SENG 330 Assignment 2

Purpose:

This project is to experiment with utilizing code support tools such as Doxygen for documentation, Google Testing Framework for testing and Protocol Buffers for serializing data, and Git for version control. The code that is build around this is a simple implementation of the Prototype design pattern using C++. The scenario of a exercise gym has continued over from Assignment 1, providing the context for the code in this assignment.

What has been accomplished:

-A C++ implementation of the Prototype design pattern has been written and compiles.

-The implementation has been documented with Doxygen supported comments and the corresponding documentation has been produced

-This has all been controlled through a Github repository, with commits for significant milestones in the project process

-Introduction material and installation/compilation instructions for a Windows OS are contained in this file

What has not been accomplished:

-Neither Google Protocol Buffers nor the Google Testing Frameworks have been employed for this project

-Installation and setup issues of the two tools pushed the other components forward in priority and time did not allow for either to be employed

Installation Instructions:

The first step is to either clone or download the Github repository to your computer, you may download all files packaged together as a in .zip format directly from the repository page or you can clone the repository to a Github desktop application.

-Github Desktop can be found here with its own set of instructions included: https://desktop.github.com/

A build of the code has already been included inside the Github repository

If you want to rebuild the code you may follow these instructions:

-Ensure you have Microsoft's Visual Studio 2015 installed, and along with it the Visual Studio ++ package.

-Visual Studio 2015 can be found here: https://www.visualstudio.com/en-us/products/vs-2015-product-editions.aspx

-With Visual Studio set up you may begin building the project

-Open the SENG330_Assign2 Visual Studio Project

-This will launch Visual Studio and open the project

-Next click on the Build option on the top tool bar

-Then select build solution

-This will create a Debug folder wherever you had the folder containing the project files

-Inside the Debug folder you will find the SENG330_Assign2 application executable

-Simply launch that executable and the application will run

Link to Github repository:

