**University of Michigan – Dearborn**

**CIS 200 – Computer Science 2**

**Project 02**

Nahrin Sharna

[nsharna@umich.edu](mailto:nsharna@umich.edu)

Date: 03/11/2019

# Source Code:

## Employee Header – Base Class(Employee.h):

#include<string>

#pragma once

using namespace std;

class Employee {

protected:

string first, last;

string SSN, EN;

int statusMethod;

public:

Employee();

Employee(string f, string l, string ssn, string en);

string getFirst();

void setFirst(string f);

string getLast();

void setLast(string l);

string getSSN();

int setSSN(string ssn);

string getEN();

int setEN(string en);

int getstatusMethod();

void print();

};

## Employee.cpp:

#include "Employee.h"

#include<iostream>

#include<string>

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: To initialize variable first name, last name, social security number, employee number and statusMethod to its default value

\*/

Employee::Employee() {

string first = "";

string last = "";

string SSN = "000-00-0000";

string EN = "000-X";

int statusMethod = 0;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To store input parameter value to the declared variable respectively

\*/

Employee::Employee(string f, string l, string ssn, string en) {

first = f;

last = l;

SSN = ssn;

EN = en;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date:03/05/2019

Purpose: To set the string f in the first name variable

\*/

void Employee::setFirst(string f) {

first = f;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To return the stored value of first name

\*/

string Employee::getFirst() {

return first;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To set the string l to the variable last name

\*/

void Employee::setLast(string l) {

last = l;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To return the stored value of last name

\*/

string Employee::getLast() {

return last;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: When input is valid, set the string ssn to social security variable SSN, and set the statusmethod to zero

Else, when the input is invalid, set the statusMethod to 1

Return statusMethod

\*/

int Employee::setSSN(string ssn) {

if (ssn.length() == 11 && (ssn.at(3) == '-') && (ssn.at(6) == '-') && isdigit(ssn.at(0)) && isdigit(ssn.at(1))

&& isdigit(ssn.at(2)) && isdigit(ssn.at(4)) && isdigit(ssn.at(5)) && isdigit(ssn.at(7)) &&

isdigit(ssn.at(8)) && isdigit(ssn.at(9)) && isdigit(ssn.at(10)))

{

SSN = ssn;

statusMethod = 0;

}

else {

statusMethod = 1;

}

return statusMethod;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/07/2019

Modification Date: 03/07/2019

Purpose: return the stored value of statusMethod

\*/

int Employee::getstatusMethod() {

return statusMethod;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date:03/05/2019

Purpose: To return the stored value of SSN

\*/

string Employee::getSSN() {

return SSN;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: When input is valid, set the string en to employee number variable EN, and set the statusMethod to zero

Else, when the input is invalid, set the statusMethod to 2

Return statusMethod

\*/

int Employee::setEN(string en) {

if (en.length() == 5 && en.at(3) == '-' && isdigit(en[0]) && isdigit(en[1]) && isdigit(en[2]) && en[4] >= 'A' && en[4] <= 'M')

{

EN = en;

statusMethod = 0;

}

else {

statusMethod = 2;

}

return statusMethod;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To return the stored value of EN

\*/

string Employee::getEN() {

return EN;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To print the first name, last name, last 4 digits of social security and employee number

\*/

void Employee::print() {

cout << "First Name: " << first << endl;

cout << "Last Name: " << last << endl;

cout << "Social Security: XXX-XX-" << SSN[7] << SSN[8] << SSN[9] << SSN[10] << endl;

cout << "Employee Number: " << EN << endl;

}

## SalaryEmployeePay.h :

#pragma once

#include"Employee.h"

class SalaryEmployeePay : public Employee { //child class of Employee

private:

float AnnualPay;

float WeeklyPay;

int taxCode;

public:

SalaryEmployeePay();

SalaryEmployeePay(float ann, float wk, int tax);

int setAnnualPay(float ann);

float getAnnualPay();

float getWeeklyPay();

int setTaxCode(int tax);

int getTaxCode();

## };

## SalaryEmployeePay.cpp:

#include"SalaryEmployeePay.h"

#include"Employee.h"

#include<iostream>

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: To initialize variable AnnualPay, WeeklyPay, taxCode to its default value

\*/

SalaryEmployeePay::SalaryEmployeePay()

{

AnnualPay = 0.0;

WeeklyPay = 0.0;

taxCode = 0.0;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: To store input parameter value to the declared variables repectively

\*/

SalaryEmployeePay::SalaryEmployeePay(float ann, float wk, int tax)

{

AnnualPay = ann;

WeeklyPay = wk;

taxCode = tax;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: If the input parameter is valid, store the input value to AnnualPay, set statusMethod to zero

Else, when the input is invalid, set the statusMethod to 3

return statusMethod

\*/

int SalaryEmployeePay::setAnnualPay(float ann)

{

if (ann > 0) {

AnnualPay = ann;

statusMethod = 0;

}

else {

statusMethod = 3;

}

return statusMethod;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: To return the stored value of AnnualPay

\*/

float SalaryEmployeePay::getAnnualPay()

{

return AnnualPay;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: If the input parameter is valid, store the input value to taxCode, set statusMethod to zero

Else, when the input is invalid, set the statusMethod to 4

return statusMethod

\*/

int SalaryEmployeePay::setTaxCode(int tax)

{

if (tax >= 1 && tax <= 3) {

taxCode = tax;

statusMethod = 0;

}

else {

statusMethod = 4;

}

return statusMethod;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: Return the stored value of taxCode

\*/

float SalaryEmployeePay::getTaxCode()

{

return taxCode;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: Return the weekly pay of employee based on different taxCode

\*/

float SalaryEmployeePay::getWeeklyPay()

{

if (taxCode == 1)

