

1. Problem Statement

Program uses user inputs to calculate and display employee payments and related private information for a company.

2. Requirements

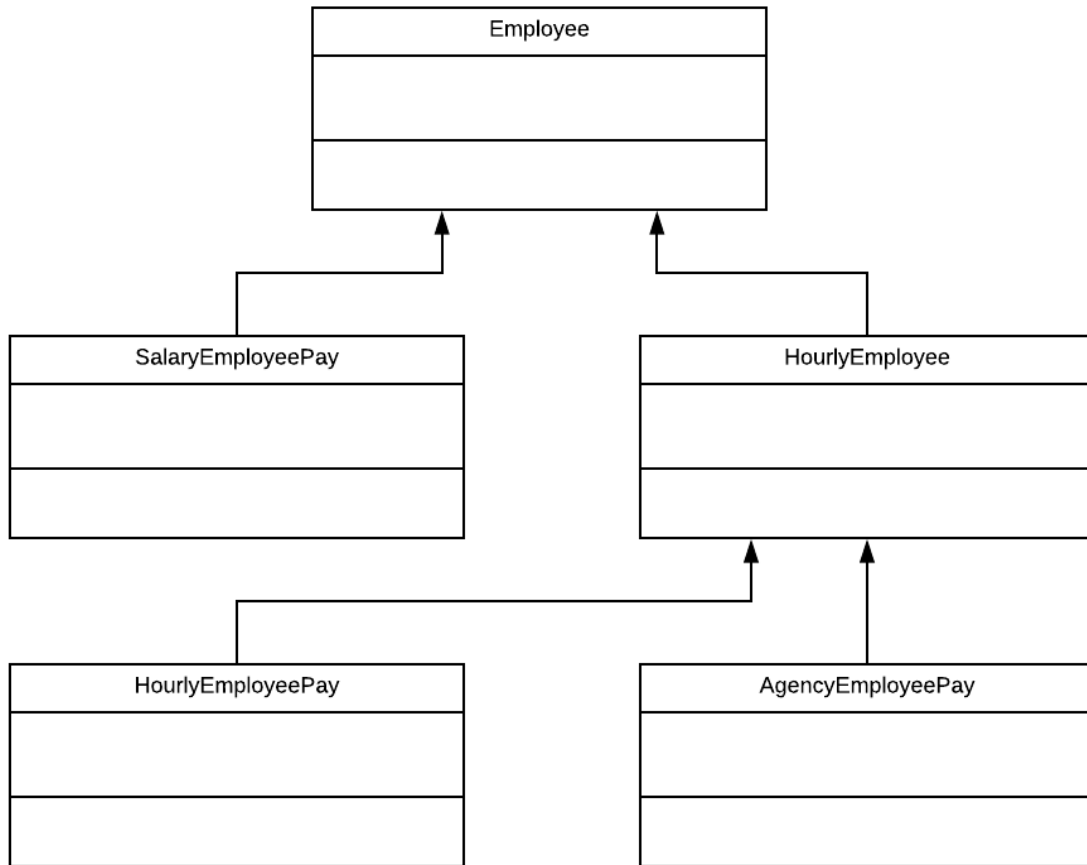
a. Assumptions

- i. User types in appropriate integer number to select an option from the menu
- ii. User types social security number without the dashes
- iii. User types employee number without the dashes
- iv. User types first and last name with space in the middle
- v. 40 hours per week for every employee
- vi. User types either 'F' for Full time work status or 'P' for Part time work status

b. Specifications

- i. Base class *Employee* has Employee name, Social Security Number, and Employee number stored in respective variables with a given format:
 1. Social Security Number format: xxx-xxx-xxxx (x as in number)
 2. Employee Number format: xxx-L (L as in letter)
- ii. Getter and setter functions for each private variables in each class
- iii. Print method to print each data in each class
- iv. Constructor and destructor to be added in each class
- v. Class *SalaryEmployeePay*, derived from *Employee* class, has following data:
 1. Annual pay
 2. Weekly pay – Calculated from annual pay
 3. Tax rate code (1 = 0.25, 2 = 0.20, 3 = 0.15)
- vi. Class *HourlyEmployee*, derived from *Employee* class, has following data:
 1. Hourly pay rate – based on weekly pay, assume a 40-hour work week
 2. # of hours worked
- vii. Class *HourlyEmployeePay*, derived from *HourlyEmployee* class, has following data:
 1. Overtime pay rate – Calculated from *HourlyEmployee* hourly pay rate
 2. Tax rate code (1 = 0.25, 2 = 0.20, 3 = 0.15)
 3. Work Status (F = Full time, P = Part time)
- viii. Class *AgencyEmployeePay*, derived from *HourlyEmployee* class, has a company name
- ix. Output file “pay.dat” to store sample data for various employee and the calculations

3. UML Classes Design Part 1:

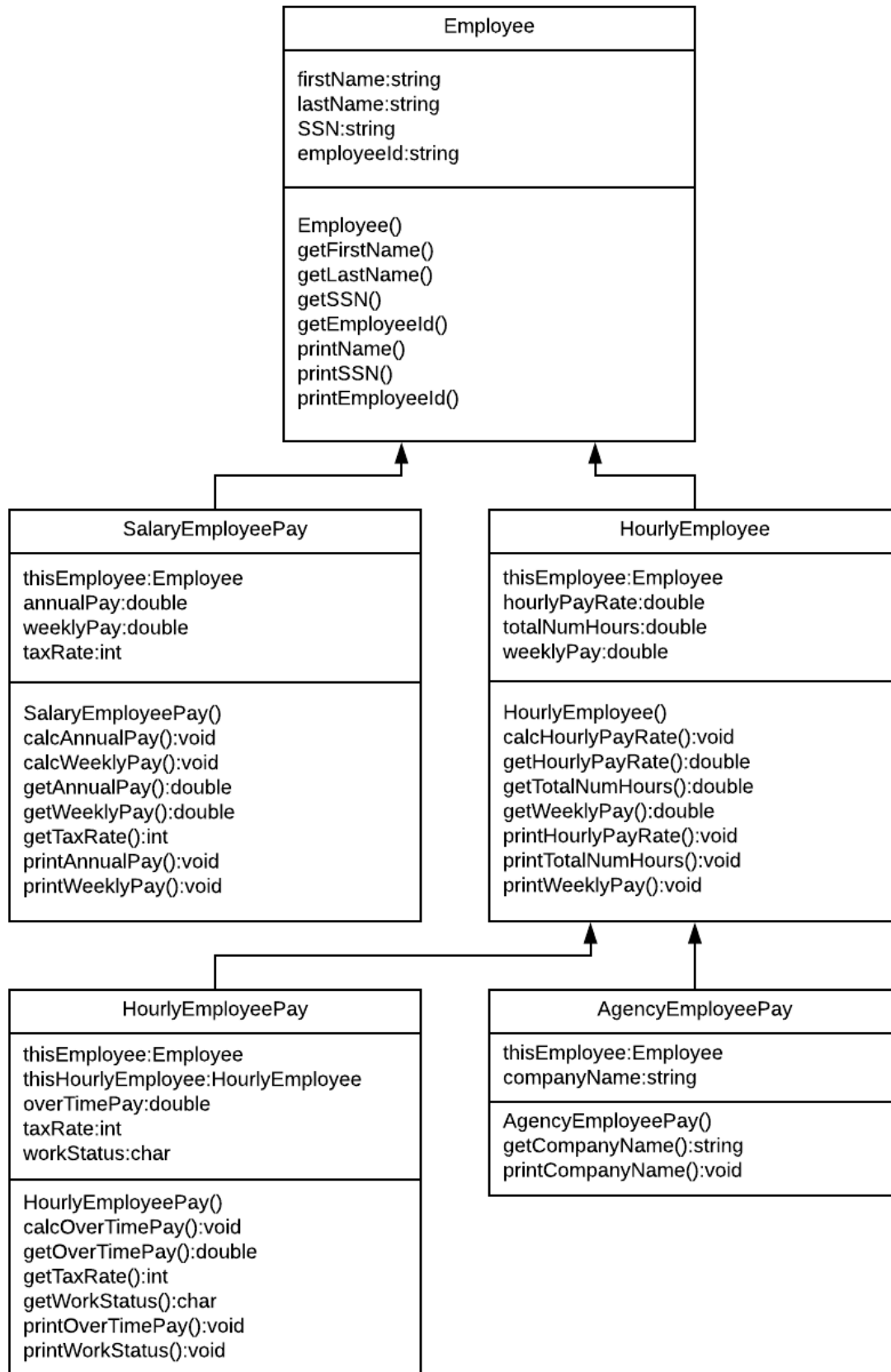


4. Decomposition Diagram

Main		
Input	Process	Output
First & last name	Assign first and last name in separate variables	When function called, print first and last name
Social Security Number (SSN)	Assign SSN to its variable in the format xxx-xxx-xxxx	When function called, print SSN with dashes
	Check if SSN is valid with no alphabetic characters	
Employee Number	Assign employee # to its variable in the format xxx-L	When function called, print employee # with dash
	Check if employee # is valid and matches its format	

