Screenshots:

Graphical user interface, text

Description automatically generated with medium confidence

Contents

[Source.cpp 2](#_Toc68388249)

[Employee.cpp 3](#_Toc68388250)

[Employee.h 4](#_Toc68388251)

[Hourly.h 5](#_Toc68388252)

[Hourly.cpp 6](#_Toc68388253)

[Salary.h 7](#_Toc68388254)

[Salary.cpp 8](#_Toc68388255)

[Manager.h 8](#_Toc68388256)

[Manager.cpp 9](#_Toc68388257)

**Code:**

# Source.cpp

/\*

Edward Alvarado

Class: CIS247C

Date: 4/3/2021

Week 5 Lab Inheritence

\*/

#include <iostream>

#include <string>

#include <conio.h>

#include "Hourly.h"

#include "Manager.h"

using namespace std;

//prototypes

void displayEmployee(Employee\* emp);

/// Entry point to the applicatrion

int main()

{

Hourly h1(40, 10, "Dan", "Doe" , "000123456", "8133871234");

Manager m1(4000,40000, "Man", "Doe", "000123458", "8133871234");

Salary s1(42000, "Ran", "Boe", "000123451", "8133871274");

cout << "Size of Hourly: " << sizeof(Hourly);

cout << "\nMemory of Hourly: " << &h1;

cout << endl;

displayEmployee(&h1);

displayEmployee(&s1);

displayEmployee(&m1);

// Pause

cout << "\nPress any key to continue...";

\_getch();

return 0;

}

void displayEmployee(Employee\* emp)

{

Manager\* m1 = dynamic\_cast<Manager\*>(emp); // try to convert Employee parent object to a Manager child object

if (m1 != NULL) // if the mgr is not NULL, then we have a Manager object!

{

cout << "\nBonus: $" << m1->getBonus() << endl;

return;

}

Salary\* s1 = dynamic\_cast<Salary\*>(emp); // try to convert Employee parent object to a Manager child object

if (s1 != NULL) // if the mgr is not NULL, then we have a Manager object!

{

cout << "\nAnnual Salary: $" << s1->getAnnualSalary() << endl;

return;

}

Hourly\* h1 = dynamic\_cast<Hourly\*>(emp); // try to convert Employee parent object to a Manager child object

if (h1 != NULL) // if the mgr is not NULL, then we have a Manager object!

{

cout << "\nHourly: $" << h1->calculatePay() << endl;

}

}/// Entry point to the applicatrion

int main()

{

//creating a default person object

Person person1;

cout << person1.toString() << endl;

// change the values by setrting the nameand ssn, and add glasses object

person1.setName("John");

person1.setSsn("111-11-1111");

# Employee.cpp

#include "Employee.h"

Employee::Employee()

{

}

Employee::Employee(string fName, string lName, string ssn, string phone)

{

this->fName = fName;

this->lName = lName;

this->ssn = ssn;

this->phone = phone;

}

Employee::~Employee(){}

string Employee::toString()

{

return "First Name: " + fName + ", Last Name: " + lName + ", SSN: " + ssn + ", Phone Number: " + phone + ".";

}

float Employee::calculatePay()

{

return 0.0f;

}

string Employee::getfName()

{

return fName;

}

void Employee::setfName(string fName)

{

if (fName.length() > 0 && fName.length() < 30)

this->fName = fName;

else this->fName = "unknown";

}

string Employee::getlName()

{

return lName;

}

void Employee::setlName(string lName)

{

if (lName.length() > 0 && lName.length() < 30)

this->lName = lName;

else this->lName = "unknown";

}

string Employee::getssn()

{

return ssn;

}

void Employee::setssn(string ssn)

{

if (ssn.length() == 9)

this->ssn = ssn;

else

this->ssn = "unknown";

}

string Employee::getphone()

{

return phone;

}

void Employee::setphone(string phone)

{

if (phone.length() == 10 || phone.length() == 11)

this->phone = phone;

else

this->phone = "unknown";

}

# Employee.h

#pragma once

#include <string>

using namespace std;

class Employee

{

protected:

// attributes to be carried over to child

string fName;

string lName;

string ssn;

string phone;

public:

// constructors and destructors

Employee();

Employee(string fName, string lName, string ssn, string phone);

virtual ~Employee();

