**C++ Part II (INFO1-CE9265) Spring 2015 – Homework 1**

Clement Chan

**Question 1:**

Employee.h

#ifndef EMPLOYEE\_H

#define EMPLOYEE\_H

#include <string>

#include <iostream>

namespace SavitchEmployees

{

class Employee

{

protected:

std::string name;

std::string ssn;

double netPay;

public:

Employee();

Employee(std::string theName, std::string theSsn);

std::string getName() const;

std::string getSsn() const;

double getNetPay() const;

void setName(std::string newName);

void setSsn(std::string newSsn);

void setNetPay(double newNetPay);

void printCheck() const;

};

} //SavitchEmployees

#endif

Employee.cpp

#include <string>

#include <cstdlib>

#include <iostream>

#include "employee.h"

using std::cout;

namespace SavitchEmployees

{

Employee::Employee() : name("No name yet"), ssn("No number yet"), netPay(0)

{

//deliberately empty

}

Employee::Employee(std::string theName, std::string theNumber) : name(theName),

ssn(theNumber), netPay(0)

{

//deliberately empty

}

std::string Employee::getName() const

{

return name;

}

std::string Employee::getSsn() const

{

return ssn;

}

double Employee::getNetPay() const

{

return netPay;

}

void Employee::setName(std::string newName)

{

name = newName;

}

void Employee::setSsn(std::string newSsn){

ssn = newSsn;

}

void Employee::setNetPay(double newNetPay){

netPay = newNetPay;

}

void Employee::printCheck()const{

cout <<"\nERROR: printCheck function called for an \n"

<<"Undifferentiated employee. Aborting the program. \n"

<<"Check with the author of the progress about this bug.\n";

exit(1);

}

}

SalariedEmployee.h

#ifndef SALARIEDEMPLOYEE\_H

#define SALARIEDEMPLOYEE\_H

#include <string>

#include "employee.h"

namespace SavitchEmployees

{

class SalariedEmployee : public Employee

{

protected:

double salary; //weekly

public:

SalariedEmployee();

SalariedEmployee(std::string theName, std::string theSsn);

SalariedEmployee(std::string theName, std::string theSsn,

double theWeeklySalary);

double getSalary() const;

void setSalary(double newSalary);

void printCheck();

};

}

#endif

SalariedEmployee.cpp

#include <string>

#include <cstdlib>

#include <iostream>

#include "SalariedEmployee.h"

namespace SavitchEmployees

{

SalariedEmployee::SalariedEmployee() : Employee(), salary(0)

{

//deliberately empty

}

SalariedEmployee::SalariedEmployee(std::string theName, std::string theNumber) :

Employee(theName, theNumber)

{

//deliberately empty

}

SalariedEmployee::SalariedEmployee(std::string theName, std::string theNumber,

double theWeeklySalary) : Employee(theName, theNumber), salary(theWeeklySalary)

{

//deliberately empty

}

double SalariedEmployee::getSalary() const

{

return salary;

}

void SalariedEmployee::setSalary(double newSalary)

{

salary = newSalary;

}

void SalariedEmployee::printCheck(){

setNetPay(salary);

std::cout<< "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

std::cout<< "Pay to the order of " << getName() << std::endl;

std::cout<< "The sum of " << getNetPay() << " Dollars\n" ;

std::cout<< "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

std::cout<< "Check Stub Not Negotiable \n";

std::cout<< "Employee Number: " << getSsn() << std::endl;

std::cout<< "Salaried Employee. Regular Pay: " << getSalary() << std::endl;

std::cout<< "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

std::cout<< "\n";

}

}

Administrator.h

#ifndef ADMIN\_H

#define ADMIN\_H

#include <string>

#include "SalariedEmployee.h"

namespace SavitchEmployees

{

class Administrator : public SalariedEmployee

{

private:

std::string Admin\_Title;

std::string Response;

std::string Supervisor;

public:

Administrator();

Administrator(std::string theName, std::string theSsn, std::string

theAdmin\_Title, std::string theResponse, std::string theSupervisor,

double thesalary);

void setName();

void setAdminTitle();

void setSalary();

void setSupervisor();

void setResponse();

void setssn();

std::string getSupervisor();

std::string getAdminTitle();

std::string getResponse();

void print();

void printCheck();