Menu Selection	Call the function with respect to number selected by user	
Company name	Assign company name to its appropriate variable	When function called, print employee # with dash
Annual pay	Assign annual pay to its appropriate variable	
	Calculate weekly pay	
	Calculate hourly pay rate	
Work Status	Assign the character to its appropriate variable	
# of hours worked	Assign # of hours worked to its variable	
Tax Rate Code	Check if tax rate code is between 1 and 3	
	Use tax rate code to figure out the pay	

5. UML Classes Design Part 2



6. Test Strategy

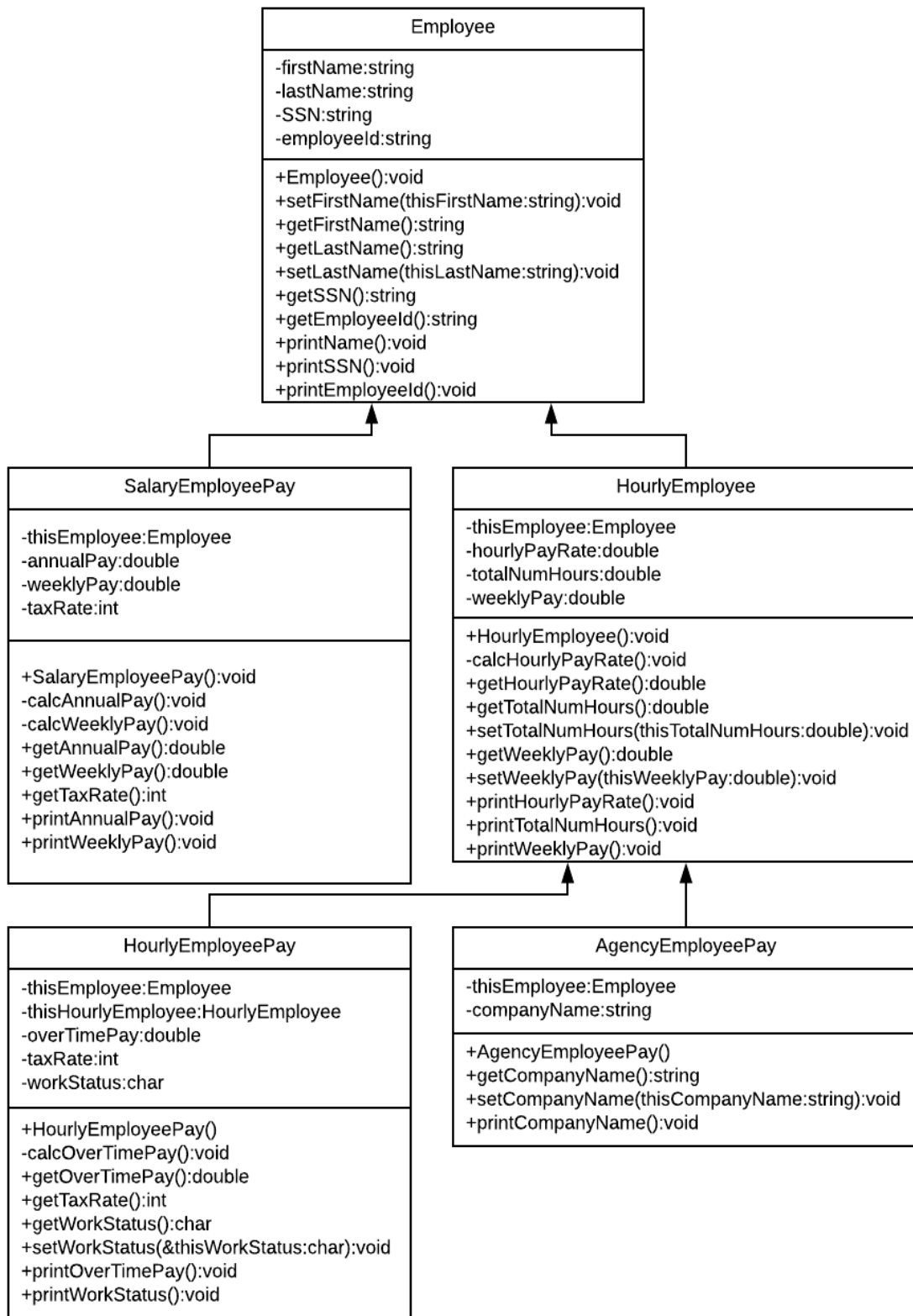
- a. Valid Data
- b. Invalid Data

7. Test Plan Version 1

Test Strategy	Test Number	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid	1	SSN has no alphabetic characters. ONLY numbers.				
Valid	2	Employee # has 3 numbers and 1 letter at the end				
Valid	3	Only positive number accepted for annual pay				
Valid	4	Only values between 1 and 3 are accepted for tax rate code				
Valid	5	Hourly pay rate ≥ 10				
Valid	6	Hourly pay rate ≤ 75				
Valid	7	Only values 'F' or 'P' accepted for work status				
Invalid	1	User inputs alphabets in SSN				
Invalid	2	User inputs only numbers for employee number				

Invalid	3	User inputs only letters for employee number				
Invalid	4	User inputs negative # for annual pay				
Invalid	5	User inputs value for tax rate code that is not between 1 and 3				
Invalid	6	Hourly pay rate is less than 10				
Invalid	7	Hourly pay rate is more than 75				
Invalid	8	User inputs more than 60 hours worked				
Invalid	9	User inputs value for work status other than 'F' or 'P'				

8. UML Classes Design Part 3



9. Initial Algorithm

- a. In main function
 - i. Include all the classes and libraries to this file
 - ii. Make various employee instances
 1. Get data from user and store it in a specific variable
 2. For each instance calculate employee's pay
 3. Display her/his pay information on the screen by calling print functions
 4. Save the same displays in a file *pay.dat*
- b. In *Employee* class
 - i. The constructor
 1. Ask user for employee name and store it in appropriate variable
 2. Ask for Social Security #
 - a. Loop until social security format correct
 3. Ask for employee #
 - a. Loop until employee # format is correct
 - ii. In *setFirstName(string thisFirstName)* function
 1. Set thisFirstName to current first name's variable
 - iii. In *setLastName(string thisLastName)* function
 1. Set thisLastName to current last name's variable
 - iv. In *getFirstName()* function
 1. return current firstName variable
 - v. In *getLastName()* function
 1. return current lastName variable
 - vi. In *getSSN()* function
 1. return current Social Security Number variable
 - vii. In *getEmployeeId()* function
 1. return current employee id variable
 - viii. In *printName()* function
 1. add first and last name variables and print it to screen
 - ix. In *printSSN()* function
 1. Print Social Security Number variable to screen
 - x. In *printEmployeeId()* function
 1. Print Employee Id variable to screen
- c. In *SalaryEmployeePay* – derived from *Employee* class
 - i. Constructor
 1. Ask user for annual pay
 - a. Loop until user enters positive number for annual pay
 2. Ask user for a tax rate code
 - a. Loop until user enters # between 1 and 3
 - i. 1 = 0.25
 - ii. 2 = 0.20
 - iii. 3 = 0.15