WeeklyPay = (AnnualPay - (AnnualPay \* 0.25)) / 52;

else if (taxCode == 2)

WeeklyPay = (AnnualPay - (AnnualPay \* 0.20)) / 52;

else if(taxCode == 3)

WeeklyPay = (AnnualPay - (AnnualPay \* 0.15)) / 52;

return WeeklyPay;

}

## HourlyEmployee.h:

#pragma once

#include "Employee.h"

class HourlyEmployee : public Employee { //child class of Employee

protected:

float payRate;

float hours;

public:

HourlyEmployee();

HourlyEmployee(float p, float h);

int setPayRate(float p);

float getPayRate();

int setHours(float h);

float getHours();

};

## HourlyEmployee.cpp:

#include "HourlyEmployee.h"

#include<iostream>

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To initialize the variables for employee's hours and pay rate to their default value

\*/

HourlyEmployee::HourlyEmployee()

{

payRate = 0.0;

hours = 0.0;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To store the input parameter value to the decalred varibales repectively

\*/

HourlyEmployee::HourlyEmployee(float p, float h)

{

payRate = p;

hours = h;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: If the input parameter is valid, store the input value to payRate, set statusMethod to zero

Else, when the input is invalid, set the statusMethod to 5

return statusMethod

\*/

int HourlyEmployee::setPayRate(float p)

{

if (p >= 10.00 && p <= 75.00) {

payRate = p;

statusMethod = 0;

}

else {

statusMethod = 5;

}

return statusMethod;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To return the stored value of the payRate

\*/

float HourlyEmployee::getPayRate()

{

return payRate;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: If the input parameter is valid, store the input value to hours, set statusMethod to zero

Else, when the input is invalid, set the statusMethod to 6

return statusMethod

\*/

int HourlyEmployee::setHours(float h)

{

if (h >= 0.0 && h <= 60.0) {

hours = h;

statusMethod = 0;

}

else {

statusMethod = 6;

}

return statusMethod;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To return the stored value of hours

\*/

float HourlyEmployee::getHours()

{

return hours;

}

## HourlyEmloyeePay.h:

#pragma once

#include "HourlyEmployee.h"

class HourlyEmployeePay : public HourlyEmployee { //child class of HourlyEmployee

private:

float overTymPay;

int taxCode;

char status;

public:

HourlyEmployeePay();

HourlyEmployeePay(float op, int tax, char s);

int setTaxCode(int tax);

int getTaxCode();

int setStatus(char s);

char getStatus();

float getOverTymPay();

float getcalHourlyPay();

};

## HourlyEmployeePay.cpp:

#include "HourlyEmployee.h"

#include <iostream>

#include "HourlyEmployeePay.h"

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To initialize the variables overTymPay and taxCode to its null value

\*/

HourlyEmployeePay::HourlyEmployeePay()

{

overTymPay = 0.0;

taxCode = 0;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To store the input parameter value to the decalred varibales repectively

\*/

HourlyEmployeePay::HourlyEmployeePay(float op, int tax, char s)

{

overTymPay = op;

taxCode = tax;

status = s;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: If the input parameter is valid, store the input value to taxCode, set statusMethod to zero

Else, when the input is invalid, set the statusMethod to 7

return statusMethod

\*/

int HourlyEmployeePay::setTaxCode(int tax)

{

if (tax >= 1 && tax <= 3) {

taxCode = tax;

statusMethod = 0;

}

else {

statusMethod = 7;

}

return statusMethod;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To return the stored value of taxCode

\*/

int HourlyEmployeePay::getTaxCode()

{

return taxCode;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/07/2019

Purpose: If the input parameter is valid, store the input value to status, set statusMethod to zero

Else, when the input is invalid, set the statusMethod to 8

return statusMethod

\*/

int HourlyEmployeePay::setStatus(char s)

{

if ((s == 'F') || (s == 'P')) {

status = s;

statusMethod = 0;

}

else {

statusMethod = 8;

}

return statusMethod;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To return the stored value of status

\*/

char HourlyEmployeePay::getStatus()

{

return status;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To return the value of 1.5 times payRate

\*/

float HourlyEmployeePay::getOverTymPay()

{

return payRate \* 1.5;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: When total hours is equal or less than 40, calculate netwage with regular payRate considering different taxCode

When total hours is greater than 40, calculate netwage considering 1.5 times of payRate for extra hours

return the netwage

\*/

float HourlyEmployeePay::getcalHourlyPay()

{

float grosswage, netwage;

if (hours <= 40) {

if (taxCode == 1)

netwage = hours \* payRate\*(1 - 0.25);

else if(taxCode == 2)

netwage = hours \* payRate\*(1 - 0.20);

else if(taxCode == 3)

netwage = hours \* payRate\*(1 - 0.15);

return netwage;

}

else if (hours > 40) {

if (taxCode == 1) {

grosswage = ((40\*payRate) + ((hours - 40)\*payRate\*1.5));

netwage = grosswage - (grosswage\*0.25);

}

else if (taxCode == 2) {

grosswage = ((40\*payRate) + ((hours - 40)\*payRate\*1.5));

netwage = grosswage - (grosswage\*0.20);

}

else if (taxCode == 3) {

grosswage = ((40\*payRate) + ((hours - 40)\*payRate\*1.5));

netwage = grosswage - (grosswage\*0.15);

}

return netwage;

}

}

## AgencyEmployeePay.h:

#pragma once

#include"HourlyEmployee.h"

class AgencyEmployeePay: public HourlyEmployee { //child class of HourlyEmployee

private:

string company;

public:

AgencyEmployeePay();

AgencyEmployeePay(string c);

void setCompany(string c);

string getCompany();

};

## AgencyEmployeePay.cpp:

#include "AgencyEmployeePay.h"

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To initialize company variable to its default value