};

}

#endif

Administrator.cpp

#include <string>

#include <cstdlib>

#include <iostream>

#include "Administrator.h"

namespace SavitchEmployees

{

Administrator::Administrator() : SalariedEmployee(), Admin\_Title("No Title yet"),

Response("No Response yet"), Supervisor("No Supervisor yet")

{

//deliberately empty

}

Administrator::Administrator(std::string theName, std::string theSsn, std::string

theAdmin\_Title, std::string theResponse, std::string theSupervisor,

double thesalary) : SalariedEmployee(theName, theSsn, thesalary),

Admin\_Title(theAdmin\_Title), Response(theResponse), Supervisor(theSupervisor)

{

//deliberately empty

}

void Administrator::setName()

{

std::string admin\_name;

std::cout<<"Please enter the admin's name: " << std::endl;

std::cin >> admin\_name;

name = admin\_name;

}

void Administrator::setSupervisor()

{

std::string name\_supervisor;

std::cout<<"Please enter the admin's supervisor name: " << std::endl;

std::cin >> name\_supervisor;

Supervisor = name\_supervisor;

}

std::string Administrator::getSupervisor()

{

return Supervisor;

}

void Administrator::setSalary()

{

double newSalary;

std::cout<<"Please enter the admin's salary: " << std::endl;

std::cin >> newSalary;

salary = newSalary;

}

void Administrator::setAdminTitle()

{

std::string admintitle;

std::cout<<"Please enter the admin's title: " << std::endl;

std::cin >> admintitle;

Admin\_Title= admintitle;

}

std::string Administrator::getAdminTitle()

{

return Admin\_Title;

}

void Administrator::setResponse()

{

std::string Responsibility;

std::cout<<"Please enter the admin's responsibility: " << std::endl;

std::cin >> Responsibility;

Response = Responsibility;

}

std::string Administrator::getResponse()

{

return Response;

}

void Administrator::setssn()

{

std::string theSsn;

std::cout<<"Please enter the admin's Ssn: " << std::endl;

std::cin >> theSsn;

ssn = theSsn;

}

void Administrator::print(){

std::cout <<" " << std::endl;

std::cout <<"---------------------------------Summary of Admin -------------------

-" << std::endl;

std::cout <<"The Name of the Administrator is: " << getName() << std::endl;

std::cout <<"The Name of the Supervisor is: " << getSupervisor() << std::endl;

std::cout <<"The Admin's SSN is: " << getSsn() << std::endl;

std::cout <<"The Name of the Admin's Salary is: " << getSalary() << std::endl;

std::cout <<"The Name of the Admin's Title is: " << getAdminTitle() << std::endl;

std::cout <<"The Name of the Admin's Responsibility is: " << getResponse() <<

std::endl;

}

void Administrator::printCheck(){

setNetPay(salary);

std::cout<< "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

std::cout<< "Pay to the order of " << getName() << std::endl;

std::cout<< "The sum of " << getNetPay() << " Dollars\n" ;

std::cout<< "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

std::cout<< "Check Stub Not Negotiable \n";

std::cout<< "Employee Number: " << getSsn() << std::endl;

std::cout<< "Salaried Employee. Regular Pay: " << getSalary() << std::endl;

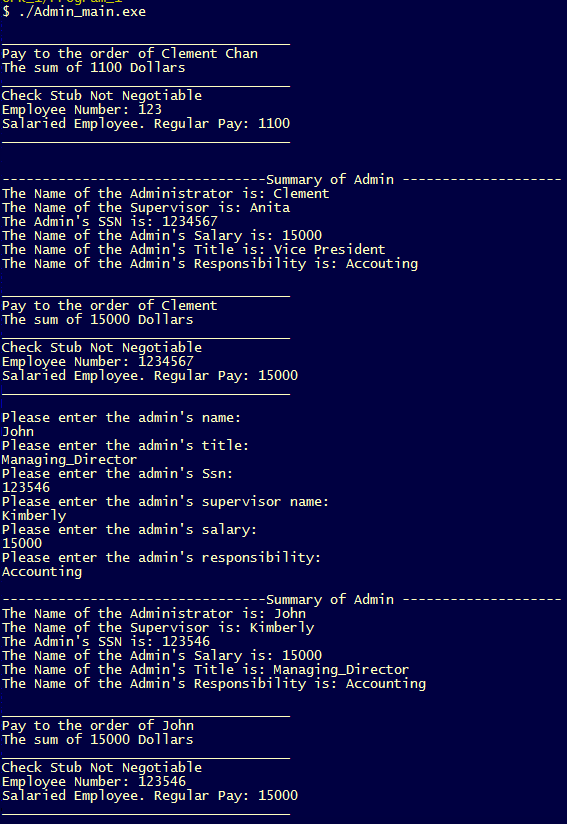
std::cout<< "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