- ii. In *calcWeeklyPay()* function
 - 1. Calculate and store weekly pay in its appropriate variable
 - a. $\text{Weekly pay} = \text{annual pay} / 52.142$
- iii. In *getAnnualPay()* function
 - 1. return current annual pay variable
- iv. In *getWeeklyPay()* function
 - 1. return calculated weekly pay variable
- v. In *getTaxRate()* function
 - 1. return current selected tax rate code variable
- vi. In *printAnnualPay()* function
 - 1. Print annual pay variable to screen
- vii. In *printWeeklyPay()* function
 - 1. Print weekly pay variable to screen
- d. In *HourlyEmployee* class – derived from *Employee* class
 - i. Constructor
 - 1. Loop following until hourly pay rate is calculated to be between 10 and 75
 - a. Ask user for hours worked in a week and store it in a variable
 - b. Ask user for weekly pay and store it in appropriate variable
 - ii. In *calcHourlyPayRate()* function
 - 1. Using weekly pay and total # of hours worked, calculate hourly pay rate
 - a. $\text{Hourly pay rate} = \text{weekly pay} / \text{total \# of hours worked}$
 - iii. In *getHourlyPayRate()* function
 - 1. return calculated hourly pay rate
 - iv. In *getTotalNumHours()* function
 - 1. return total number of hours variable
 - v. In *setTotalNumHours(double thisTotalNumHours)* function
 - 1. $\text{totalNumHours} = \text{thisTotalNumHours}$
 - vi. In *getWeeklyPay()* function
 - 1. return current weekly pay variable
 - vii. In *setWeeklyPay(double thisWeeklyPay)* function
 - 1. $\text{weeklyPay} = \text{thisWeeklyPay}$
 - viii. In *printHourlyPayRate()* function
 - 1. Print calculated hourly pay rate to screen
 - ix. In *printTotalNumHours()* function
 - 1. Print total number of hours variable to screen
 - x. In *printWeeklyPay()* function
 - 1. Print current weekly pay variable to screen
- e. In *HourlyEmployeePay* class - derived from *HourlyEmployee* class
 - i. Constructor

1. Ask user to input tax rate code between 1 and 3 and loop until it is in this range
 - a. 1 = 0.25
 - b. 2 = 0.20
 - c. 3 = 0.15
2. Ask the user for their work status. 'F' = Full time and 'P' = Part time
 - a. Loop until user inputs 'F' or 'P'
- ii. In *calcOverTimePay()* function
 1. Using hourly rate and total # of hours from previous class to calculate overtime pay
 - a. If total # of hours > 40 then Overtime pay = 1.5 * hourly rate. Else, overtime pay = 0
- iii. In *getOverTimePay()* function
 1. return calculated over time pay variable
- iv. In *getTaxRate()* function
 1. return current tax rate code variable
- v. In *getWorkStatus()* function
 1. return current work status variable
- vi. In *setWorkStatus(char thisWorkStatus)* function
 1. workStatus = thisWorkStatus
- vii. In *printOverTimePay()* function
 1. Print calculated over time pay to screen
- viii. In *printWorkStatus()* function
 1. Print current work status variable
- f. In *AgencyEmployeePay* class - derived from *HourlyEmployee* class
 - i. Constructor
 1. Ask user to input a company name and set it to its appropriate variable
 - ii. In *geCompanyName()* function
 1. return current company name variable
 - iii. In *setCompanyName(string thisCompanyName)* function
 1. companyName = thisCompanyName
 - iv. In *printCompanyName()* function
 1. Print current company name variable to screen

10. Test Plan Version 2

Test Strategy	Test Number	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid	1	SSN has no alphabetic characters. ONLY numbers.	“123456789”	“SSN successfully taken”		
Valid	2	Employee # has 3 numbers and 1 letter at the end	“123N”	“Employee # successfully taken”		
Valid	3	Only positive number accepted for annual pay	“100”	“Annual pay successfully taken”		
Valid	4	Only values between 1 and 3 are accepted for tax rate code	“1”	“Tax rate of 25% selected successfully”		
Valid	5	Hourly pay rate between 10 and 75	“50”	“Hourly pay rate successfully calculated”		
Valid	6	Only values ‘F’ or ‘P’ accepted for work status	‘F’	“Work status successfully updated to Full Time”		
Invalid	1	User inputs alphabets in SSN	“12345abcd”	“Invalid Input: incorrect SSN format”		
Invalid	2	User inputs only numbers for employee number	“1234”	“Invalid Input: Incorrect Employment # format”		

Invalid	3	User inputs only letters for employee number	“ABCD”	“Invalid Input: Incorrect Employment # format”		
Invalid	4	User inputs negative # for annual pay	“-20”	“Invalid Input: Negative # not allowed”		
Invalid	5	User inputs value for tax rate code that is not between 1 and 3	“5”	“Invalid Input: Out of bounds tax rate code”		
Invalid	6	Hourly pay rate is less than 10	“1”	“Invalid Input: Hourly pay cannot be less than 10”		
Invalid	7	Hourly pay rate is more than 75	“100”	“Invalid Input: Hourly pay cannot be more than 75”		
Invalid	8	User inputs more than 60 hours worked	“100”	“Invalid Input: Total hours worked in a week cannot exceed 60”		
Invalid	9	User inputs value for work status other than ‘F’ or ‘P’	“A”	“Invalid Input: Only values ‘F’ or ‘P’ is allowed for work status”		

11. Code

Project_2_Home_v2 (*Main()* function inside)

```
#include "stdafx.h"
#include <fstream>
#include "SalaryEmployeePay.h"
#include "HourlyEmployeePay.h"
#include "AgencyEmployeePay.h"

using namespace std;

bool checkIfNum(string);
bool checkId(string);
bool checkSSN(string);
bool checkWorkStatus(char);

int main()
{
    ofstream output;
    output.open("pay.dat");
    string firstName, lastName, SSN, employeeId, companyName;
    double totalNumHours, weeklyPay, hourlyPayRate, annualPay;
    int userInput, taxRate;
    char workStatus;

    do {
        cout << "Type 1 for Salary Employee\nType 2 for Hourly Employee\nType 0 to quit:" << endl;
        cin >> userInput;

        if (userInput == 2) {
            system("CLS");
            HourlyEmployeePay myEmployee;

            cout << "Your full name: ";
            cin >> firstName;
            cin >> lastName;
            myEmployee.setFirstName(firstName);
            myEmployee.setLastName(lastName);

            //.....

            do {
                cout << endl << "Social Security #: ";
                cin >> SSN;
                if (checkIfNum(SSN) == false)
                {
                    cout << endl << "Sorry, Social security number is not in right format.\nTry again:";
                }
            }
        }
    }
}
```

```

} while (checkIfNum(SSN) == false);
myEmployee.setSSN(SSN);
cout << endl << "SSN successfully taken";

//.....

do {
    cout << endl << "Employee Id: ";
    cin >> employeeId;
    if (checkId(employeeId) == false)
    {
        cout << endl << "Sorry, Id is not in right format. Try
again:";
    }
} while (checkId(employeeId) == false);
myEmployee.setEmployeeId(employeeId);
cout << endl << "Employee # successfully taken";

//.....

do {
    do {
        cout << endl << "Number of hours worked in a week:";
        cin >> totalNumHours;
        if (totalNumHours >= 60 || totalNumHours < 0)
        {
            cout << "Sorry, number of hours worked in a week
can only be positive and less than 60.\nPlease try again." << endl;
        }
        cout << endl << "# of hours worked successfully taken";
    } while (totalNumHours >= 60 || totalNumHours < 0);
    cout << endl << "Hourly Pay Rate:";
    cin >> hourlyPayRate;
    myEmployee.setHourlyPayRate(hourlyPayRate);
    myEmployee.setTotalNumHours(totalNumHours);
    if (myEmployee.getHourlyPayRate() < 10)
    {
        cout << endl << "Sorry, hourly pay rate cannot be less
than 10.\nPlease try again:" << endl;
    }
    else if (myEmployee.getHourlyPayRate() > 75) {
        cout << endl << "Sorry, hourly pay rate cannot be more
than 75.\nPlease try again:" << endl;
    }
    else if (totalNumHours > 60)
    {
        cout << endl << "Sorry, total number of hours cannot be
more than 60" << endl;
    }
} while (myEmployee.getHourlyPayRate() < 10 ||
myEmployee.getHourlyPayRate() > 75 || totalNumHours > 60);
cout << endl << "Hourly pay rate successfully calculated";

//.....