\*/

AgencyEmployeePay::AgencyEmployeePay()

{

company = "";

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To store the input parameter value to the decalred varibale

\*/

AgencyEmployeePay::AgencyEmployeePay(string c)

{

company = c;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To set the parameter value to variable company

\*/

void AgencyEmployeePay::setCompany(string c)

{

company = c;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/05/2019

Purpose: To return the stored value of compnay

\*/

string AgencyEmployeePay::getCompany()

{

return company;

}

## Main Function (Source.cpp):

/\*

Author: Nahrin Sharna

Creation Date: 03/05/2019

Modification Date: 03/09/2019

Purpose: To store employee information and display employee's pay information

\*/

#include<iostream>

#include<string>

#include<fstream>

#include"Employee.h"

#include "SalaryEmployeePay.h"

#include"HourlyEmployee.h"

#include"HourlyEmployeePay.h"

#include"AgencyEmployeePay.h"

using namespace std;

int main() {

ofstream outs;

outs.open("pay.dat");

string first, last, SSN, EN, name;

float annualPay, wkPay, payRate, Hours;

int tax, tax2;

char s;

Employee emp1;

SalaryEmployeePay sal1;

HourlyEmployee info1;

HourlyEmployeePay calInfo1;

AgencyEmployeePay name1;

cout << " ----------------------------------------------------------------------------------------------" << endl;

cout << " Testing for Employee Class " << endl;

cout << " ----------------------------------------------------------------------------------------------" << endl;

outs << " Testing for Employee Class " << endl;

outs << "Employee Information: " << endl;

cout << "Please enter your first name: " << endl;

cin >> first;

emp1.setFirst(first);

cout << "First Name: " << emp1.getFirst() << endl;

outs << "First Name: " << emp1.getFirst() << endl;

cout << "Please enter your last name: " << endl;

cin >> last;

emp1.setLast(last);

cout << "Last Name: " << emp1.getLast() << endl;

outs << "Last Name: " << emp1.getLast() << endl;

cout << "Please enter your social security number in the format of(XXX-XX-XXXX): ";

cin >> SSN;

emp1.setSSN(SSN);

//statusMethod =1 means the SSN is not in a right form carrying digits from 0-9

while (emp1.getstatusMethod() == 1) {

cout << "Sorry!! Invalid Information." << endl;

cout << "Please enter your social security number in the format of(XXX-XX-XXXX): ";

cin >> SSN;

emp1.setSSN(SSN);

}

cout << "Social Security is: XXX-XX-" << SSN[7] << SSN[8] << SSN[9] << SSN[10] << endl;

outs << "Social Security is: XXX-XX-" << SSN[7] << SSN[8] << SSN[9] << SSN[10] << endl;

cout << "Please enter you employee number(XXX-L): " << endl;

cout << "First 3 characters should be digits(0-9) and last one should be letter(A-M)" << endl;

cin >> EN;

emp1.setEN(EN);

//statusMethod = 2 means Employee Number is not valid: first 3 are not digits or last one is not letter(A-M)

while (emp1.getstatusMethod() == 2) {

cout << "Sorry!! Invalid Information." << endl;

cout << "Please enter you employee number(XXX-L): " << endl;

cout << "First 3 characters should be digits(0-9) and last one should be letter(A-M)" << endl;

cin >> EN;

emp1.setEN(EN);

}

cout << "Employee Number: " << emp1.getEN() << endl;

outs << "Employee Number: " << emp1.getEN() << endl;

emp1.print(); // print the information stored in emp1 object of Employee class

cout << " ----------------------------------------------------------------------------------------------" << endl;

cout << " Testing for SalaryEmployeePay Class " << endl;

cout << " ----------------------------------------------------------------------------------------------" << endl;

outs << " Testing for SalaryEmployeePay Class " << endl;

outs << "Pay information: " << endl;

cout << "Please enter your annual pay: " << endl;

cin >> annualPay;

sal1.setAnnualPay(annualPay);

//statusMethod 3 means inout annual value is negative

while (sal1.getstatusMethod() == 3) {

cout << "Sorry!! Negative numbers can not be accepted." << endl;

cout << "Please enter your annual pay: " << endl;

cin >> annualPay;

sal1.setAnnualPay(annualPay);

}

cout << "Annual Pay is: " << sal1.getAnnualPay() << endl;

outs << "Annual Pay is: " << sal1.getAnnualPay() << endl;

cout << "Please enter your tax code(1 for 25 % , 2 for 20 % , 3 for 15 % ): " << endl;

cin >> tax;

sal1.setTaxCode(tax);

//statusMethod 4 means taxCode is not within the range if 1-3

while (sal1.getstatusMethod() == 4) {

cout << "Sorry!! Invalid tax code." << endl;

cout << "Please enter your tax code(1 for 25 % , 2 for 20 % , 3 for 15 % ): " << endl;

cin >> tax;

sal1.setTaxCode(tax);

}

cout << "Tax Code: " << sal1.getTaxCode() << endl;

outs << "Tax Code: " << sal1.getTaxCode() << endl;

cout << "Your weekly pay is: " << sal1.getWeeklyPay() << endl;

outs << "Weekly pay is: " << sal1.getWeeklyPay() << endl;

cout << " ----------------------------------------------------------------------------------------------" << endl;

cout << " Testing for HourlyEmployee Class " << endl;

cout << " ----------------------------------------------------------------------------------------------" << endl;

outs << " Testing for HourlyEmployee Class " << endl;

outs << "Employee information: " << endl;

cout << "Please enter your first name: " << endl;

cin >> first;

info1.setFirst(first);

cout << "First Name: " << info1.getFirst() << endl;

outs << "First Name: " << info1.getFirst() << endl;

cout << "Please enter your last name: " << endl;

cin >> last;

info1.setLast(last);

cout << "Last Name: " << info1.getLast() << endl;

outs << "Last Name: " << info1.getLast() << endl;

cout << "Please enter your social security number in the format of(XXX-XX-XXXX): ";

cin >> SSN;

info1.setSSN(SSN);

