**Lab Description:**

\*Lab description not on write up

\*\*In README file

**Main.cpp:**

#include <iostream> // cout, endl

#include <stdio.h> // printf

#include "Employee.h" // Employee class

#include "Executive.h" // Executive class

#include "Hourly.h" // Hourly class

int main() {

std::cout << "Basic Employee Tests: " << std::endl; // Employee tests

Employee gen1("Ryan", 10.25);

std::cout << "Printed Info: " << std::endl;

gen1.print();

std::cout << "The returned payrate from the pay() function is: " << gen1.pay() << std::endl;

std::cout << "===============================================\n" << std::endl;

std::cout << "Basic Hourly Tests: " << std::endl; // Hourly Tests

Hourly hour("Skylar", 15.65);

std::cout << "Print off initialization: " << std::endl;

hour.print();

std::cout << "Adding some hours worked.... " << std::endl;

hour.addHours(40.5);

std::cout << "Printing now: " << std::endl;

hour.print();

std::cout << "The returned payrate form the pay() function is: " << hour.pay() << std::endl;

std::cout << "===============================================\n" << std::endl;

std::cout << "Basic Executive Tests: " << std::endl; // Executive Tests

Executive exec("Mary", 40.5, 125000);

std::cout << "Printed Info: " << std::endl;

exec.print();

std::cout << "Setting a new bonus...." << std::endl;

exec.setBonus(500000);

exec.print();

std::cout << "The returned payrate from the pay() function is: " << exec.pay() << std::endl;

std::cout << "===============================================\n" << std::endl;

std::cout << "Starting Employee pointer array tests: " << std::endl;

Employee\* empArr[3];

Employee\* emp1 = new Employee("James", 10.25);

Employee\* emp2 = new Hourly("Fred", 19.26);

Employee\* emp3 = new Executive("Charles", 12.6, 750000);

empArr[0] = emp1;

empArr[1] = emp2;

empArr[2] = emp3;

std::cout << "Basic Employee Tests: " << std::endl; // Employee pointer tests

std::cout << "Printed Info: " << std::endl;

empArr[0]->print();

std::cout << "The returned payrate from the pay() function is: " << empArr[0]->pay() << std::endl;

std::cout << "===============================================\n" << std::endl;

std::cout << "Basic Hourly Tests: " << std::endl; // Hourly pointer tests

std::cout << "Print off initialization: " << std::endl;

empArr[1]->print();

std::cout << "Adding some hours worked.... " << std::endl;

((Hourly\*)empArr[1])->addHours(61.2);

std::cout << "Printing now: " << std::endl;

empArr[1]->print();

std::cout << "The returned payrate form the pay() function is: " << empArr[1]->pay() << std::endl;

std::cout << "===============================================\n" << std::endl;

std::cout << "Basic Executive Tests: " << std::endl; // Executive pointer tests

std::cout << "Printed Info: " << std::endl;

empArr[2]->print();

std::cout << "Setting a new bonus...." << std::endl;

((Executive\*)empArr[2])->setBonus(1250000);

empArr[2]->print();

std::cout << "The returned payrate from the pay() function is: " << empArr[2]->pay() << std::endl;

std::cout << "===============================================\n" << std::endl;

return 0;

}

**Sample Output:**

Basic Employee Tests:

Printed Info:

Ryan's payrate is 10.25

The returned payrate from the pay() function is: 10.25

===============================================

Basic Hourly Tests:

Print off initialization:

Skylar's payrate is 15.65

Worked 0 hour(s)

Adding some hours worked....

Printing now:

Skylar's payrate is 15.65

Worked 40.5 hour(s)

The returned payrate form the pay() function is: 633.825

===============================================

Basic Executive Tests:

Printed Info:

Mary's payrate is 40.5

Bonus: 125000

Setting a new bonus....

Mary's payrate is 40.5

Bonus: 500000

The returned payrate from the pay() function is: 500040

===============================================

Starting Employee pointer array tests:

Basic Employee Tests:

Printed Info:

James's payrate is 10.25

The returned payrate from the pay() function is: 10.25

===============================================

Basic Hourly Tests:

Print off initialization:

Fred's payrate is 19.26

Worked 0 hour(s)

Adding some hours worked....