```

```

do {
    cout << endl << "Tax Rate Code: ";
    cin >> taxRate;
    if (taxRate != 1 && taxRate != 2 && taxRate != 3)
    {
        cout << endl << "Sorry, only tax rate codes between 1
and 3 are allowed.\nPlease try again" << endl;
    }
} while (taxRate != 1 && taxRate != 2 && taxRate != 3);
myEmployee.setTaxRate(taxRate);
cout << endl << "Tax rate of " << myEmployee.getTaxRate() << "%
selected successfully";

//.....

do {
    cout << endl << "Work Status: ";
    cin >> workStatus;
    if (checkWorkStatus(workStatus) == false)
    {
        cout << endl << "Sorry, invalid work status. Only
accepted values are F and P.\nPlease try again" << endl;
    }
} while (checkWorkStatus(workStatus) == false);
myEmployee.setWorkStatus(workStatus);
if (myEmployee.getWorkStatus() == 'F') {
    cout << "Work status successfully updated to Full Time" <<
endl;
}
else {
    cout << "Work status successfully updated to Part Time" <<
endl;
}
//.....

AgencyEmployeePay agency;

cout << "Company Name: ";
cin >> companyName;
agency.setCompanyName(companyName);
cout << "Company name successfully set to " <<
agency.getCompanyName() << endl << endl;

//-----Print Results-----
-----

output << "Hourly Employee Data and Calculations" << endl;

myEmployee.printName();
output << "Name: " << myEmployee.getFirstName() << " " <<
myEmployee.getLastName() << endl;

```

```

myEmployee.printEmployeeId();
output << "Employee Id: " << myEmployee.getEmployeeId() << endl;

myEmployee.printSSN();
output << "Social Security Number: " << myEmployee.getSSN() << endl;

myEmployee.printTotalNumHours();
output << "Number of hours worked in a week: " <<
myEmployee.getTotalNumHours() << endl;

myEmployee.printWeeklyPay();
output << "Weekly Pay: $" << myEmployee.getWeeklyPay() << endl;

myEmployee.printOverTimePay();
output << "Overtime pay: $" << myEmployee.getOverTimePay() << endl;

myEmployee.printHourlyPayRate();
output << "Hourly Pay Rate: $" << myEmployee.getHourlyPayRate() <<
endl;

myEmployee.printTaxRate();
output << "Tax Rate: " << myEmployee.getTaxRate() << "%" << endl;

myEmployee.printWorkStatus();
if (myEmployee.getWorkStatus() == 'F') {
    output << "Work Status: Full time" << endl;
}
else {
    output << "Work Status: Part time" << endl;
}

agency.printCompanyName();
output << "Company Name: " << agency.getCompanyName() << endl;

cout << endl;
output << endl;
//-----
}
else if (userInput == 1) {
    system("CLS");

    SalaryEmployeePay mySalEmp;

    cout << "Your full name: ";
    cin >> firstName;
    cin >> lastName;
    mySalEmp.setFirstName(firstName);
    mySalEmp.setLastName(lastName);

    do {
        cout << endl << "Social Security #: ";
        cin >> SSN;
        if (checkIfNum(SSN) == false)
        {

```



```

        cout << endl << "Sorry, Social security number is not
in right format.\nTry again:";
    }
} while (checkIfNum(SSN) == false);
mySalEmp.setSSN(SSN);
cout << endl << "SSN successfully taken";

do {
    cout << endl << "Employee Id: ";
    cin >> employeeId;
    if (checkId(employeeId) == false)
    {
        cout << endl << "Sorry, Id is not in right format. Try
again:";
    }
} while (checkId(employeeId) == false);
mySalEmp.setEmployeeId(employeeId);
cout << endl << "Employee # successfully taken";

do {
    cout << endl << "Annual pay:";
    cin >> annualPay;

    if (annualPay < 0) {
        cout << "Sorry, only positive values accepted.\nPlease
try again." << endl;
    }
} while (annualPay < 0);
mySalEmp.setAnnualPay(annualPay);
cout << endl << "Annual pay successfully taken";

do {
    cout << endl << "Tax Rate Code: ";
    cin >> taxRate;
    if (taxRate != 1 && taxRate != 2 && taxRate != 3)
    {
        cout << endl << "Sorry, only tax rate codes between 1
and 3 are allowed.\nPlease try again" << endl;
    }
} while (taxRate != 1 && taxRate != 2 && taxRate != 3);
mySalEmp.setTaxRate(taxRate);
cout << endl << "Tax rate of " << mySalEmp.getTaxRate() << "%
selected successfully" << endl << endl;

//-----Print Results-----
-----

output << "Salary Employee Data and Calculations" << endl;

mySalEmp.printName();
output << "Name: " << mySalEmp.getFirstName() << " " <<
mySalEmp.getLastName() << endl;

mySalEmp.printEmployeeId();
output << "Employee Id: " << mySalEmp.getEmployeeId() << endl;

```

```

mySalEmp.printSSN();
output << "Social Security Number: " << mySalEmp.getSSN() << endl;

mySalEmp.printAnnualPay();
output << "Annual Pay: $" << mySalEmp.getAnnualPay() << endl;

mySalEmp.printTaxRate();
if (mySalEmp.getTaxRate() == 1) {
    cout << "Tax Rate: 25%" << endl;
}
else if (mySalEmp.getTaxRate() == 2) {
    cout << "Tax Rate: 20%" << endl;
}
else {
    cout << "Tax Rate: 15%" << endl;
}

mySalEmp.printWeeklyPay();
output << "Weekly Pay: $" << mySalEmp.getWeeklyPay() << endl;

cout << endl;
output << endl;
//-----
}
} while (userInput != 0);

output.close();
return 0;
}

bool checkWorkStatus(char status)
{
    bool result = true;
    status = toupper(status);
    if (status != 'F' && status != 'P')
    {
        result = false;
    }

    return result;
}

bool checkIfNum(string ssn)
{
    bool result = true;
    if (ssn.length() == 9) {
        for (int i = 0; i < ssn.length() && result == true; i++)
        {
            if (isdigit(ssn.at(i))) {
                result = true;
            }
            else {
                result = false;
            }
        }
    }
}

```

```
    }
    else {
        result = false;
    }
    return result;
}

bool checkId(string employeeId) {
    bool result = true;
    if (employeeId.length() == 4) {
        for (int i = 0; i < employeeId.length() - 1 && result == true; i++)
        {
            if (isdigit(employeeId.at(i))) {
                result = true;
            }
            else {
                result = false;
            }
        }
        if (result && isalpha(employeeId.at(3)))
        {
            result = true;
        }
        else {
            result = false;
        }
    }
    else {
        result = false;
    }

    return result;
}
```

SalaryEmployeePay.h

```
#ifndef SalaryEmployeePay_H
#define SalaryEmployeePay_H

#include "Employee.h"

using namespace std;

class SalaryEmployeePay : public Employee
{
public:
    SalaryEmployeePay();
    double getAnnualPay();
    double getWeeklyPay();
    int getTaxRate();
    void printAnnualPay();
    void printWeeklyPay();
    void setAnnualPay(double thisAnnualPay);
    void setTaxRate(int thisTaxRate);
    void printTaxRate();
}
```

```
private:
    void calcWeeklyPay();

    /*Employee thisEmployee;*/
    double annualPay;
    double weeklyPay;
    int taxRate;
};

#endif
```

SalaryEmployeePay.cpp

```
#include "stdafx.h"
#include <string>
#include <iostream>
#include "SalaryEmployeePay.h"
using namespace std;

SalaryEmployeePay::SalaryEmployeePay()
{
    annualPay = NULL;
    taxRate = NULL;
}

void SalaryEmployeePay::setAnnualPay(double thisAnnualPay)
{
    annualPay = thisAnnualPay;
}

void SalaryEmployeePay::setTaxRate(int thisTaxRate)
{
    taxRate = thisTaxRate;
}

void SalaryEmployeePay::calcWeeklyPay()
{
    if (taxRate == 1) {
        weeklyPay = (annualPay / 52.142) - ((annualPay / 52.142) * 0.25);
    }
    else if (taxRate == 2)
    {
        weeklyPay = (annualPay / 52.142) - ((annualPay / 52.142) * 0.2);
    }
    else {
        weeklyPay = (annualPay / 52.142) - ((annualPay / 52.142) * 0.15);
    }
}

double SalaryEmployeePay::getAnnualPay()
{
    return annualPay;
}

double SalaryEmployeePay::getWeeklyPay()
{
    calcWeeklyPay();
    return weeklyPay;
}
```

```
}

int SalaryEmployeePay::getTaxRate()
{
    if (taxRate == 1) {
        return 25;
    }
    else if (taxRate == 2) {
        return 20;
    }
    else {
        return 15;
    }
}

void SalaryEmployeePay::printAnnualPay()
{
    cout << "Annual Pay: $" << annualPay << endl;
}

void SalaryEmployeePay::printWeeklyPay()
{
    calcWeeklyPay();
    cout << "Weekly Pay: $" << weeklyPay << endl;
}

void SalaryEmployeePay::printTaxRate()
{
    if (taxRate == 1) {
        cout << "Tax Rate: 25%" << endl;
    }
    else if (taxRate == 2) {
        cout << "Tax Rate: 20%" << endl;
    }
    else {
        cout << "Tax Rate: 15%" << endl;
    }
}
```