https://github.com/ZacBroit/SENG330_Assign2

Source Code:

```
#include <iostream>

#include <string>

using namespace std;


///Super Class GymMachine

/**

* GymMachine is the super class to all the different classes representing machines within a gym,

* in this current case just Treadmills and Rowing Machines.

*

* GymMachine holds both the amount of power consumed by the machine and the size of the machine

* as both of these variables apply to all machines in this scenario(all machines will theoretically communicate with a central

* system even something possibly non-electric like a rowing machine will still consume power for the onboard computer)

*/

class GymMachine

{

public:

	///Amount of power a particular machine will consume

	int powerConsumption;

	///The size of the machine in square feet

	int size;


	///Constructor for GymMachine, takes both power and size parameters

	GymMachine(int power, int area){

		//cout << "In GymMachine Constructor! \n";

		size = area;
```

```
        powerConsumption = power;
    };

    ///Copy Constructor for GymMachine, takes pointer to a GymMachine as a parameter to copy
    GymMachine(const GymMachine* other){
        powerConsumption = other->powerConsumption;
        size = other->size;
    }

};

///Sub Class RowingMachine
/**
* RowingMachine is a sub class of GymMachine
*
* RowingMachine's have one unique variable which is the resistance set for the machine
* RowingMachine's also work with the Prototype design pattern, so a clone function
* is used to create a new instance of the class based off of the pre-existing prototype
* referred to as 'row'
*/
class RowingMachine : public GymMachine
{
public:
    ///The amount of resistance to pulling the handle of the Rowing Machine
    int resistance;
```

```
    ///Constructor for RowingMachine, since it inherits from GymMachine it passes in default values of 25 and 7 for power and size to the super class constructor
    RowingMachine(int resist): GymMachine(25, 7)
    {
        //cout << "In Rowing Machine Constructor! \n";
        resistance = resist;
    }

    ///Copy Constructor for RowingMachine, calls the copy constructor of GymMachine to complete the copy
    RowingMachine(const RowingMachine* other): GymMachine(other){
        //cout << "In Rowing Machine copy constructor \n";
    }

    ///Clone function used as part of Prototype design pattern, returns a new RowingMachine based off the prototype 'row'
    RowingMachine* clone(int resist){
        //cout << "In Rowing Machine clone \n";
        return new RowingMachine(this);
    }
};

///Sub Class Treadmill
/**
* Treadmill is a sub class of GymMachine
*
* Treadmill's have one unique variable which is their maximum speed for the belt
* Treadmill's also work with the Prototype design pattern, so a clone function
* is used to create a new instance of the class based off of the pre-existing prototype
```

```
* referred to as 'tread'
*/
class Treadmill : public GymMachine
{
public:
	///The max speed that the spinning belt of the Treadmill can run at
	int maxSpeed;



	///Constructor for Treadmill, since it inherits from GymMachine it passes in default values of 10 and 5 for power and size to the super class constructor
	Treadmill(int speed) : GymMachine(10, 5)
	{
		//cout << "In Treadmill Constructor! \n";
		maxSpeed = speed;
	}


	///Copy Constructor for Treadmill, uses copy constructor of GymMachine
	Treadmill(const Treadmill* other): GymMachine(other){
		//cout << "In Treadmill copy constructor \n";
	}


	///clone function used in Prototype design pattern, creates new Treadmill from prototype 'tread'
	Treadmill* clone(int speed){
		//cout << "In Treadmill clone \n";
		return new Treadmill(this);
	}
```

```
};


//client code

Treadmill* tread = new Treadmill(0);
RowingMachine* row = new RowingMachine(0);

RowingMachine* createRowingMachine(int resist){
	RowingMachine* tmp = row->clone(resist);
	tmp->resistance = resist;
	return tmp;
}

Treadmill* createTreadmill(int speed){
	Treadmill* tmp = tread->clone(speed);
	tmp->maxSpeed = speed;
	return tmp;
}

int main(){

	/*
	//testing code
	cout << "In client code! \n";
	cout << "Trying to test creating a Treadmill object \n";

	Treadmill* test = createTreadmill(20);
```

```
cout << "Made Treadmill with max speed: " << test->maxSpeed << "\n";
cout << "Treadmill power is: " << test->powerConsumption << "\n";

Treadmill * test2 = createTreadmill(30);
cout << "Made Treadmill with max speed: " << test2->maxSpeed << "\n";
cout << "Treadmill power is: " << test2->powerConsumption << "\n";

*/

//assignment code
cout << "Here you can add more gym machine's to your gym \n";
cout << "Type in 'Treadmill' to add in a new Treadmill, \n";
cout << "Type in 'Rowing' to add in a new Rowing Machine, \n";
cout << "Or, type in 'exit' to finish adding new machines \n";

string machineType;
cin >> machineType;
//Loops until user enters in 'exit'
while(machineType.compare("exit")!=0){
        if(machineType.compare("Treadmill")==0){
                int speed;
                cout << "What is the max speed of the Treadmill?: \n";
                cin >> speed;
                Treadmill* tmp = createTreadmill(speed);
        }
        else if(machineType.compare("Rowing")==0){
                int resist;
                cout << "What is the resistance of the Rowing Machine?: \n";
```