//statusMethod =1 means the SSN is not in a right form carrying digits from 0-9

while (info1.getstatusMethod() == 1) {

cout << "Sorry!! Invalid Information." << endl;

cout << "Please enter your social security number in the format of(XXX-XX-XXXX): ";

cin >> SSN;

info1.setSSN(SSN);

}

cout << "Social Security is: XXX-XX-" << SSN[7] << SSN[8] << SSN[9] << SSN[10] << endl;

outs << "Social Security is: XXX-XX-" << SSN[7] << SSN[8] << SSN[9] << SSN[10] << endl;

cout << "Please enter you employee number(XXX-L): " << endl;

cout << "First 3 characters should be digits(0-9) and last one should be letter(A-M)" << endl;

cin >> EN;

info1.setEN(EN);

//statusMethod = 2 means Employee Number is not valid: first 3 are not digits or last one is not letter(A-M)

while (info1.getstatusMethod() == 2) {

cout << "Sorry!! Invalid Information." << endl;

cout << "Please enter you employee number(XXX-L): " << endl;

cout << "First 3 characters should be digits(0-9) and last one should be letter(A-M)" << endl;

cin >> EN;

info1.setEN(EN);

}

cout << "Employee Number: " << info1.getEN() << endl;

outs << "Employee Number: " << info1.getEN() << endl;

info1.print();

cout << endl << endl;

cout << "Please enter your pay rate: " << endl;

cin >> payRate;

info1.setPayRate(payRate);

//statusMethod 5 means payRate is either less than 10 or greater than 75

while (info1.getstatusMethod() == 5) {

cout << "Sorry!! Pay Rate can not be less than $10.00 or greater than $75.00." << endl;

cout << "Please enter your pay rate: " << endl;

cin >> payRate;

info1.setPayRate(payRate);

}

cout << "Your Pay Rate is: " << info1.getPayRate() << endl;

outs << "Pay Rate: " << info1.getPayRate() << endl;

cout << "Please enter the hours you have worked: " << endl;

cin >> Hours;

info1.setHours(Hours);

//statusMethod 6 means input hours is negative or greater than 60

while (info1.getstatusMethod() == 6) {

cout << "Sorry!! Hours can not be negative or exceed 60." << endl;

cout << "Please enter the hours you have worked: " << endl;

cin >> Hours;

info1.setHours(Hours);

}

cout << "Your total hours are: " << info1.getHours() << endl;

outs << "Hours Worked: " << info1.getHours() << endl;

cout << " ----------------------------------------------------------------------------------------------" << endl;

cout << " Testing for HourlyEmployeePay Class " << endl;

cout << " ----------------------------------------------------------------------------------------------" << endl;

outs << " Testing for HourlyEmployeepay Class " << endl;

outs << "Pay Information: " << endl;

calInfo1.setHours(Hours);

calInfo1.setPayRate(payRate);

calInfo1.getHours();

calInfo1.getPayRate();

cout << "Your pay rate is: " << calInfo1.getPayRate() << endl;

outs << "Pay rate: " << calInfo1.getPayRate() << endl;

cout << "Your total hours is: " << calInfo1.getHours() << endl;

outs << "Total hours: " << calInfo1.getHours() << endl;

cout << "Please enter your employee status(F for fulltime, P for part time): " << endl;

cin >> s;

calInfo1.setStatus(s);

//statusMethod 8 means user input is not F or P

while (calInfo1.getstatusMethod() == 8) {

cout << "Sorry!! Invalid status information." << endl;

cout << "Please enter your employee status(F for fulltime, P for part time): " << endl;

cin >> s;

calInfo1.setStatus(s);

}

if (calInfo1.getStatus() == 'F') {

cout << "Your Work Status is: Full Time." << endl;

outs << "Work Status: Full Time. " << endl;

}

if (calInfo1.getStatus() == 'P') {

cout << "Your Work Status is: Part Time." << endl;

outs << "Work Status: Part Time. " << endl;

}

cout << "Please enter your tax code(1 for 25 % , 2 for 20 % , 3 for 15 % ): " << endl;

cin >> tax2;

calInfo1.setTaxCode(tax2);

//statusMethod 7 means taxCode is not within the range if 1-3

while (calInfo1.getstatusMethod() == 7) {

cout << "Sorry!! Invalid tax code." << endl;

cout << "Please enter your tax code(1 for 25 % , 2 for 20 % , 3 for 15 % ): " << endl;

cin >> tax;

calInfo1.setTaxCode(tax);

}

cout << "Your Tax Code is: " << calInfo1.getTaxCode() << endl;

outs << "Tax Code: " << calInfo1.getTaxCode() << endl;

cout << "Your hourly income is: " << calInfo1.getcalHourlyPay() << endl;

outs << "Hourly Income: " << calInfo1.getcalHourlyPay() << endl;

cout << " ----------------------------------------------------------------------------------------------" << endl;

cout << " Testing for AgencyEmployeePay Class " << endl;

cout << " ----------------------------------------------------------------------------------------------" << endl;

outs << " Testing for AgencyEmployeePay Class " << endl;

outs << "Employee's company information: " << endl;

cout << "Please enter the company name: " << endl;

cin >> name;

name1.setCompany(name);

cout << "The company to pay you is: " << name1.getCompany() << endl;

outs << "Company Name: " << name1.getCompany() << endl;

name1.setFirst(info1.getFirst());

name1.setLast(info1.getLast());

name1.setSSN(info1.getSSN());

name1.setEN(info1.getEN());

cout << "Your First Name: " << name1.getFirst() << endl;

outs << "First Name: " << name1.getFirst() << endl;

cout << "Your Last Name: " << name1.getLast() << endl;

outs << "Last Name: " << name1.getLast() << endl;

cout << "Your Social Security: XXX-XX-"<<SSN[7] << SSN[8] << SSN[9] <<SSN[10] << endl;

outs << "Social Security: " << name1.getSSN() << endl;

cout << "Your Employee Number: " << name1.getEN() << endl;

outs << "Employee Number: " << name1.getEN() << endl;

cout << "Thank you for using our software." << endl;

system("pause");

return 0;