Printing now:

Fred's payrate is 19.26

Worked 61.2 hour(s)

The returned payrate form the pay() function is: 1178.71

===============================================

Basic Executive Tests:

Printed Info:

Charles's payrate is 12.6

Bonus: 750000

Setting a new bonus....

Charles's payrate is 12.6

Bonus: 1.25e+06

The returned payrate from the pay() function is: 1.25001e+06

===============================================

**Employee.h:**

#ifndef \_EMPLOYEE\_H\_

#define \_EMPLOYEE\_H\_

#include <iostream> // cout, endl

class Employee {

protected:

double payRate;

std::string name;

public:

Employee(); // Default Ctor

Employee(std::string, double); // Non-Default Ctor

virtual double pay(); // Returns the amount the employee should be paid

virtual void print() const; // Prints to stdout the employee information

};

#endif

**Employee.cpp:**

#include "Employee.h"

/\*

\* Default Constructor:

\* Initializes all values to base

\*/

Employee::Employee() {

name = "";

payRate = 0;

}

/\*

\* Non-Default Constructor:

\* Initializes all protected members of the employee class

\*/

Employee::Employee(std::string empName, double empRate) {

name = empName;

payRate = empRate;

}

/\*

\* pay Function:

\* Returns the amount the employee should be paid

\*/

double Employee::pay() {

return payRate;

}

/\*

\* print Function:

\* Prints to stdout the employee information

\*/

void Employee::print() const {

std::cout << name << "'s payrate is " << payRate << std::endl;

}

**Hourly.h:**

#ifndef \_HOURLY\_

#define \_HOURLY\_

#include "Employee.h"

#include <iostream>

class Hourly : public Employee {

private:

float hoursWorked;

public:

Hourly(std::string, double); // Non-Default Ctor

void addHours(float); // Adds passed float to hoursWorked

double pay() override; // Overrides pay() parent function and returns hoursWorked times payRate

void print() const override; // Overrides print() parent function and prints out all info

};

#endif

**Hourly.cpp:**

#include "Hourly.h"

/\*

\* Non-Default Constructor

\* Initializes all private members

\*/

Hourly::Hourly(std::string empName, double rate) : Employee(empName, rate) {

hoursWorked = 0;

}

/\*

\* addHours Function

\* Adds passed float to hoursWorked

\*/

void Hourly::addHours(float add) {

hoursWorked += add;

}

/\*

\* pay Function

\* Overrides pay() parent function and returns hoursWorked time payRate

\*/

double Hourly::pay() {

return payRate \* hoursWorked;

}

/\*

\* print Function

\* Overrides the print() parent function and prints out all info

\*/

void Hourly::print() const {

Employee::print();

std::cout << "Worked " << hoursWorked << " hour(s)" << std::endl;

}

**Executive.h:**

#ifndef \_EXECUTIVE\_

#define \_EXECUTIVE\_

#include "Employee.h"

#include <iostream>

class Executive : public Employee {

private:

double bonus;

public:

Executive(std::string, double, double); // Non-Default Ctor

void setBonus(double); // Sets the bonus for the executive

double pay() override; // Overrides the pay() parent method and returns the total pay

void print() const override; // Overrides the print() parent method and prints the Executive info

};

#endif

**Executive.cpp:**

#include "Executive.h"

/\*

\* Non-Default Constructor

\* Initializes all private members

\*/

Executive::Executive(std::string empName, double rate, double bon) : Employee(empName, rate) {

bonus = bon;

}

/\*

\* setBonus Function

\* Sets the bonus for the executive

\*/

void Executive::setBonus(double bon) {

bonus = bon;

}

/\*

\* pay Function

\* Overrides the pay() parent method and returns the total pay

\*/

double Executive::pay() {

double total = Employee::pay() + bonus;

bonus = 0;

return total;

}

/\*

\* print Function

\* Overrides the print() parent method and prints the Executive info

\*/

void Executive::print() const {

Employee::print();

std::cout << "Bonus: " << bonus << std::endl;

}