std::cout<< "\n";

}

}

**Output**

****

Compile\_All.txt:

g++ -c Employee.cpp && g++ -c Main.cpp && g++ -c SalariedEmployee.cpp && g++ -c Administrator.cpp && g++ -o Admin\_main Employee.o Main.o SalariedEmployee.o Administrator.o

**Question 4:**

Person.h

#ifndef PERSON\_H

#define PERSON\_H

#include <string>

#include <iostream>

#include <cstdlib>

class Person{

private:

std::string name;

public:

Person(); //Default Constructor

Person(std::string theName);//Parameterized Constructor

Person(const Person& theObject);//Copy Constructor

std::string getName() const;

Person &operator=(const Person& rtSide); //operator overloading

friend std::istream& operator >> (std::istream& inStream, Person& personObject);

friend std::ostream& operator << (std::ostream& outStream, const Person&

personObject);

};

#endif

Person.cpp

#include "Person.h"

Person::Person(){

name = "Name Not Set";

}

Person::Person(std::string theName){

name = theName;

}

Person::Person(const Person& theObject){

name = theObject.name;

}

std::string Person::getName() const{

return name;

}

Person &Person::operator=(const Person& rtSide){

name = rtSide.name;

return \*this;

}

std::istream& operator >> (std::istream& inStream, Person& personObject){

std::string Person\_Name;

std::cout <<" " << std::endl;

std::cout <<"Please enter the Owner's name: " << std::endl;

inStream >> Person\_Name;

personObject.name = Person\_Name;

}

std::ostream& operator << (std::ostream& outStream, const Person& personObject){

std::cout <<" " << std::endl;

std::cout << "The name of the person entered is: " << std::endl;

outStream << personObject.name;

return outStream;

}

Vehicle.h

#ifndef VEHICLE\_H

#define VEHICLE\_H

#include <string>

#include "Person.h"

class Vehicle{

protected:

std::string manufact\_name;

int number\_of\_cylinders;

Person Owner;

public:

Vehicle();

Vehicle(std::string themanufactname, int thenumofcylinders);

std::string getownerName(const Person &P) const;

//Setter and Getter for manufacturer\_name

void set\_manufact\_name(std::string manufactory\_name);

std::string get\_manufact\_name() const;

//Setter and Getter for number of cylinders

void set\_number\_of\_cylinders(int num\_cylinders);

int get\_num\_of\_cylinders() const;

};

#endif

Vehicle.cpp

#include "Vehicle.h"

Vehicle::Vehicle(): manufact\_name("No\_Name\_yet"), number\_of\_cylinders(0){

//Constructor, Empty

}

Vehicle::Vehicle(std::string themanufactname, int thenumofcylinders): manufact\_name(themanufactname), number\_of\_cylinders(thenumofcylinders){

//Parameterized Constructor, Empty

}

std::string Vehicle::getownerName(const Person &P) const{

return P.getName();

}

void Vehicle::set\_manufact\_name(std::string manufactory\_name){

manufact\_name = manufactory\_name;

}

std::string Vehicle::get\_manufact\_name() const{

return manufact\_name;

}

void Vehicle::set\_number\_of\_cylinders(int num\_cylinders){

number\_of\_cylinders = num\_cylinders;

}

int Vehicle::get\_num\_of\_cylinders() const{

return number\_of\_cylinders;

}

Truck.h

#ifndef TRUCK\_H

#define TRUCK\_H

#include <string>

#include "Vehicle.h"

class Truck : public Vehicle{

protected:

double load\_capacity;

int towing\_capacity;

public:

Truck(); //Default Constructor

Truck(std::string themanufactname, int thenumofcylinders, double theloadcapacity, int thetowingcapacity); //Parameterized Constructor

Truck(const Truck& T); // Copy Constructors

//Setter

void setloadcap(double theloadcap);

void settowingcap(int thetowingcap);

//Getter

double getloadcap() const;

int gettowingcap() const;

//Print details and operator overloading

void printTruckDetails(const Person &P);

Truck& operator = (const Truck& rhs);

};

#endif

Truck.cpp

#include "Truck.h"

Truck::Truck() : Vehicle(), load\_capacity(0), towing\_capacity(0){

//Default Constructors

}

Truck::Truck(std::string themanufactname, int thenumofcylinders, double theloadcapacity, int thetowingcapacity) : Vehicle(themanufactname, thenumofcylinders), load\_capacity(theloadcapacity), towing\_capacity(thetowingcapacity){

//Parameterized Constructors

}

Truck::Truck(const Truck& T){

//Copy Constructors

manufact\_name = T.get\_manufact\_name();

number\_of\_cylinders = T.get\_num\_of\_cylinders();

load\_capacity = T.getloadcap();

towing\_capacity = T.gettowingcap();

}

//Operator "=" overloading

Truck& Truck::operator = (const Truck& rhs){

manufact\_name = rhs.get\_manufact\_name();

number\_of\_cylinders = rhs.get\_num\_of\_cylinders();

load\_capacity = rhs.getloadcap();

towing\_capacity = rhs.gettowingcap();

return \*this;

}

//Setter

void Truck :: setloadcap(double theloadcap){

load\_capacity = theloadcap;

}

void Truck :: settowingcap(int thetowingcap){

towing\_capacity = thetowingcap;

}

//Getter

double Truck :: getloadcap() const{

return load\_capacity;

}

int Truck :: gettowingcap() const{

return towing\_capacity;

}

void Truck :: printTruckDetails(const Person &P){

std::cout << "---------------- Truck Information And Output -----------------------" << std::endl;

std::cout << "The Owner of the Truck is: " << getownerName(P)<< std::endl;

std::cout << "The Truck's Manufacturer is: " << get\_manufact\_name() << std::endl;

std::cout << "The number of cylinders of the Truck is: " << get\_num\_of\_cylinders() << std::endl;

std::cout << "The loading capacity of the Truck is: " << getloadcap() << std::endl;

std::cout << "The towing capacity of the Truck is: " << gettowingcap() << std::endl;

std::cout << "---------------------------------------------------------------------" << std::endl;

}

Main.cpp

#include "Person.h"

#include "Truck.h"

int main(){

Vehicle V;

//Entering Truck's detail

Truck T("Honda", 15, 1000, 1500);

Truck T1;

//Setting Vehicle's detail

V.set\_number\_of\_cylinders(10);

V.set\_manufact\_name("Toyota");

//Enterting Person's information

Person P;

std::cin >> P;

//Getting information from Vehicle

std::cout << "The owner's name is: " << V.getownerName(P) << std::endl;

std::cout << "The Manufacturer name is: " << V.get\_manufact\_name() << std::endl;

std::cout << "The number of cylinders is: " << V.get\_num\_of\_cylinders() <<

std::endl;

//Getting information from Trucks

T.printTruckDetails(P);

//Testing operator "="

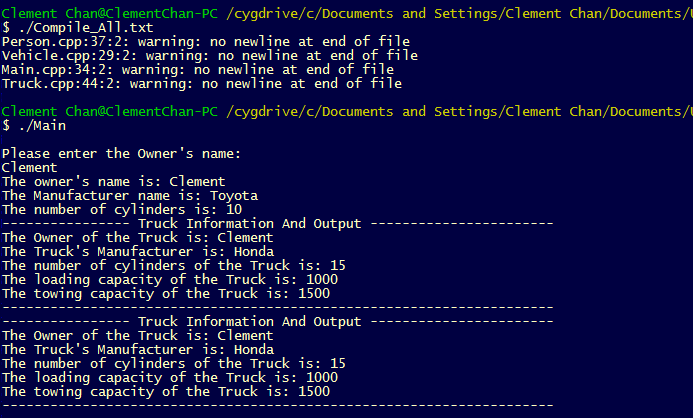
T1 = T;