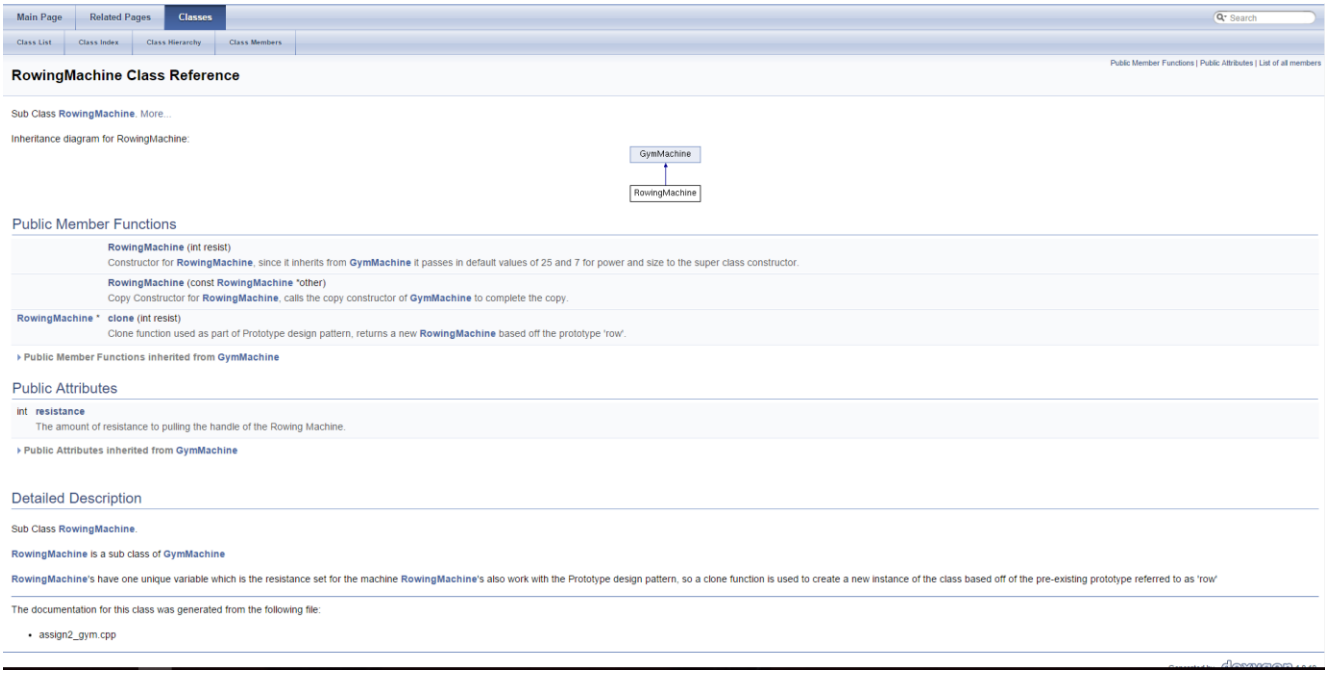
```
                cin >> resist;
                RowingMachine* tmp = createRowingMachine(resist);
            }
            else{
                cout << "Please type one of the following: \n";
                cout << "'Treadmill' to add a new Treadmill \n";
                cout << "'Rowing' to add a new Rowing Machine \n";
                cout << "'exit' to finish adding machines \n\n";
            }
            cout << "Your last entry was: " << machineType << "\n";
            cout << "What would you like to add next?: \n";
            cin >> machineType;
        }

        return 0;

}
```

Example of Doxygen Documentation:

RowingMachine Class:



Main Page | Related Pages | Classes | Search

Class List | Class Index | Class Hierarchy | Class Members

Public Member Functions | Public Attributes | List of all members

RowingMachine Class Reference

Sub Class **RowingMachine**. More...

Inheritance diagram for RowingMachine:

Public Member Functions

RowingMachine (int resist)
Constructor for **RowingMachine**, since it inherits from **GymMachine** it passes in default values of 25 and 7 for power and size to the super class constructor.

RowingMachine (const **RowingMachine** *other)
Copy Constructor for **RowingMachine**, calls the copy constructor of **GymMachine** to complete the copy.

RowingMachine * **clone** (int resist)
Clone function used as part of Prototype design pattern, returns a new **RowingMachine** based off the prototype 'row'.

Public Member Functions inherited from GymMachine

Public Attributes

int **resistance**
The amount of resistance to pulling the handle of the Rowing Machine.

Public Attributes inherited from GymMachine

Detailed Description

Sub Class **RowingMachine**.

RowingMachine is a sub class of **GymMachine**

RowingMachine's have one unique variable which is the resistance set for the machine **RowingMachine**'s also work with the Prototype design pattern, so a clone function is used to create a new instance of the class based off of the pre-existing prototype referred to as 'row'

The documentation for this class was generated from the following file:

- assign2_gym.cpp