}

# Initial Test Plan:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1. | Valid | User input first name | First Name =  “Nahrin” | First Name: Nahrin |  |  |
| 2. | Valid | User input last name | Last Name =  “Sharna” | Last Name: Sharna |  |  |
| 3. | Invalid | When the user input Social Security in the right format(XXX-XX-XXXX) | SSN = “123234567” | Sorry!! Invalid Information.  Please enter your social security number in the format of(XXX-XX-XXXX): |  |  |
| 4. | Invalid | When the length of SSN is not equal to 11 | SSN =  “123-34-456789” | Sorry!! Invalid Information.  Please enter your social security number in the format of(XXX-XX-XXXX): |  |  |
| 5. | Invalid | When the SSN is in the right format(XXX-XX-XXXX) but any of the X’s are not digits(0-9) | SSN =  “A23-B2-23c6” | Sorry!! Invalid Information.  Please enter your social security number in the format of(XXX-XX-XXXX): |  |  |
| 6. | Valid | When the SSN is in the right format, every X’s are digits (0-9) and length is 11 | SSN =  “123-23-4567” | Social Security: XXX-XX-4567 |  |  |
| 7. | Invalid | When the input employee number EN is not in right format (XXX-L) | EN = “123AM” | Sorry!! Invalid Information.  Please enter your employee number(XXX-L): |  |  |
| 8 | Invalid | When the EN is in right format but the last character is not between A to M | EN = “123-P” | Sorry!! Invalid Information.  Please enter your employee number(XXX-L): |  |  |
| 9 | Invalid | When the any of the first X’s in EN is not digit(0-9) | EN = “aB2-A” | Sorry!! Invalid Information.  Please enter your employee number(XXX-L): |  |  |
| 10 | Valid | When the EN is in right format (XXX-L), X’s are between 0-9 and L is between A to M | EN = “123-A” | Employee Number =  “123-A” |  |  |
| 11 | Invalid | When the annual salary is negative | Annual Pay =  -3000 | Sorry!! Negative numbers can not be accepted. |  |  |
| 12. | Valid | When the annual salary is positive | Annual Pay =  40000 | Annual Pay is: 40000 |  |  |
| 13. | Invalid | When the user’s tax code is not between 1 to 3 | Tax Code = -1 | Sorry!! Invalid tax code. |  |  |
| 14 | Valid | When the user’s tax code is between 1 to 3 | Tax Code = 1 | Tax Code: 1 |  |  |
| 15. | Valid | Calculate the weekly pay from Annual Pay when tax code is 1(25%). | Annual Pay = 40000  Tax code = 1 | Your weekly pay is: 576.923 |  |  |
| 16 | Valid | Calculate the weekly pay from Annual Pay when tax code is 2(20%). | Annual Pay = 40000  Tax code = 2 | Your weekly pay is: 615.385 |  |  |
| 17 | Valid | Calculate the weekly pay from Annual Pay when tax code is 3(15%). | Annual Pay = 40000  Tax code = 3 | Your weekly pay is:  653.846 |  |  |
| 18 | Valid | Store new information for another employee of type HourlyEmployee and print() to display | First: Tasnim  Last: Sharmin  SSN: 222-32-2342  EN: 123-D | First Name: Tasnim  Last Name: Sharmin  Social Security: XXX-XX-2342  Employee Number:123-D |  |  |
| 19. | Invalid | When user’s pay rate is less than 10 | Pay rate: 9 | Sorry!! Pay Rate can not be less than $10.00 or greater than $75.00 |  |  |
| 20. | Invalid | When user’s pay rate is greater than 75 | Pay rate: 76 | Sorry!! Pay Rate can not be less than $10.00 or greater than $75.00 |  |  |
| 21. | Valid | When the pay rate is between 10.00 to 75.00 | Pay rate: 15 | Your Pay Rate is: 15 |  |  |
| 22. | Invalid | When employee’s total hour is negative | Hours = -10 | Sorry!! Hours can not be negative or exceed 60 |  |  |
| 23 | Invalid | When employee’s total hour is more than 60 | Hours = 65 | Sorry!! Hours can not be negative or exceed 60 |  |  |
| 24. | Valid | When total hours are not negative and not greater than 60 | Hours = 55 | Your total hours are: 55 |  |  |
| 25. | Invalid | When employee’s input for status is not F or P | Status = g | Sorry!! Invalid status information |  |  |
| 26. | Valid | When employee’s input for status is F or P | Status = F | Your Work Status is: Full Time |  |  |
| 27. | Valid | Calculate hourly net income when tax code is 1 and hour is greater than 40 | Pay rate = 15  Hours = 55  Tax code = 1 | Your hourly income is:  703.125 |  |  |
| 28 | Valid | Calculate hourly net income when tax code is 2 and hour is greater than 40 | Pay rate = 15  Hours = 55  Tax code = 2 | Your hourly income is:  750 |  |  |
| 29 | Valid | Calculate hourly net income when tax code is 3 and hour is greater than 40 | Pay rate = 15  Hours = 55  Tax code = 3 | Your hourly income is:  796.875 |  |  |
| 30 | Valid | Calculate hourly net income when tax code is 1 and hour is not greater than 40 | Pay rate = 15  Hours = 35  Tax code = 1 | Your hourly income is: 393.75 |  |  |
| 31 | Valid | Calculate hourly net income when tax code is 2 and hour is not greater than 40 | Pay rate = 15  Hours = 35  Tax code = 2 | Your hourly income is: 420 |  |  |
| 32 | Valid | Calculate hourly net income when tax code is 3 and hour is not greater than 40 | Pay rate = 15  Hours = 35  Tax code = 3 | Your hourly income is: 446.25 |  |  |
| 33 | Valid | Employee enter the name of company | Company = “Ford” | Your company to pay you is: Ford |  |  |

