**Object-Oriented Programming**

At the end of this worksheet is the code for the Distance.h file. Create a c++ project and add the Distance.h file to your project. In the main .cpp file, make sure to include “Distance.h”.

**Constructors**

1. Observe the Distance class. What variables are in the private section? **Feet, Inches**
2. Observe the adjustDist() method. Why do you suppose it is private instead of public? **It is a function that doesn’t need to be accessed by other functions and is for internal use only in adjusting.**
3. How many constructors are present in the Distance class? **2**
4. Let's code an object using the default constructor. Key in the following.

Distance d1;

cout << "Distance 1: " << d1 << endl;

1. What do you think will be displayed in the output? Run the project to see if you were correct. **I believe it would print “Distance 1: Feet: 0 Inches : 0” because in the overloaded operator, it defines how it should be printed.**
2. Observe the Overloaded Distance Constructor. How many parameters are there in the parenthesis? **2**
3. Let’s code an object using the Overloaded Constructor. Key in the following.

Distance d2(1, 15);

cout << "Distance 2: " << d2 << endl;

1. Will it display as 1 foot and 15 inches? Why or why not. **No, because after declaring the variable, the adjustDist() function is called and changes it to 2 feet 3 inches**
2. Notice the second parameter has a default value. This means that if it is not present, the value will default to 0. Therefore we can create an object using one parameter. Key in the code to create a third object named d3. This time use only one parameter of 5. Add a cout statement for that object as well. What do you think will have the value 5, inches or feet? Run the program to see if you are correct. **Feet should have the value of 5 because the default parameter is the inches, meaning the parameter you set will go to the feet. This is also the case because parameters are assigned in the same order they are declared, and feet is declared first in the list.**

**Extraction/Insertion Operators**

1. We have been using the insertion << operator to display the Distance object. Notice the keyword friend before the operator declaration. Look below in the function definition. Is the keyword friend used there as well? **No**
2. A friend function is not part of the class. Therefore it needs to be sent the object as a parameter. In this case there are two parameters being sent, what are they? **The stream, which is the console in this case, and the object to be manipulated**
3. Observe how the private variables are being used in the friend function. Notice that the instance of the object is used to access the variable. Let's try an experiment. Remove the “d.” from in front of the variable so it is just left with feet instead of d.feet. Run the program. What error do you receive? Why do you think that error is there? **You get an undeclared identifier error because “feet” is not a global variable and is not defined in the scope, whereas with the d. Object identifier, it tells the compiler that it should find that variable in the defined “d” object.**
4. Observe the Extraction operator >>. How many variables are being added to the stream input? **2 Variables**
5. Go back to main and add a cin statement for the d1 object (and a cout statement to prompt the user to enter a distance. Run the program and key in 5 2. Did it change the value of d1? **Yes**
6. Run the program again. This time only enter one number such as 5. Does the program allow you to only enter one number or does it force both values? Why do you think it behaves this way? **It forces both values because in the overloading of the extraction operator, we call for 2 different values. If you only give it a single value, it doesn’t have all the information you need to give you the response.**

**Operators**

1. Observe the operator+ in the Distance class. Is it a friend function? How many parameters does the operator have? **No, it is not a friend function because it is a member function. It has a single parameter.**
2. Go back to main(). We will use the + operator to add two distances together by keying in the following code

Distance d4 = d2 + d3;

cout << "Distance 4: " << d4 << endl;

1. Run the program, did it add the values together correctly? (Should be 7 feet 3 inches unless you used other values) **Yes**
2. Add the code for a subtraction operator that will subtract the two Distances.
3. Create a variable d5 that will subtract two distances. Run the program to make sure it works.
4. Copy and paste your completed code over the code below.

**Distance.h**

#include <iostream>

using namespace std;

class Distance {

private:

int feet;

int inches;

void adjustDist();

public:

//Default Constructor

Distance() ;

//Overloaded Constructor

Distance(int f, int i = 0);

//Overloaded Addition Operator

Distance operator+(const Distance&);

//Overloaded Minus Operator

Distance operator-(const Distance&);

//Insertion Operator

friend ostream &operator<<( ostream &output, const Distance &D );

//Extraction Operator

friend istream &operator>>( istream &input, Distance &D );

};

Distance::Distance() {

feet = 0;

inches = 0;

}

Distance::Distance(int f, int i) {

feet = f;

inches = i;

adjustDist();

}

void Distance::adjustDist(){

if(inches > 12){

int addFeet = inches / 12;

feet += addFeet;

inches %= 12;

}

}

Distance Distance::operator+(const Distance& d){

Distance dist;

dist.inches = inches + d.inches;

dist.feet = feet + d.feet;

adjustDist();

return dist;

}

Distance Distance::operator-(const Distance& d){

Distance dist;

dist.inches = inches - d.inches;

dist.feet = feet - d.feet;

adjustDist();

return dist;

}

ostream &operator<<( ostream &out, const Distance &d ) {

out << "Feet: " << d.feet << " Inches : " << d.inches;

return out;

}

istream &operator>>( istream &input, Distance &d ) {

input >> d.feet >> d.inches;

return input;

}

**Distance.cpp**

#include <iostream>

#include "Distance.h"

using namespace std;

int main() {

Distance d1;

cout << "Distance 1: " << d1 << endl;

Distance d2(1, 15);

cout << "Distance 2: " << d2 << endl;

Distance d3(5);

cout << "Distance 3: " << d3 << endl;

Distance d4 = d2 + d3;

cout << "Distance 4: " << d4 << endl;

Distance d5 = d2 - d3;

cout << "Distance 5: " << d5 << endl;

return 0;

}