Employee.h

```
#ifndef EMPLOYEE_H
#define EMPLOYEE_H

#include <string>
#include <iostream>

using namespace std;

class Employee
{
public:
    Employee();
```

```
    Employee(string thisFirstName, string thisLastName, string thisSSN, string
thisEmployeeId);
    string getFirstName();
    void setFirstName(string thisFirstName);
    string getLastName();
    void setLastName(string thisLastName);
    string getSSN();
    string getEmployeeId();
    void printSSN();
    void printEmployeeId();
    void setSSN(string thisSSN);
    void setEmployeeId(string thisEmployeeId);
    void printName();
private:
    string firstName;
    string lastName;
    string SSN;
    string employeeId;
};

#endif // !EMPLOYEE_H
```

Employee.cpp

```
#include "stdafx.h"
#include "Employee.h"

Employee::Employee()
{
    firstName = "";
    lastName = "";
    SSN = "";
    employeeId = "";
}

Employee::Employee(string thisFirstName, string thisLastName, string thisSSN, string
thisEmployeeId)
{
    firstName = thisFirstName;
    lastName = thisLastName;
    SSN = thisSSN;
    employeeId = thisEmployeeId;
}

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

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

string Employee::getLastName()
{
```

```
        return lastName;
    }

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

    string Employee::getSSN()
    {
        string result;
        result = "XXX-XX-" + SSN.substr(5, 4);
        SSN = result;
        return result;
    }

    string Employee::getEmployeeId()
    {
        employeeId[3] = toupper(employeeId[3]);
        string result = employeeId.substr(0, 3) + "-" + employeeId.substr(3, 1);
        return result;
    }

    void Employee::printName()
    {
        cout << "Name: " << firstName << " " << lastName << endl;
    }

    void Employee::printSSN()
    {
        string result;
        result = "XXX-XX-" + SSN.substr(5, 4);
        cout << "Social Security Number: " << result << endl;
    }

    void Employee::printEmployeeId()
    {
        employeeId[3] = toupper(employeeId[3]);
        string result = employeeId.substr(0, 3) + "-" + employeeId.substr(3, 1);
        cout << "Employee Id: " << result << endl;
    }

    void Employee::setSSN(string thisSSN)
    {
        SSN = thisSSN;
    }

    void Employee::setEmployeeId(string thisEmployeeId)
    {
        employeeId = thisEmployeeId;
    }
}
```

HourlyEmployee.h

```
#ifndef HourlyEmployee_H
#define HourlyEmployee_H

#include "Employee.h"

class HourlyEmployee : public Employee
{
public:
    HourlyEmployee();
    double getHourlyPayRate();
    double getTotalNumHours();
    void setTotalNumHours(double thisTotalNumHours);
    void printHourlyPayRate();
    void printTotalNumHours();
    double totalNumHours;    //user input
    double hourlyPayRate;    //input

    void setHourlyPayRate(double thisHourlyPayRate);
private:

    /*Employee thisEmployee;*/

};

#endif
```

HourlyEmployee.cpp

```
#include "stdafx.h"
#include <string>
#include <iostream>
#include "HourlyEmployee.h"

HourlyEmployee::HourlyEmployee()
{
    totalNumHours = NULL;
    hourlyPayRate = NULL;
}

double HourlyEmployee::getHourlyPayRate()
{
    return hourlyPayRate;
}

double HourlyEmployee::getTotalNumHours()
{
    return totalNumHours;
}

void HourlyEmployee::setTotalNumHours(double thisTotalNumHours)
{
    totalNumHours = thisTotalNumHours;
}
```



```
void HourlyEmployee::setHourlyPayRate(double thisHourlyPayRate)
{
    hourlyPayRate = thisHourlyPayRate;
}

void HourlyEmployee::printHourlyPayRate()
{
    cout << "Hourly Pay: $" << hourlyPayRate << endl;
}

void HourlyEmployee::printTotalNumHours()
{
    cout << "Number of hours worked in a week: " << totalNumHours << endl;
}
```

HourlyEmployeePay.h

```
#ifndef HourlyEmployeePay_H
#define HourlyEmployeePay_H

#include "HourlyEmployee.h"

class HourlyEmployeePay : public HourlyEmployee
{
public:
    HourlyEmployeePay();
    double getOverTimePay();
    int getTaxRate();
    char getWorkStatus();
    void setWorkStatus(char &thisWorkStatus);
    void printOverTimePay();
    void printWorkStatus();
    void setTaxRate(char thisTaxRate);
    double getWeeklyPay();
    void printWeeklyPay();
    void printTaxRate();
private:
    void calcOverTimePay();
    void calcWeeklyPay();

    /*Employee thisEmployee;
    HourlyEmployee thisHourlyEmployeePay;*/
    double overTimePay;           //Calculated by function
    int taxRate;                  //User input
    char workStatus;              //User input
    double weeklyPay;

};

#endif
```

HourlyEmployeePay.cpp

```
#include "stdafx.h"
#include <string>
#include <iostream>
#include "HourlyEmployeePay.h"
#include "Employee.h"

HourlyEmployeePay::HourlyEmployeePay()
{
    taxRate = NULL;
    workStatus = NULL;
    overTimePay = NULL;
    weeklyPay = NULL;
}

void HourlyEmployeePay::calcWeeklyPay()
{
    if (taxRate == 1) {
        weeklyPay = (hourlyPayRate * totalNumHours) - (hourlyPayRate *
totalNumHours * 0.25);
    }
    else if (taxRate == 2)
    {
        weeklyPay = (hourlyPayRate * totalNumHours) - (hourlyPayRate *
totalNumHours * 0.2);
    }
    else {
        weeklyPay = (hourlyPayRate * totalNumHours) - (hourlyPayRate *
totalNumHours * 0.15);
    }
}

double HourlyEmployeePay::getWeeklyPay()
{
    calcWeeklyPay();
    return weeklyPay;
}

void HourlyEmployeePay::printWeeklyPay()
{
    calcWeeklyPay();
    cout << "Weekly Pay: $" << weeklyPay << endl;
}

double HourlyEmployeePay::getOverTimePay()
{
    calcOverTimePay();
    return overTimePay;
}

int HourlyEmployeePay::getTaxRate()
{
    if (taxRate == 1) {
        return 25;
    }
}
```

```
    }
    else if (taxRate == 2) {
        return 20;
    }
    else {
        return 15;
    }
}

char HourlyEmployeePay::getWorkStatus()
{
    toupper(workStatus);
    return workStatus;
}

void HourlyEmployeePay::setWorkStatus(char &thisWorkStatus)
{
    workStatus = toupper(thisWorkStatus);
}

void HourlyEmployeePay::printOverTimePay()
{
    calcOverTimePay();
    cout << "Overtime pay: $" << overTimePay << endl;
}

void HourlyEmployeePay::printWorkStatus()
{
    if (workStatus == 'F') {
        cout << "Work Status: Full time" << endl;
    }
    else {
        cout << "Work Status: Part time" << endl;
    }
}

void HourlyEmployeePay::calcOverTimePay()
{
    overTimePay = 1.5 * getHourlyPayRate();
}

void HourlyEmployeePay::setTaxRate(char thisTaxRate)
{
    taxRate = thisTaxRate;
}

void HourlyEmployeePay::printTaxRate()
{
    if (taxRate == 1) {
        cout << "Tax Rate: 25%" << endl;
    }
    else if (taxRate == 2) {
        cout << "Tax Rate: 20%" << endl;
    }
    else {
        cout << "Tax Rate: 15%" << endl;
    }
}
```

```
}
```

AgencyEmployeePay.h

```
#ifndef AgencyEmployeePay_H
#define AgencyEmployeePay_H

#include "HourlyEmployee.h"
#include <string>
#include <iostream>

class AgencyEmployeePay : public HourlyEmployee
{
public:
    AgencyEmployeePay();
    string getCompanyName();
    void setCompanyName(string thisCompanyName);
    void printCompanyName();
private:
    /*Employee thisEmployee;*/
    string companyName;
};

#endif
```

AgencyEmployeePay.cpp

```
#include "stdafx.h"
#include <string>
#include <iostream>
#include "AgencyEmployeePay.h"

AgencyEmployeePay::AgencyEmployeePay()
{
    companyName = "";
}

string AgencyEmployeePay::getCompanyName()
{
    return companyName;
}

void AgencyEmployeePay::setCompanyName(string thisCompanyName)
{
    companyName = thisCompanyName;
}

void AgencyEmployeePay::printCompanyName()
{
    cout << "Company Name: " << companyName << endl;
}
```

