has no idea what you want it to do. The existing operator functions for >> and << has versions that work with basic datatypes like integers, strings and etc., but it does not know what to do when one of the parameters is of a custom type.

Add these 2 methods to the ***public*** section of the **Character Class**:

```
friend istream& operator>>(istream& aIstream, Character& aCharacter)
{
    aIstream >> aCharacter.ID >> aCharacter.Profession;
    return aIstream;
}

friend ostream& operator<<(ostream& aOstream, Character& aCharacter)
{
    aOstream << "ID:" << aCharacter.ID << " Profession:" << aCharacter.Profession;
    return aOstream;
}
```

Find out:

- Does it work now? Why?
- Why do we use ***istream*** and ***ostream*** Datatypes as the parameter and return values? Hint: Look at the function's behavior – Do **aIstream** and **aOstream** objects behave in a way that is familiar to you?
- If you try "**cin>>Something;**" and "**cout>>Something;**" in the Main Function, will it work? Why?

Part C: How to Integrate an External Library? Feat. SFML

To complete this part, refer to the SFML installation and Getting Started guide I have prepared on Canvas Week 1 Activities. That document features an older version, so Download the latest one and apply the same steps.

Doing this successfully will show you why OOP in C++ is a necessary skill: because it is the only way to utilize software libraries (the true form of Code Reuse).

Also, it will be useful for expressing your Problem Set and Programming Project creations in more advanced ways.