**Screen Shot:**



**Code:**

// ---------------------------------------------------------------------------

//Program Header

//Program Name: Employee Class Integration

//Programmer: Renee Thomas

//CIS247, Week 6 Lab

//Program Description: Update Employee Class to inclued a Salaryied and Hourly employees

// ---------------------------------------------------------------------------

# include <iostream>

# include <string>

# include <iomanip>

using namespace std;

// create prototypes for functions

void DisplayApplicationInformation();

void DisplayDivider(string);

string getInput(string);

void TerminateApplication();

// create Benefits Class

class Benefits{

private:

string healthInsurance;

double lifeInsurance;

int vacation;

public:

Benefits();

Benefits(string hI, double lI, int vac);

void displayBenefits();

string getHealthInsurance();

void setHealthInsurance(string hI);

double getLifeInsurance();

void setLifeInsurance(double lI);

int getVacation();

void setVacation(int vac);

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Benefits Class Functions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Benefits::Benefits(){

healthInsurance = "Not provided";

lifeInsurance = 0.0;

vacation = 14;

}

Benefits::Benefits(string hI, double lI, int vac){

healthInsurance = hI;

lifeInsurance = lI;

vacation = vac;

}

void Benefits::displayBenefits(){

cout<<"Heath Insurance: \t"<<healthInsurance<<endl;

cout<<"Life Insurance: \t"<<lifeInsurance<<endl;

cout<<"Vacation: \t\t"<<vacation<<endl;

}

string Benefits::getHealthInsurance(){return healthInsurance;}

void Benefits::setHealthInsurance(string hI){healthInsurance = hI;}

double Benefits::getLifeInsurance(){return lifeInsurance;}

void Benefits::setLifeInsurance(double lI){lifeInsurance = lI;}

int Benefits::getVacation(){return vacation;}

void Benefits::setVacation(int vac){vacation = vac;}

// create Employee Class

class Employee

{

private:

static int numEmployees;

protected:

string firstName;

string lastName;

char gender;

int dependents;

double annualSalary;

Benefits bens;// create an instance of Benefits inside Employee class

public:

Employee();

Employee(string fn, string ln, char gen, int dep, double sal, Benefits bens\_in); // don't forget to use the class name as the data type!

virtual double calculatePay() = 0;

virtual void displayEmployee() = 0;

void displayNumEmployees();

string getFirstName();

void setFirstName(string fn);

string getLastName();

void setLastName(string ln);

char getGender();

void setGender(char gen);

int getDependents();

void setDependents(int dep);

double getAnnualSalary();

void setAnnualSalary(double sal);

static int getNumEmployees();

void setDependents(string dep);

void setAnnualSalary(string sal);

void setBenefits(Benefits);

Benefits getBenefits();

void setNumEmployees(int);

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Employee Class Functions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int Employee::numEmployees = 0;

// default constructor

Employee::Employee(): bens()

{

firstName = "Not Given";

lastName = "Not Given";

gender = 'U';

dependents = 0;

annualSalary = 20000;

bens = Benefits();

numEmployees ++;

}

// Employee constructor with multi arguements (remember to use class name as the data type when integrating into parameters)

Employee::Employee(string fn, string ln, char gen, int dep, double sal, Benefits bens\_in ): bens(bens\_in)

{

firstName = fn;

lastName = ln;

gender = gen;

dependents = dep;

annualSalary = sal;

bens = bens\_in;// set original instance to parameter instance

numEmployees ++;

}

/\*

// create function for calculatePay

double Employee:: calculatePay(){

double wsal = getAnnualSalary()/52;

return wsal;

}

// create function to display all employee information

void Employee::displayEmployee()

{

cout<<"\nEmployee Information"<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

cout<<"Employee Name: \t"<<firstName<<" "<<lastName<<endl;

cout<<"Gender: \t"<<gender<<endl;

cout<<"Dependents: \t"<<dependents<<endl;

cout<<"Annual Salary: \t"<< setprecision(2) << showpoint << fixed << annualSalary<<endl;

cout<<"Weekly Salary: \t"<< setprecision(2) << showpoint << fixed <<calculatePay()<<endl;

bens.displayBenefits();

cout<<endl<<endl;

}

\*/

void Employee::displayNumEmployees(){

cout<<"Number of employees entered:"<<getNumEmployees()<<endl;