//behaviors in the public section

virtual float calculatePay();

virtual string toString();

//accessors and mutators

string getfName();

void setfName(string fName);

string getlName();

void setlName(string lName);

string getssn();

void setssn(string ssn);

string getphone();

void setphone(string phone);

};

# Hourly.h

#pragma once

#include <string>

using namespace std;

class Employee

{

protected:

// attributes to be carried over to child

string fName;

string lName;

string ssn;

string phone;

public:

// constructors and destructors

Employee();

Employee(string fName, string lName, string ssn, string phone);

virtual ~Employee();

//behaviors in the public section

virtual float calculatePay();

virtual string toString();

//accessors and mutators

string getfName();

void setfName(string fName);

string getlName();

void setlName(string lName);

string getssn();

void setssn(string ssn);

string getphone();

void setphone(string phone);

};

# Hourly.cpp

#include "Hourly.h"

#include "Employee.h"

Hourly::Hourly()

{

hours = 0.0f;

rate = 0.0f;

}

Hourly::Hourly(float hours, float rate, string fname, string lname, string ssn, string phone):Employee(fname, lname, ssn, phone)

{

this->hours = hours;

this->rate = rate;

}

Hourly::~Hourly() {}

float Hourly::calculatePay()

{

return (hours \* rate);

}

string Hourly::toString()

{

string strHours = to\_string(hours);

string strRate = to\_string(rate);

return Employee::toString() + "Hours: " + strHours + ", Rate: " + strRate;

}

float Hourly::getHours()

{

return hours;

}

void Hourly::setHours(float hours)

{

if (hours > 0 && hours < 120)

this->hours = hours;

else

this->hours = 40.0 \*rate + (hours - 40) \* rate \* 1.5;;

}

float Hourly::getRate()

{

return rate;

}

void Hourly::setRate(float rate)

{

if (rate > 7)

this->rate = rate;

else

this->rate = 0.0f;

}

# Salary.h

#pragma once

#include "Employee.h"

#include <string>

using namespace std;

class Salary: public Employee

{

protected:

double annualSalary;

public:

Salary();

Salary(double annualSalary, string fname, string lname, string ssn, string phone);

virtual ~Salary();

//behaviors in public section

virtual float calculatePay();

virtual string toString();

//accessors and mutators

double getAnnualSalary();

void setAnnualSalary();

};

# Salary.cpp

#include "Salary.h"

#include "Employee.h"

using namespace std;

Salary::Salary()

{

annualSalary = 0;

}

Salary::Salary(double annualSalary, string fname, string lname, string ssn, string phone):Employee(fname, lname, ssn, phone)

{

this->annualSalary = annualSalary;

}

Salary::~Salary() {}

float Salary::calculatePay()

{

return (annualSalary / 52.0f);

}

string Salary::toString()

{

string strAnnualSalary = to\_string(annualSalary);

return Employee::toString() + "Annual Salary: " + strAnnualSalary + ".";

}

double Salary::getAnnualSalary()

{

return annualSalary;

}

void Salary::setAnnualSalary()

{

if (annualSalary > 0)

this->annualSalary = annualSalary;

else

this->annualSalary = 0.0;

}

# Manager.h

#pragma once

#include "Salary.h"

#include <string>

using namespace std;

class Manager :public Salary

{

protected:

double bonus;

public:

Manager();

Manager(double bonus, double annualSalary, string fname, string lname, string ssn, string phone);

~Manager();

virtual float calculatePay();

virtual string toString();

//accessors and mutators

double getBonus();

void setBonus();

};

# Manager.cpp

#include "Manager.h"

Manager::Manager()

{

bonus = 0.0;

}

Manager::Manager(double bonus, double annualSalary, string fname, string lname, string ssn, string phone):Salary(annualSalary, fname, lname, ssn, phone)

{

this->bonus = bonus;

}

Manager::~Manager(){}

float Manager::calculatePay()

{

return bonus/52.0 + Salary::calculatePay();

}

string Manager::toString()

{

string strBonus = to\_string(bonus);

return Salary::toString() + "Bonus: " + strBonus + ".";

}

double Manager::getBonus()

{

return bonus;

}

void Manager::setBonus()

{

if (bonus > 0)

this->bonus = bonus;

else

this->bonus = 0.0;

}