CISP 41

Programming in C#

Project Evaluation Sheet

Student Name:	__ Paola Socorro, Zixing Qiao, Qianqun Xu_______	Project Number:	_____4__________
Project Name:	___Payroll Program 4_______	Visual Studio Version:	_____2010______
Date Due:	_______June 6, 2013________	Date Turned In:	________June 6th___

Above to be completed by student

Points (____ Possible)

Correctness/Efficiency:

Output is accurate	____________
Meets all requirements	____________
Provide appropriate user interface	____________
Logic is efficient	____________

Documentation/Coding Style:

Project can be open from the submitted zip file	____________
Folder is present and contains all necessary project files (no extra files)	____________
Use required coding template	____________
Use proper naming and spacing	____________
Submit all requested information	____________

Test Cases:

List all required test cases	____________
Provide output forms for important test cases	____________

Other issues: ____________

Extra Credit: ____________

Timeliness: ____________

Project Score: $\boxed{}$

CISP 41 Programming in C#

Project specification

***

- **Database - made by Zixing Qiao.**
-
- **ReportClass, Paystub report, and w_report derived class - made by Qianqun Xu.**
-
- **Payroll Class, User Interface, Forms, Payroll_Form code, splashcreen, graphics - made by Paola Socorro.**
-
- **Note(s): Small changes made to Database to correct spelling errors, Originally also made code to read and gather data from datafiles, code was replaced by the use of a database. - both by PSocorro.**
- **Project database located: Payroll_Project04\Project4\Resources**

***

This program should calculate the salary per month of a company's employees. It calculates both for employees paid per hour, and for those with a fixed salary regardless of hours.

The program deducts the cost for medical and dental benefits, life insurance, 401k account, and FSA.

It also takes into account taxes like, social security tax (10%), federal income tax(15%) and state income tax (5%).

for each employee the program also creates monthly pay stubs. as well as w-2's at the end of the year.

a database is used to keep record of the employees and their individual information like id number, name, pay per hour or salary and benefits. Etc.

The database has a master table for employee data, and a separate table to keep monthly data of hours worked.

Hours worked by salaried employees are for record keeping only. Their net pay is calculated, based on the gross monthly salary.

medical and dental benefits are deducted first, before tax deductions. Tax is based on the subsequent amount.

All commits available at GitHub
https://github.com/ZenRumi/Payroll_Project04

Limitations:
Program cannot place reports in a separate window for user view before printing.
All forms go to print preview at once.
program cannot create paystub per employee, one at a time. All are created at once, one after the other.
Program cannot update, change database in anyway.
cannot add employee, remove employee
cannot create reports for quarters.
program does not take into account the occasional switch of idnumber's position in month files. (possible error noticed but had no time to implement a solution. My idea was to search the array first and place values in the proper place according to id number position.)

CISP 41 **Programming in C#**

Project status

**Project processes all 12 months, produces pay stubs and w-2's for each employee. **

Updates from GitHub uploads:

May 20th
ZenRumi authored

May 21st
```
Testform is for my use not the final UI. Included data files.
```

May 30th
```
Changed UI,

Created standardForm

created payrollclass

added master file and monthly files to resources.

fixed flashscreen.

updated payroll form code: now reads master file and splits lines into

arrays for later use.
```

May 31st (database .mdf file and log file emailed to psocorro)

Jun02
ZenRumi authored
```
must attach database to SQL SERVER 2008 or above.

connection name should be: project4dataConnectionString.

database located in resources folder.
```

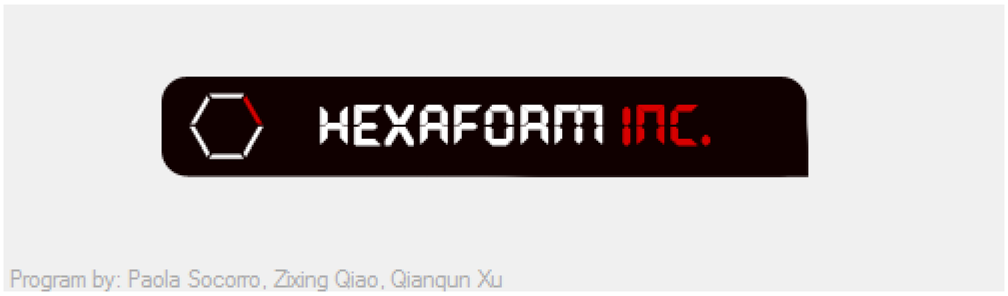
June 03
```
Program now processes each month selected in the combobox.
```

June 04 (reports code emailed to PSocorro Monday June 3rd)
```
W-2 and Paystub reports done by Joe added in.
```