}

// create getters and setters for all private variables

string Employee::getFirstName(){return firstName;}

void Employee::setFirstName(string fn){firstName=fn;}

string Employee::getLastName(){return lastName;}

void Employee::setLastName(string ln){lastName=ln;}

char Employee::getGender(){return gender;}

void Employee::setGender(char gen){gender=gen;}

int Employee::getDependents(){return dependents;}

void Employee::setDependents(int dep){dependents=dep;}

void Employee::setDependents(string dep){dependents= atoi(dep.c\_str());}

double Employee::getAnnualSalary(){return annualSalary;}

void Employee::setAnnualSalary(double sal){annualSalary=sal;}

void Employee::setAnnualSalary(string sal){annualSalary=atof(sal.c\_str());}

int Employee::getNumEmployees(){return numEmployees;}

void Employee::setBenefits(Benefits bn){bens = bn;}

Benefits Employee::getBenefits(){return bens;}

void Employee::setNumEmployees(int nE){numEmployees = nE;}

// create Salaried Class

class Salaried : public Employee{

private:

int managementLevel;

const int MIN\_MANAGEMENT\_LEVEL;

const int MAX\_MANAGEMENT\_LEVEL;

const double BONUS\_PERCENT;

public:

Salaried();

Salaried(string, string, char, int, double, Benefits, int);

Salaried(double sal, int ml);

double calculatePay();

void displayEmployee();

void setManagementLevel(int ml);

int getManagementLevel();

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Salaried Class Functions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Salaried::Salaried()

:

Employee(), MIN\_MANAGEMENT\_LEVEL(0), MAX\_MANAGEMENT\_LEVEL(3), BONUS\_PERCENT(10) {

managementLevel = 0;

}

Salaried::Salaried(string fn, string ln, char gen, int dep, double sal, Benefits ben, int ml)

:

Employee(fn, ln, gen, dep, sal, ben), MIN\_MANAGEMENT\_LEVEL(0), MAX\_MANAGEMENT\_LEVEL(3), BONUS\_PERCENT(10)

{

managementLevel = ml;

}

Salaried::Salaried(double sal, int ml)

:

Employee(), MIN\_MANAGEMENT\_LEVEL(0), MAX\_MANAGEMENT\_LEVEL(3), BONUS\_PERCENT(10)

{

annualSalary = sal;

managementLevel = ml;

}

double Salaried::calculatePay(){

double bonus = getManagementLevel()\*BONUS\_PERCENT/100;

double sal = getAnnualSalary() + bonus;

double weeklyPay= sal/52;

return weeklyPay;

}

void Salaried::displayEmployee(){

cout<<"\nEmployee Information"<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

cout<<"Employee Name: \t\t"<<firstName<<" "<<lastName<<endl;

cout<<"Gender: \t\t"<<gender<<endl;

cout<<"Dependents: \t\t"<<dependents<<endl;

cout<<"Annual Salary: \t\t"<< setprecision(2) << showpoint << fixed << annualSalary<<endl;

cout<<"Weekly Pay: \t\t"<< setprecision(2) << showpoint << fixed <<calculatePay()<<endl;

bens.displayBenefits();

cout<<"Employee Type:\t\tSalaried"<<endl;

cout<<"Management Level:\t"<<managementLevel<<endl;

cout<<endl<<endl;

}

void Salaried::setManagementLevel(int ml){

if (ml >= MIN\_MANAGEMENT\_LEVEL && ml <= MAX\_MANAGEMENT\_LEVEL)

managementLevel = ml;

else if (ml < MIN\_MANAGEMENT\_LEVEL)

managementLevel = MIN\_MANAGEMENT\_LEVEL;

else if (ml > MAX\_MANAGEMENT\_LEVEL)

managementLevel = MAX\_MANAGEMENT\_LEVEL;

}

int Salaried::getManagementLevel(){return managementLevel;}

// create Hourly Class

class Hourly : public Employee{

private:

double wage;

double hours;

string category;

const int MIN\_WAGE;

const int MAX\_WAGE;

const int MIN\_HOUR;

const int MAX\_HOUR;

public:

Hourly();

Hourly(double, double, string);

Hourly(string, string, char, int, double, double, Benefits, string);

double calculatePay();

void displayEmployee();

void setWage(double wg);

double getWage();

void setHours(double hrs);

double getHours();

void setCategory(string cat);

string getCategory();

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Hourly Class Functions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Hourly::Hourly():Employee(), MIN\_WAGE(10), MAX\_WAGE(75), MIN\_HOUR(0), MAX\_HOUR(50){

wage = 0.0;

hours = 0.0;

category = "X";

}

Hourly::Hourly(double wg, double hrs, string cat):Employee(), MIN\_WAGE(10), MAX\_WAGE(75), MIN\_HOUR(0), MAX\_HOUR(50){

wage = wg;

hours = hrs;

category = cat;

}

Hourly::Hourly(string fn, string ln, char gen, int dep, double wg, double hrs, Benefits ben, string cat)

:Employee(fn, ln, gen, dep, 0.0, ben), MIN\_WAGE(10), MAX\_WAGE(75), MIN\_HOUR(0), MAX\_HOUR(50)

{

wage= wg;

hours = hrs;

category = cat;

}

double Hourly::calculatePay(){

double wklyPy = wage \* hours;

double sal = wklyPy\*50;

setAnnualSalary(sal);

return wklyPy;