# Final Test Plan:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1. | Valid | User input first name | First Name =  “Nahrin” | First Name: Nahrin | First Name: Nahrin | Pass |
| 2. | Valid | User input last name | Last Name =  “Sharna” | Last Name: Sharna | Last Name: Sharna | Pass |
| 3. | Invalid | When the user input Social Security in the right format(XXX-XX-XXXX) | SSN = “123234567” | Sorry!! Invalid Information.  Please enter your social security number in the format of(XXX-XX-XXXX): | Sorry!! Invalid Information.  Please enter your social security number in the format of(XXX-XX-XXXX): | Pass |
| 4. | Invalid | When the length of SSN is not equal to 11 | SSN =  “123-34-456789” | Sorry!! Invalid Information.  Please enter your social security number in the format of(XXX-XX-XXXX): | Sorry!! Invalid Information.  Please enter your social security number in the format of(XXX-XX-XXXX): | Pass |
| 5. | Invalid | When the SSN is in the right format(XXX-XX-XXXX) but any of the X’s are not digits(0-9) | SSN =  “A23-B2-23c6” | Sorry!! Invalid Information.  Please enter your social security number in the format of(XXX-XX-XXXX): | Sorry!! Invalid Information.  Please enter your social security number in the format of(XXX-XX-XXXX): | Pass |
| 6. | Valid | When the SSN is in the right format, every X’s are digits (0-9) and length is 11 | SSN =  “123-23-4567” | Social Security: XXX-XX-4567 | Social Security: XXX-XX-4567 | Pass |
| 7. | Invalid | When the input employee number EN is not in right format (XXX-L) | EN = “123AM” | Sorry!! Invalid Information.  Please enter your employee number(XXX-L): | Sorry!! Invalid Information.  Please enter your employee number(XXX-L): | Pass |
| 8 | Invalid | When the EN is in right format but the last character is not between A to M | EN = “123-P” | Sorry!! Invalid Information.  Please enter your employee number(XXX-L): | Sorry!! Invalid Information.  Please enter your employee number(XXX-L): | Pass |
| 9 | Invalid | When the any of the first X’s in EN is not digit(0-9) | EN = “aB2-A” | Sorry!! Invalid Information.  Please enter your employee number(XXX-L): | Sorry!! Invalid Information.  Please enter your employee number(XXX-L): | Pass |
| 10 | Valid | When the EN is in right format (XXX-L), X’s are between 0-9 and L is between A to M | EN = “123-A” | Employee Number =  “123-A” | Employee Number =  “123-A” | Pass |
| 11 | Invalid | When the annual salary is negative | Annual Pay =  -3000 | Sorry!! Negative numbers can not be accepted. | Sorry!! Negative numbers can not be accepted. | Pass |
| 12. | Valid | When the annual salary is positive | Annual Pay =  40000 | Annual Pay is: 40000 | Annual Pay is: 40000 | Pass |
| 13. | Invalid | When the user’s tax code is not between 1 to 3 | Tax Code = -1 | Sorry!! Invalid tax code. | Sorry!! Invalid tax code. | Pass |
| 14 | Valid | When the user’s tax code is between 1 to 3 | Tax Code = 1 | Tax Code: 1 | Tax Code: 1 | Pass |
| 15. | Valid | Calculate the weekly pay from Annual Pay when tax code is 1(25%). | Annual Pay = 40000  Tax code = 1 | Your weekly pay is: 576.923 | Your weekly pay is: 576.923 | Pass |
| 16 | Valid | Calculate the weekly pay from Annual Pay when tax code is 2(20%). | Annual Pay = 40000  Tax code = 2 | Your weekly pay is: 615.385 | Your weekly pay is: 615.385 | Pass |
| 17 | Valid | Calculate the weekly pay from Annual Pay when tax code is 3(15%). | Annual Pay = 40000  Tax code = 3 | Your weekly pay is:  653.846 | Your weekly pay is:  653.846 | Pass |
| 18. | Valid | Store new information for another employee of type HourlyEmployee and print() to display | First: Tasnim  Last: Sharmin  SSN: 222-32-2342  EN: 123-D | First Name: Tasnim  Last Name: Sharmin  Social Security: XXX-XX-2342  Employee Number:123-D | First Name: Tasnim  Last Name: Sharmin  Social Security: XXX-XX-2342  Employee Number:123-D | Pass |
| 19. | Invalid | When user’s pay rate is less than 10 | Pay rate: 9 | Sorry!! Pay Rate can not be less than $10.00 or greater than $75.00 | Sorry!! Pay Rate can not be less than $10.00 or greater than $75.00 | Pass |
| 20. | Invalid | When user’s pay rate is greater than 75 | Pay rate: 76 | Sorry!! Pay Rate can not be less than $10.00 or greater than $75.00 | Sorry!! Pay Rate can not be less than $10.00 or greater than $75.00 | Pass |
| 21. | Valid | When the pay rate is between 10.00 to 75.00 | Pay rate: 15 | Your Pay Rate is: 15 | Your Pay Rate is: 15 | Pass |
| 22. | Invalid | When employee’s total hour is negative | Hours = -10 | Sorry!! Hours can not be negative or exceed 60 | Sorry!! Hours can not be negative or exceed 60 | Pass |
| 23. | Invalid | When employee’s total hour is more than 60 | Hours = 65 | Sorry!! Hours can not be negative or exceed 60 | Sorry!! Hours can not be negative or exceed 60 | Pass |
| 24. | Valid | When total hours are not negative and not greater than 60 | Hours = 55 | Your total hours are: 55 | Your total hours are: 55 | Pass |
| 25. | Invalid | When employee’s input for status is not F or P | Status = g | Sorry!! Invalid status information | Sorry!! Invalid status information | Pass |
| 26. | Valid | When employee’s input for status is F or P | Status = F | Your Work Status is: Full Time | Your Work Status is: Full Time | Pass |
| 27. | Valid | Calculate hourly net income when tax code is 1 and hour is greater than 40 | Pay rate = 15  Hours = 55  Tax code = 1 | Your hourly income is:  703.125 | Your hourly income is:  703.125 | Pass |
| 29 | Valid | Calculate hourly net income when tax code is 2 and hour is greater than 40 | Pay rate = 15  Hours = 55  Tax code = 2 | Your hourly income is:  750 | Your hourly income is:  750 | Pass |
| 29 | Valid | Calculate hourly net income when tax code is 3 and hour is greater than 40 | Pay rate = 15  Hours = 55  Tax code = 3 | Your hourly income is:  796.875 | Your hourly income is:  796.875 | Pass |
| 30 | Valid | Calculate hourly net income when tax code is 1 and hour is not greater than 40 | Pay rate = 15  Hours = 35  Tax code = 1 | Your hourly income is: 393.75 | Your hourly income is: 393.75 | Pass |
| 31 | Valid | Calculate hourly net income when tax code is 2 and hour is not greater than 40 | Pay rate = 15  Hours = 35  Tax code = 2 | Your hourly income is: 420 | Your hourly income is: 420 | Pass |
| 32 | Valid | Calculate hourly net income when tax code is 3 and hour is not greater than 40 | Pay rate = 15  Hours = 35  Tax code = 3 | Your hourly income is: 446.25 | Your hourly income is: 446.25 | Pass |
| 33 | Valid | Employee enter the name of company | Company = “Ford” | Your company to pay you is: Ford | Your company to pay you is: Ford | Pass |