T1.printTruckDetails(P);

return 0;

}

**Output**

****

Compile\_All.txt:

g++ -c Person.cpp && g++ -c Vehicle.cpp && g++ -c Main.cpp && g++ -c Truck.cpp && g++ -o Main Person.o Main.o Vehicle.o Truck.o

**Question 6:**

Payment.h

#ifndef PAYMENT\_H

#define PAYMENT\_H

#include <string>

#include <iostream>

#include <cstdlib>

class Payment{

protected:

float payment\_amount;

public:

//Constructors

Payment(); // Default Constructor

Payment(float payamount);

//Setter and Getter function

void setpayment(float payment);

float getpayment();

//Output payment details

void paymentDetails();

};

#endif

Payment.cpp

#include "Payment.h"

Payment :: Payment() : payment\_amount(0){

//default constructor

}

Payment :: Payment(float payamount) : payment\_amount(payamount){

//parameterized constructor

}

//Setter Function

void Payment :: setpayment(float payment){

payment\_amount = payment;

}

//Getter Function

float Payment :: getpayment(){

return payment\_amount;

}

//Payment Details Function

void Payment :: paymentDetails(){

std::cout << "The amount paid is: " << getpayment() << std::endl;}

CashPayment.h

#ifndef CASHPAYMENT\_H

#define CASHPAYMENT\_H

#include <string>

#include <iostream>

#include <cstdlib>

#include "Payment.h"

class CashPayment : public Payment {

public:

//Constructors

CashPayment(); // Default Constructor

CashPayment(float payamount);

//Output payment details

void paymentDetails();

};

#endif

CashPayment.cpp

#include "CashPayment.h"

CashPayment :: CashPayment() : Payment(){

//default constructor

}

CashPayment :: CashPayment(float payamount) : Payment(payamount) {

//parameterized constructor

}

//Payment Details Function

void CashPayment :: paymentDetails(){

std::cout << "The amount paid in cash is: " << getpayment() << std::endl;

}

CreditPayment.h

#ifndef CREDITPAYMENT\_H

#define CREDITPAYMENT\_H

#include <string>

#include <iostream>

#include <cstdlib>

#include "Payment.h"

class CreditPayment : public Payment {

public:

//Constructors

CreditPayment(); // Default Constructor

CreditPayment(float payamount);

//Output payment details

void paymentDetails();

};

#endif

CreditPayment.cpp

#include "CreditPayment.h"

CreditPayment :: CreditPayment() : Payment(){

//default constructor

}

CreditPayment :: CreditPayment(float payamount) : Payment(payamount) {

//parameterized constructor

}

//Payment Details Function

void CreditPayment :: paymentDetails(){

std::cout << "The amount paid in credit is: " << getpayment() << std::endl;

}

Main.cpp

#include "Payment.h"

#include "CashPayment.h"

#include "CreditPayment.h"

#include <iostream>

int main(){

//Testing Base Class -- Payment

Payment P(20.5);

P.paymentDetails();

//Testing Derived Class -- CashPayment

CashPayment CP(30.5), CP1(4000.35);

CP.paymentDetails();

CP1.paymentDetails();

//Testing Derived Class -- CreditPayment

CreditPayment CrP(500.125), CrP1(3000.5);

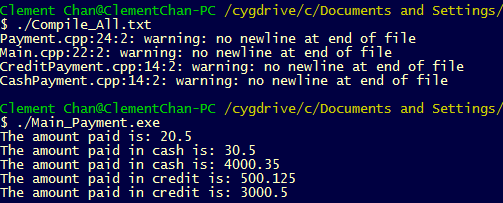
CrP.paymentDetails();

CrP1.paymentDetails();

return 0;

}

**Output**

****

Compile\_All.txt:

g++ -c Payment.cpp && g++ -c Main.cpp && g++ -c CreditPayment.cpp && g++ -c CashPayment.cpp && g++ -o Main\_Payment Payment.o Main.o CashPayment.o CreditPayment.o