12. Updated Algorithm

- a. In *checkIfNum()* function
 - i. Set a Boolean variable to true initially.
 - ii. If SSN length equals 9
 1. For loop from 0 until Boolean variable is false and reached SSN length
 2. If current digit of SSN is a number, set Boolean variable to true
 3. Else set Boolean variable to false
 - iii. Else set Boolean variable to false
 - iv. Return the Boolean variable
- b. In *checkId()* function
 - i. Set a Boolean variable to true initially
 - ii. If id length equals 4
 1. For loop from 0 until id length – 1 and until Boolean variable is false
 - a. If current digit is number, set Boolean variable to true
 - b. Else set Boolean variable to false
 2. If Boolean variable is true and current digit is a letter, set Boolean variable to true
 3. Else set Boolean variable to false
 - iii. Else set Boolean variable to false
 - iv. Return Boolean variable
- c. In *checkWorkStatus()* function
 - i. Set a Boolean variable to true initially
 - ii. Set the passed value to all upper case
 - iii. If the passed value does not equal F or P, set Boolean variable to false
 - iv. Return the Boolean variable
- d. In main function
 - i. Include all the classes and libraries to this file
 - ii. Make various employee instances
 1. Get data from user and store it in a specific variable
 - a. Check the validity of user inputs by calling check function
 2. For each instance calculate employee's pay by calling function
 3. For each instance, ask user for values to be put in to all the variables in every class and perform checks. Loop until correct value is given:
 - a. Social Security # can only have one form: "XXX-XX-####"
 - b. Employee id can only have one form: "###-L"
 - c. # of hours of worked in a week can only be positive and less than 60
 - d. Hourly pay rate cannot be more than 75 and less than 10
 - e. Tax rate code can only be between 1 and 3

- f. Work status can only be 'F' or 'P'
 - g. Annual pay can only be positive
 - 4. Display her/his pay information on the screen by calling print functions
 - 5. Save the same displays in a file *pay.dat*
 - e. In *Employee* class
 - i. The constructor
 - 1. Ask user for employee name and store it in appropriate variable
 - 2. Ask for Social Security #
 - a. Loop until social security format correct
 - 3. Ask for employee #
 - a. Loop until employee # format is correct
 - 4. Set first name, last name, SSN, and employee id to empty string.
 - ii. Constructor with parameters of all 4 private variables
 - 1. Set each private variable to values of the respective passed in variables
 - iii. In *setFirstName(string thisFirstName)* function
 - 1. Set thisFirstName to current first name's variable
 - iv. In *setLastName(string thisLastName)* function
 - 1. Set thisLastName to current last name's variable
 - v. In *getFirstName()* function
 - 1. return current firstName variable
 - vi. In *getLastName()* function
 - 1. return current lastName variable
 - vii. In *getSSN()* function
 - 1. return current Social Security Number variable
 - 2. Make a string variable called result
 - a. "XXX-XX-" + last 4 digits of social security number
 - 3. Return result
 - viii. In *getEmployeeId()* function
 - 1. return current employee id variable
 - 2. Make the last digit of employee id to uppercase
 - 3. Make a string variable called result
 - a. First 3 digits + "-" + Last digit
 - 4. Return result
 - ix. In *printName()* function
 - 1. add first and last name variables and print it to screen
 - x. In *printSSN()* function
 - 1. Print Social Security Number variable to screen
 - 2. Same as *getSSN()* and print result to screen
 - xi. In *printEmployeeId()* function
 - 1. Print Employee Id variable to screen
 - 2. Same as *getEmployeeId()* and print result to screen

- f. In *SalaryEmployeePay* – derived from *Employee* class
 - i. Constructor
 - 1. ~~Ask user for annual pay~~
 - a. ~~Loop until user enters positive number for annual pay~~
 - 2. ~~Ask user for a tax rate code~~
 - a. ~~Loop until user enters # between 1 and 3~~
 - i. ~~1 = 0.25~~
 - ii. ~~2 = 0.20~~
 - iii. ~~3 = 0.15~~
 - 3. Set annual pay and tax rate to null
 - ii. In *setAnnualPay()* function
 - 1. Set annual pay to the value passed in
 - iii. In *setTaxRate()* function
 - 1. Set tax rate to the value passed in
 - iv. In *calcWeeklyPay()* function
 - 1. ~~Calculate and store weekly pay in its appropriate variable~~
 - a. ~~Weekly pay = annual pay / 52.142~~
 - 2. If tax rate is 1
 - a. $\text{Weekly pay} = (\text{annual pay} / 52.142) - ((\text{annual pay} / 52.142) * 0.25)$
 - 3. Else if tax rate is 2
 - a. $\text{Weekly pay} = (\text{annual pay} / 52.142) - ((\text{annual pay} / 52.142) * 0.2)$
 - 4. Else Weekly pay = $(\text{annual pay} / 52.142) - ((\text{annual pay} / 52.142) * 0.15)$
 - v. In *getAnnualPay()* function
 - 1. return current annual pay variable
 - vi. In *getWeeklyPay()* function
 - 1. Call function to calculate weekly pay
 - 2. return calculated weekly pay variable
 - vii. In *getTaxRate()* function
 - 1. ~~return current selected tax rate code variable~~
 - 2. If tax rate is 1
 - a. Return 25
 - 3. Else if tax rate is 2
 - a. Return 20
 - 4. Else return 15
 - viii. In *printAnnualPay()* function
 - 1. Print annual pay variable to screen
 - ix. In *printWeeklyPay()* function
 - 1. Print weekly pay variable to screen
 - x. In *printTaxRate()* function
 - 1. If tax rate is 1

- a. Print “Tax Rate: 25%” to screen
2. Else if tax rate is 2
 - a. Print “Tax Rate: 20%” to screen
3. Else print “Tax Rate: 15%” to screen
- g. In *HourlyEmployee* class – derived from *Employee* class
 - i. Constructor
 1. ~~Loop following until hourly pay rate is calculated to be between 10 and 75~~
 - a. ~~Ask user for hours worked in a week and store it in a variable~~
 - b. ~~Ask user for weekly pay and store it in appropriate variable~~
 2. Set total number of hours and hourly pay rate to NULL
 - ii. ~~In *calcHourlyPayRate()* function~~
 1. ~~Using weekly pay and total # of hours worked, calculate hourly pay rate~~
 - a. ~~Hourly pay rate = weekly pay / total # of hours worked~~
 - iii. In *getHourlyPayRate()* function
 1. return calculated hourly pay rate
 - iv. In *getTotalNumHours()* function
 1. return total number of hours variable
 - v. In *setTotalNumHours(double thisTotalNumHours)* function
 1. totalNumHours = thisTotalNumHours
 - vi. In *getWeeklyPay()* function
 1. return current weekly pay variable
 - vii. In ~~*setWeeklyPay(double thisWeeklyPay)* function~~
 1. ~~weeklyPay = thisWeeklyPay~~
 - viii. In *setHourlyPayRate(double thisHourlyPayRate)* function
 1. hourlyPayRate = thisHourlyPayRate
 - ix. In *printHourlyPayRate()* function
 1. ~~Print calculated~~ Print hourly pay rate to screen
 - x. In *printTotalNumHours()* function
 1. Print total number of hours variable to screen
 - xi. In *printWeeklyPay()* function
 1. Print current weekly pay variable to screen
- h. In *HourlyEmployeePay* class - derived from *HourlyEmployee* class
 - i. Constructor
 1. ~~Ask user to input tax rate code between 1 and 3 and loop until it is in this range~~
 - a. ~~1 = 0.25~~
 - b. ~~2 = 0.20~~
 - c. ~~3 = 0.15~~
 2. ~~Ask the user for their work status. ‘F’ = Full time and ‘P’ = Part time~~

- a. ~~Loop until user inputs 'F' or 'P'~~
 3. Set tax rate, work status, weekly pay, and overtime pay to NULL
- ii. In *calcWeeklyPay()* function
 1. If tax rate is 1
 - a. Weekly pay = (hourly pay rate * total number of hours) – (hourly pay rate * total number of hours * 0.25)
 2. Else if tax rate is 2
 - a. Weekly pay = (hourly pay rate * total number of hours) – (hourly pay rate * total number of hours * 0.2)
 3. Else Weekly pay = (hourly pay rate * total number of hours) – (hourly pay rate * total number of hours * 0.15)
- iii. In *getWeeklyPay()* function
 1. Call function to calculate weekly pay
 2. Return weekly pay variable
- iv. In *printWeeklyPay()* function
 1. Call function to calculate weekly pay
 2. Print weekly variable to screen
- v. In *calcOverTimePay()* function
 1. Use hourly rate and total # of hours from previous class to calculate overtime pay
 - a. ~~If total # of hours > 40 then Overtime pay = 1.5 * hourly rate. Else, overtime pay = 0~~
 - b. Overtime pay = 1.5 * hourlyPayRate
- vi. In *getOverTimePay()* function
 1. Call function to calculate overtime pay
 2. return calculated over time pay variable
- vii. In *getTaxRate()* function
 1. ~~return current tax rate code variable~~
 2. If tax rate is 1
 - a. Return 25
 3. Else if tax rate is 2
 - a. Return 20
 4. Else return 15
- viii. In *getWorkStatus()* function
 1. Make work status variable to all upper case
 2. return current work status variable
- ix. In *setWorkStatus(char thisWorkStatus)* function
 1. workStatus = all uppercase of variable thisWorkStatus
- x. In *printOverTimePay()* function
 1. Call function to calculate overtime pay
 2. Print calculated overtime pay to screen
- xi. In *printWorkStatus()* function
 1. ~~Print current work status variable~~