# UML Diagram:

|  |
| --- |
| **Employee** |
| #first: string  #last: string  #SSN: string  #EN: string  #statusMethod: int |
| +Employee()  +Employee(f: string, l: string, ssn: string, en: string)  +getFirst(): string  +setFirst(f:string): void  +getLast(): string  +setLast(l: string): void  +getSSN(): string  +setSSN(ssn: string): string  +getEN(): string  +setEN(en: string): int  +getstatusMethod(): int  +print(): void |

|  |
| --- |
| **SalaryEmployeePay** |
| -AnnualPay: float  -WeeklyPay: float  -taxCode: int |
| +SalaryEmployeePay()  +SalaryEmployeePay(ann: float, wk: float, tax: int)  +setAnnualPay(ann: float): int  +getAnnualPay(): float  +getWeeklyPay(): float  +setTaxCode(tax: int): int  +getTaxCode(): int |

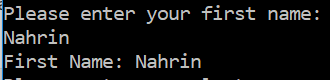
|  |
| --- |
| **HourlyEmployee** |
| #payRate: float  #hours: float |
| +HourlyEmployee()  +HourlyEmployee(p: float, h: float)  +setPayRate(p: float): int  +getPayRate(): float  +setHours(h: float): int  +getHours(): float |

|  |
| --- |
| **HourlyEmployeePay** |
| -overTymPay: float  -taxCode: int  -status: char |
| +HourlyEmployeePay()  +HourlyEmployeePay(op: float, tax: int, s: char)  +setTaxCode(tax: int): int  +getTaxCode(): int  +setStatus(s: char): int  +getStatus(): char  +getOverTymPay(): float  +getcalHourlyPay(): float |

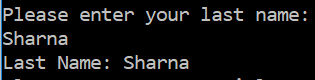
|  |
| --- |
| **AgencyEmployeePay** |
| -company: string |
| +AgencyEmployeePay()  + AgencyEmployeePay(c: string)  +setCompany(c: string): void  +getCompany(): string |

# Screenshots:

## Test Case 1:



## Test Case 2:



## Test Case 3:



## Test Case 4:



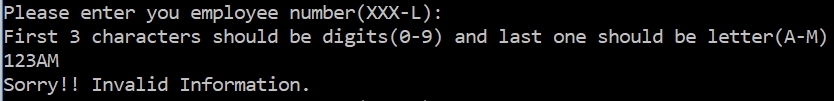
## Test Case 5:



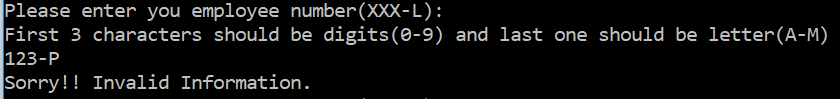
## Test Case 6:



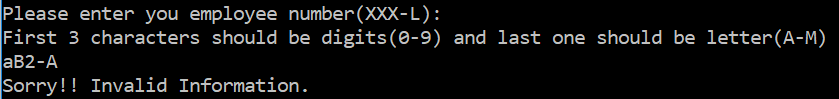
## Test Case 7:



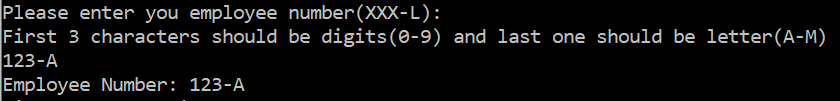
## Test Case 8:



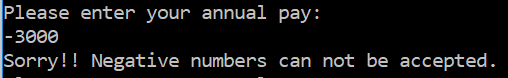
## Test Case 9:



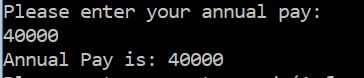
## Test Case 10:



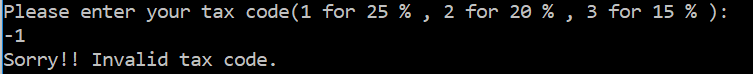
## Test Case 11:



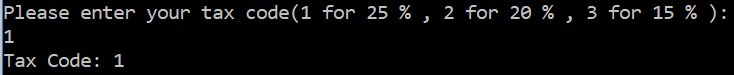
## Test Case 12:



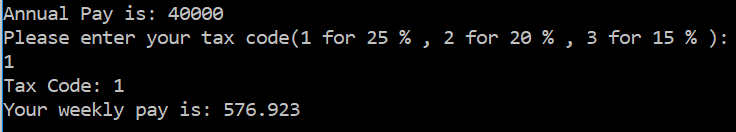
## Test Case 13:



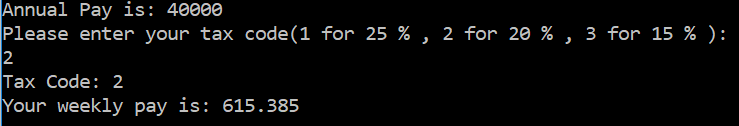
## Test Case 14:



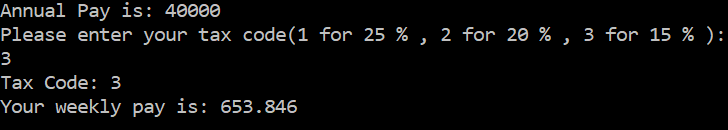
## Test Case 15:



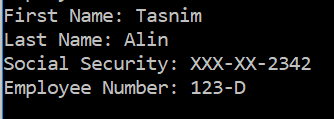
## Test Case 16:



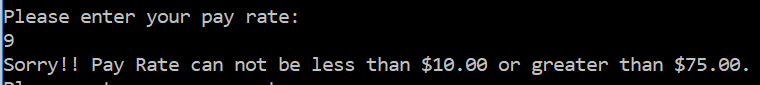
## Test Case 17:



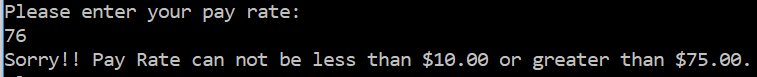
## Test Case 18:



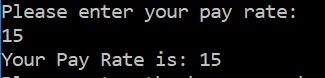
## Test Case 19:



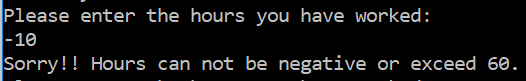
## Test Case 20:



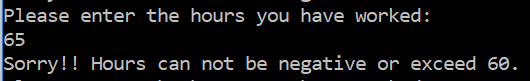
## Test Case 21



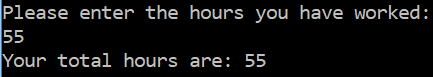
## Test Case 22:



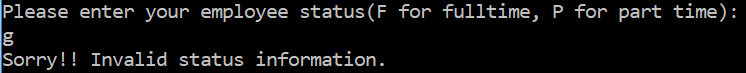
## Test Case 23:



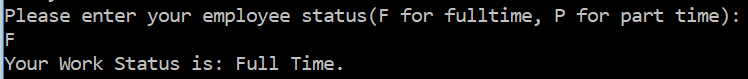
## Test Case 24:



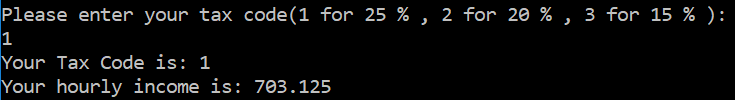
## Test Case 25:



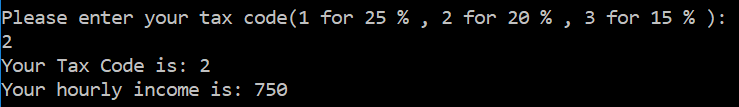
## Test Case 26:



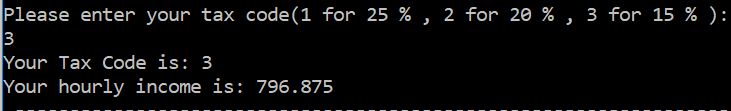
## Test Case 27:



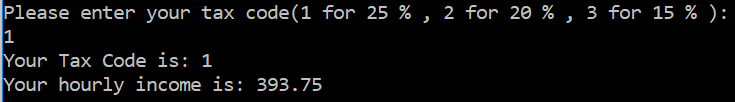
## Test Case 28:



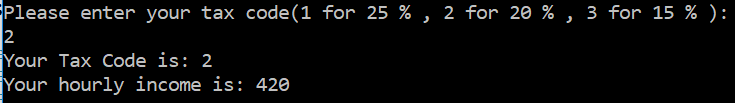
## Test Case 29:



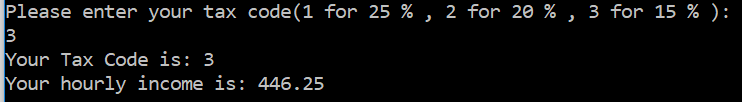
## Test Case 30:



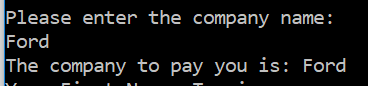
## Test Case 31:



## Test Case 32:



## Test Case 33:



**Output file “pay.dat” Screenshot:**