}

void Hourly::displayEmployee(){

cout<<"\nEmployee Information"<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

cout<<"Employee Name:\t\t"<<firstName<<" "<<lastName<<endl;

cout<<"Gender:\t\t\t"<<gender<<endl;

cout<<"Dependents:\t\t"<<dependents<<endl;

cout<<"Annual Salary:\t\t"<< setprecision(2) << showpoint << fixed << annualSalary<<endl;

cout<<"Weekly Salary:\t\t"<< setprecision(2) << showpoint << fixed <<calculatePay()<<endl;

bens.displayBenefits();

cout<<"Employee Type:\t\tHourly"<<endl;

cout<<"Category:\t\t"<<category<<endl;

cout<<"Wage:\t\t\t"<<wage<<endl;

cout<<"Hours:\t\t\t"<<hours<<endl;

cout<<endl<<endl;

}

void Hourly::setWage(double wg){

if (wg >= MIN\_WAGE && wg <= MAX\_WAGE)

wage = wg;

else if (wg < MIN\_WAGE)

wage = MIN\_WAGE;

else if (wg > MAX\_WAGE)

wage = MAX\_WAGE;

}

double Hourly::getWage(){return wage;}

void Hourly::setHours(double hrs){

if (hrs >= MIN\_HOUR && hrs <= MAX\_HOUR)

hours = hrs;

else if (hrs < MIN\_HOUR)

hours = MIN\_HOUR;

else if (hrs > MAX\_HOUR)

hours = MAX\_HOUR;

}

double Hourly::getHours(){return hours;}

void Hourly::setCategory(string cat){

if (category == "temporary" || category == "part time" || category == "full time")

category = cat;

else

category = "Unknown";

}

string Hourly::getCategory(){return category;}

int main(){

// call function for header information

DisplayApplicationInformation();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Start Program\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// call function for divider for start program

DisplayDivider("Start Program");

DisplayDivider("Employee 1");

// create pointers for Salaried and Hourly passing constuctors

Employee \*employee1 = new Salaried(10000, 3);

Employee \*employee2 = new Hourly(50, 40, "Full Time");

employee1->setFirstName(getInput("First Name: "));

employee1->setLastName(getInput("Last Name: "));

employee1->setGender(getInput("Gender: ")[0]);

employee1->setDependents(getInput("Number of Dependents: "));

string input;

// create instance of Benefits class to input into Salaried

Benefits emp1bens;

// set healthInsurance

emp1bens.setHealthInsurance(getInput("Health Insurance: "));

// set lifeInsurance

input = getInput("Life Insurance: ");

double lI = atof(input.c\_str());

emp1bens.setLifeInsurance(lI);

// set vacation

input = getInput("Vacation: ");

int vac = atoi(input.c\_str());

emp1bens.setVacation(vac);

// set Benefits object inside Employee class

employee1->setBenefits(emp1bens);

// call class function to display employee information

employee1->displayEmployee();

DisplayDivider("Employee 2");

employee2->setFirstName(getInput("First Name: "));

employee2->setLastName(getInput("Last Name: "));

employee2->setGender(getInput("Gender: ")[0]);

employee2->setDependents(getInput("Number of Dependents: "));

// create instance of Benefits class to input into Hourly

Benefits emp2bens;

// set healthInsurance

emp2bens.setHealthInsurance(getInput("Health Insurance: "));

// set lifeInsurance

input = getInput("Life Insurance: ");

lI = atof(input.c\_str());

emp2bens.setLifeInsurance(lI);

// set vacation

input = getInput("Vacation: ");

vac = atoi(input.c\_str());

emp2bens.setVacation(vac);

// set Benefits object inside Employee class

employee2->setBenefits(emp2bens);

// call class function to display employee information

employee2->displayEmployee();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End Program\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// call function to leave program

TerminateApplication();

// keep page open until user hits any key

cin.ignore();

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Non Class Functions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// create header function

void DisplayApplicationInformation(){

// create header to introduce the program

cout<<"\n--------------------------------------------"<<endl;

cout<<"| Welcome the Basic User Interface Program |"<<endl;

cout<<"| CIS247, Week 6 Lab |"<<endl;

cout<<"| Name: RENEE THOMAS |"<<endl;

cout<<"| This program adds a Salaried and Hourly |"<<endl;

cout<<"| class using pointers. |"<<endl;

}

// create divider string function

void DisplayDivider(string outputTitle){

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* "<<outputTitle<< " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl<<endl;

}

// create function to get information from user

string getInput(string inputType){

// declare variable strInput

string strInput;

// get information from user

cout<<"Enter your " <<inputType;

//put user input into variable strInput

getline(cin, strInput);

//output user input

return strInput;

}

// create good bye message

void TerminateApplication(){

cout<<"\n\nThank you for running the Employee class program for hourly and salaried employees."<<endl;

cout<<"Press any key to leave the program..."<<endl;

}