2. If work status is 'F'
 - a. Print "Work Status: Full time" to screen
3. Else
 - a. Print "Work Status: Part time" to screen
- xii. In *setTaxRate()* function
 1. Set tax rate to value of the passed in variable
- xiii. In *printTaxRate()* function
 1. If tax rate is 1
 - a. Print "Tax Rate: 25%" to screen
 2. Else if tax rate is 2
 - a. Print "Tax Rate: 20%" to screen
 3. Else print "Tax Rate: 15%" to screen
- i. In *AgencyEmployeePay* class - derived from *HourlyEmployee* class
 - i. Constructor
 1. ~~Ask user to input a company name and set it to its appropriate variable~~
 2. Set company name variable to empty string
 - ii. In *geCompanyName()* function
 1. return current company name variable
 - iii. In *setCompanyName(string thisCompanyName)* function
 1. companyName = thisCompanyName
 - iv. In *printCompanyName()* function
 1. Print current company name variable to screen

13. Test Plan Version 3

Test Strategy	Test Number	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid	1	SSN has no alphabetic characters. ONLY numbers.	"123456789"	"SSN successfully taken"	"SSN successfully taken"	Pass
Valid	2	Employee # has 3 numbers and 1 letter at the end	"123N"	"Employee # successfully taken"	"Employee # successfully taken"	Pass
Valid	3	Only positive number accepted for annual pay	"100"	"Annual pay successfully taken"	"Annual pay successfully taken"	Pass
Valid	4	Only values	"1"	"Tax rate of 25%"	"Tax rate of 25%"	Pass

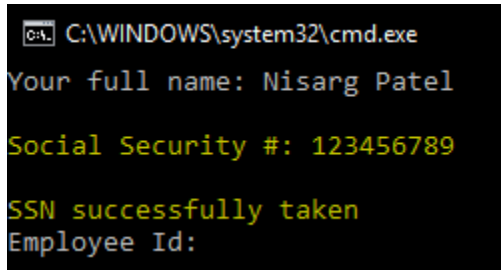
		between 1 and 3 are accepted for tax rate code		selected successfully”	25% selected successfully”	
Valid	5	Hourly pay rate between 10 and 75	“50”	“Hourly pay rate successfully calculated”	“Hourly pay rate successfully calculated”	Pass
Valid	6	Only values ‘F’ or ‘P’ accepted for work status	‘F’	“Work status successfully updated to Full Time”	“Work status successfully updated to Full Time”	Pass
Valid	7	# of hours worked less than 60	“59”	“# of hours worked successfully taken”	“# of hours worked successfully taken”	Pass
Valid	8	# of hours worked equals 0	“0”	“# of hours worked successfully taken” and weekly pay calculates to 0	“# of hours worked successfully taken” and weekly pay calculates to 0	Pass
Invalid	1	User inputs alphabets in SSN	“12345abcd”	“Invalid Input: incorrect SSN format”	Try again message saying SSN is incorrect	Pass

Invalid	2	User inputs only numbers for employee number	“1234”	“Invalid Input: Incorrect Employment # format”	Try again message saying SSN is incorrect	Pass
Invalid	3	User inputs only letters for employee number	“ABCD”	“Invalid Input: Incorrect Employment # format”	Try again message saying SSN is incorrect	Pass
Invalid	4	User inputs negative # for annual pay	“-20”	“Invalid Input: Negative # not allowed”	Try again message saying only positive numbers accepted	Pass
Invalid	5	User inputs value for tax rate code that is not between 1 and 3	“5”	“Invalid Input: Out of bounds tax rate code”	Sorry message saying only values between 1 and 3 are allowed	Pass
Invalid	6	Hourly pay rate is less than 10	“1”	“Invalid Input: Hourly pay cannot be less than 10”	Sorry message saying input cannot	Pass

					be less than 10	
Invalid	7	Hourly pay rate is more than 75	“100”	“Invalid Input: Hourly pay cannot be more than 75”	Sorry message saying input cannot be more than 75	Pass
Invalid	8	User inputs more than 60 hours worked	“100”	“Invalid Input: Total hours worked in a week cannot exceed 59”	Sorry message saying input cannot be more than 59	Pass
Invalid	9	User inputs value for work status other than ‘F’ or ‘P’	“A”	“Invalid Input: Only values ‘F’ or ‘P’ is allowed for work status”	Sorry message saying only values accepted are F and P	Pass

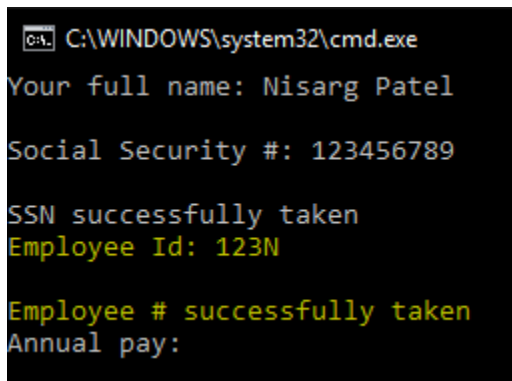
14. Screenshots

Valid Test Case 1:



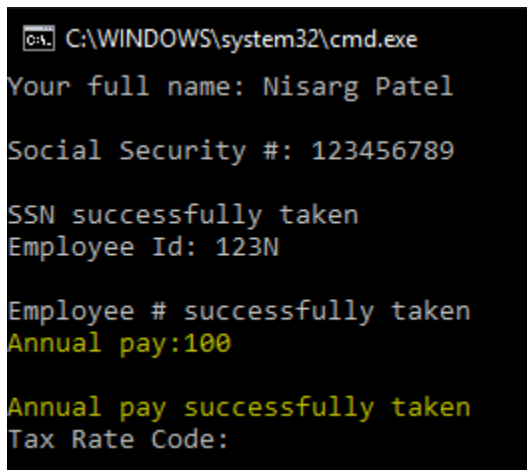
```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel
Social Security #: 123456789
SSN successfully taken
Employee Id:
```

Valid Test Case 2:



```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel
Social Security #: 123456789
SSN successfully taken
Employee Id: 123N
Employee # successfully taken
Annual pay:
```

Valid Test Case 3:



```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel
Social Security #: 123456789
SSN successfully taken
Employee Id: 123N
Employee # successfully taken
Annual pay:100
Annual pay successfully taken
Tax Rate Code:
```

Valid Test Case 4:

```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel
Social Security #: 123456789
SSN successfully taken
Employee Id: 123N
Employee # successfully taken
Annual pay:100
Annual pay successfully taken
Tax Rate Code: 1
Tax rate of 25% selected successfully
```

Valid Test Case 5:

```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel
Social Security #: 123456789
SSN successfully taken
Employee Id: 123N
Employee # successfully taken
Number of hours worked in a week:59
# of hours worked successfully taken
Hourly Pay Rate:50
Hourly pay rate successfully calculated
Tax Rate Code:
```

Valid Test Case 6:

```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel
Social Security #: 123456789
SSN successfully taken
Employee Id: 123N
Employee # successfully taken
Number of hours worked in a week:0
# of hours worked successfully taken
Hourly Pay Rate:50
Hourly pay rate successfully calculated
Tax Rate Code: 1
Tax rate of 25% selected successfully
Work Status: F
Work status successfully updated to Full Time
```

Valid Test Case 7:

```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel
Social Security #: 123456789
SSN successfully taken
Employee Id: 123N
Employee # successfully taken
Number of hours worked in a week:59
# of hours worked successfully taken
Hourly Pay Rate:
```


Valid Test Case 8:

```
Number of hours worked in a week:0
# of hours worked successfully taken
Hourly Pay Rate:50
Hourly pay rate successfully calculated
Tax Rate Code: 1
Tax rate of 25% selected successfully
Work Status: F
Work status successfully updated to Full Time
Name: Nisarg Patel
Employee Id: 123-N
Social Security Number: XXX-XX-6789
Number of hours worked in a week: 0
Weekly Pay: $0
```

Invalid Test Case 1:

```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel
Social Security #: 12345abcd
Sorry, Social security number is not in right format.
Try again:
Social Security #:
```

Invalid Test Case 2:

```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel
Social Security #: 123456789
SSN successfully taken
Employee Id: 1234
Sorry, Id is not in right format. Try again:
Employee Id:
```

Invalid Test Case 3:

```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel

Social Security #: 123456789

SSN successfully taken
Employee Id: ABCD

Sorry, Id is not in right format. Try again:
Employee Id:
```

Invalid Test Case 4:

```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel

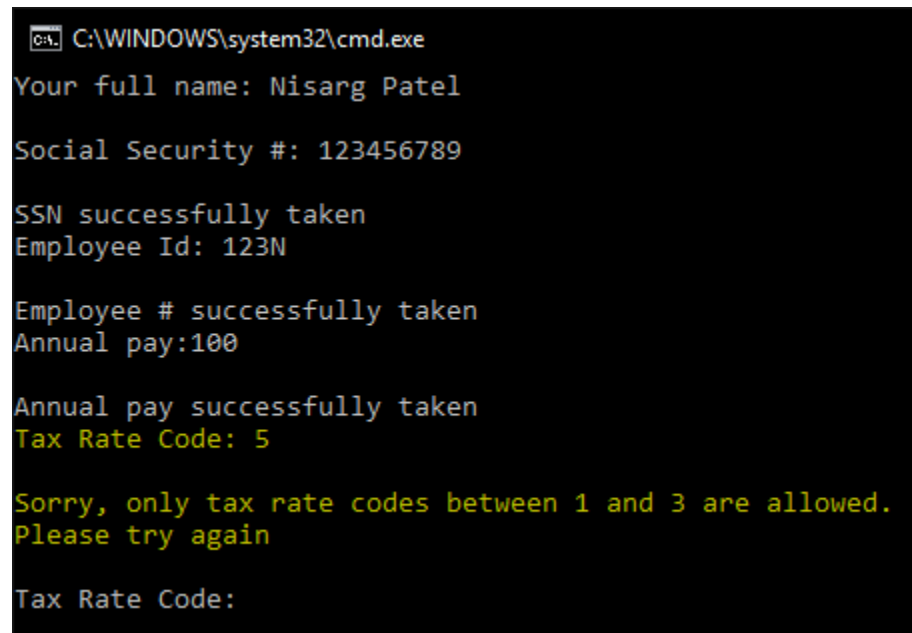
Social Security #: 123456789

SSN successfully taken
Employee Id: 123N

Employee # successfully taken
Annual pay:-20
Sorry, only positive values accepted.
Please try again.

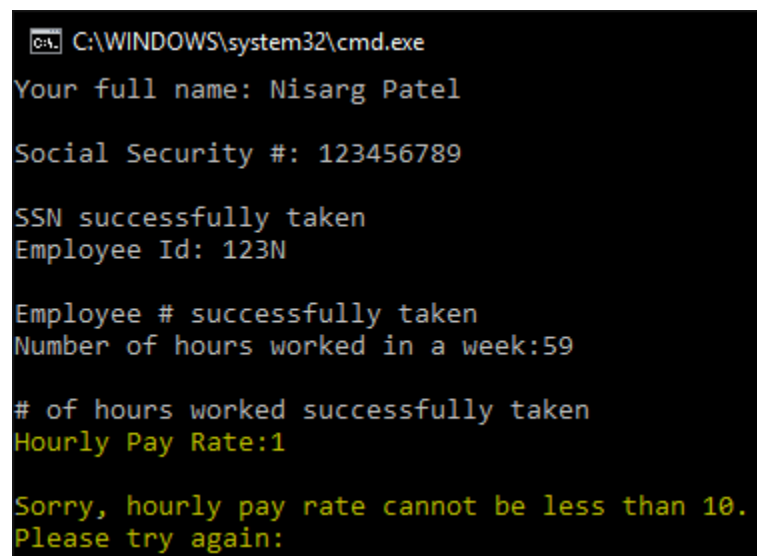
Annual pay:
```

Invalid Test Case 5:



```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel
Social Security #: 123456789
SSN successfully taken
Employee Id: 123N
Employee # successfully taken
Annual pay:100
Annual pay successfully taken
Tax Rate Code: 5
Sorry, only tax rate codes between 1 and 3 are allowed.
Please try again
Tax Rate Code:
```

Invalid Test Case 6:



```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel
Social Security #: 123456789
SSN successfully taken
Employee Id: 123N
Employee # successfully taken
Number of hours worked in a week:59
# of hours worked successfully taken
Hourly Pay Rate:1
Sorry, hourly pay rate cannot be less than 10.
Please try again:
```

Invalid Test Case 7:

```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel

Social Security #: 123456789

SSN successfully taken
Employee Id: 123N

Employee # successfully taken
Number of hours worked in a week:59

# of hours worked successfully taken
Hourly Pay Rate:100

Sorry, hourly pay rate cannot be more than 75.
Please try again:
```

Invalid Test Case 8:

```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel

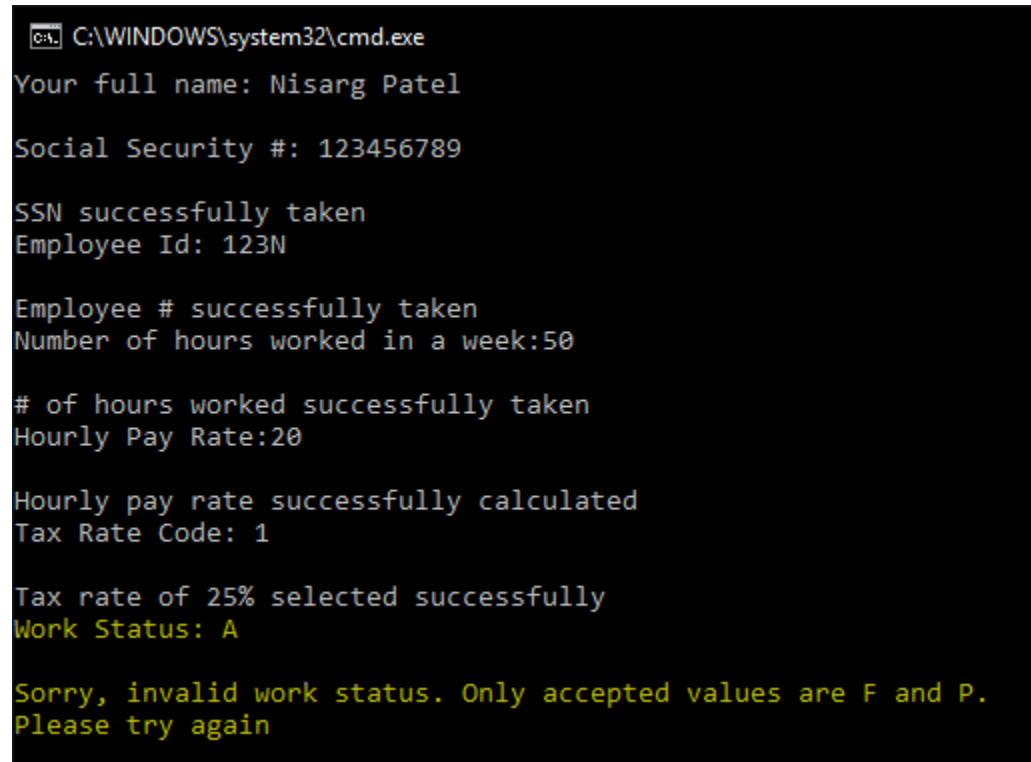
Social Security #: 123456789

SSN successfully taken
Employee Id: 123N

Employee # successfully taken
Number of hours worked in a week:100
Sorry, number of hours worked in a week can only be positive and less than 60.
Please try again.

# of hours worked successfully taken
Number of hours worked in a week:
```

Invalid Test Case 9:



```
C:\WINDOWS\system32\cmd.exe
Your full name: Nisarg Patel
Social Security #: 123456789
SSN successfully taken
Employee Id: 123N
Employee # successfully taken
Number of hours worked in a week:50
# of hours worked successfully taken
Hourly Pay Rate:20
Hourly pay rate successfully calculated
Tax Rate Code: 1
Tax rate of 25% selected successfully
Work Status: A
Sorry, invalid work status. Only accepted values are F and P.
Please try again
```

15. Status

Program works perfectly with assumptions in mind. All test cases passed with screenshots of all possible inputs from a user.