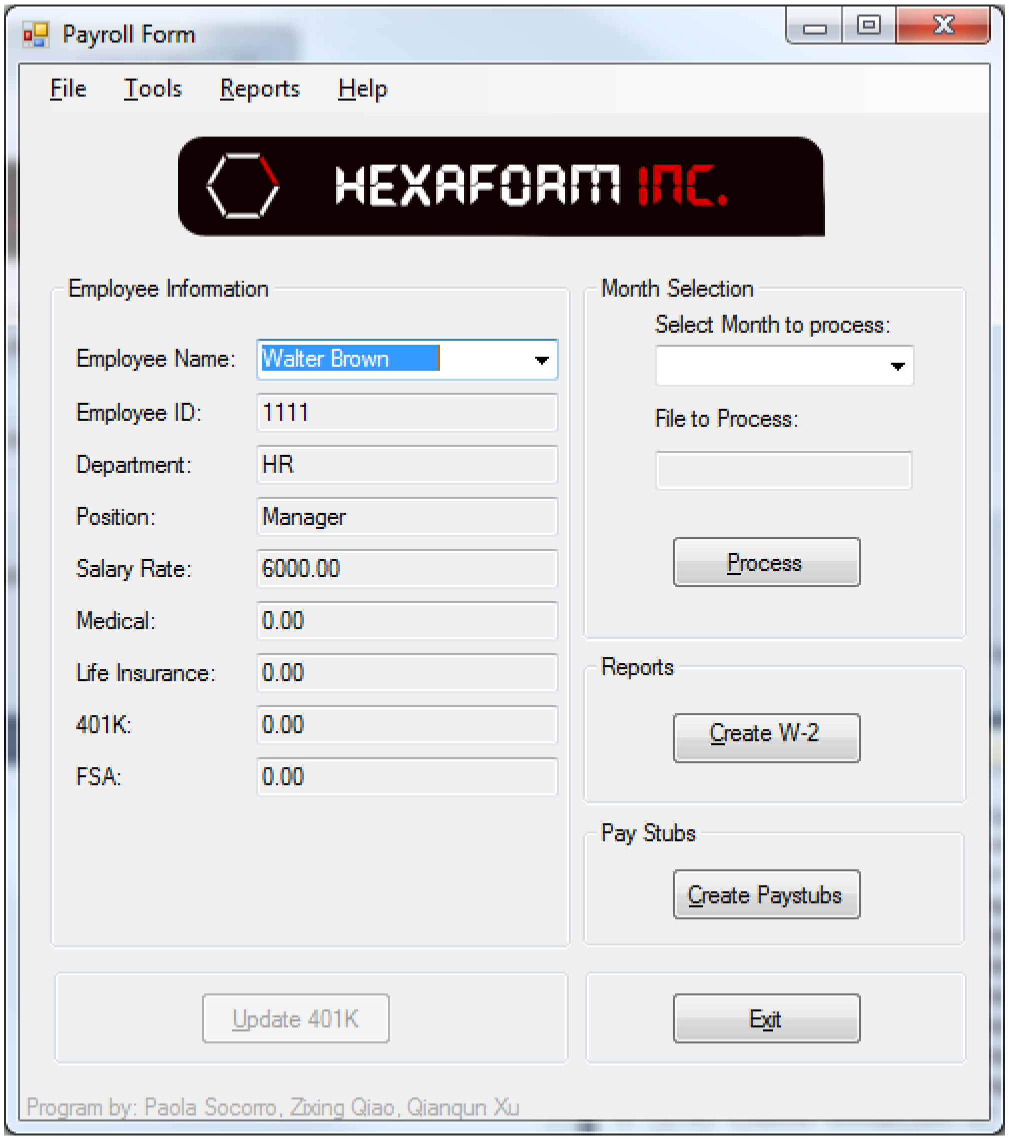
June 5th (9th commit)
```
reports implement. User will now be able to print pay stubs and w-2s at

the end of the year.
```

Sketch of user interface

Splash Screen.
Graphic: Designed by Paola Socorro. Hexaform Inc.

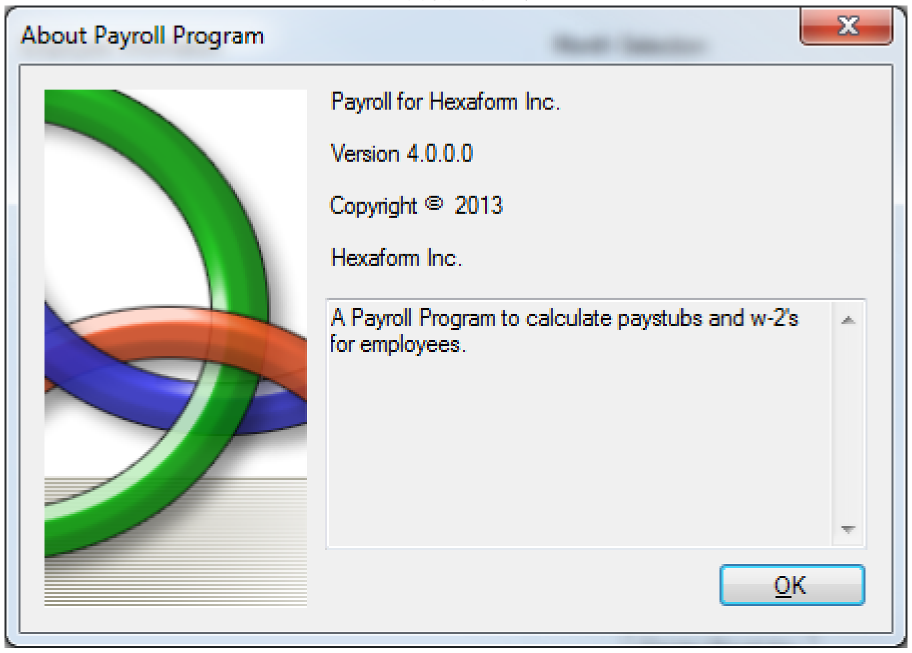


Main form



Employee Name:	Walter Brown
Employee ID:	1111
Department:	HR
Position:	Manager
Salary Rate:	6000.00
Medical:	0.00
Life Insurance:	0.00
401K:	0.00
FSA:	0.00

About Form.



Payroll for Hexaform Inc.

Version 4.0.0.0

Copyright © 2013

Hexaform Inc.

A Payroll Program to calculate paystubs and w-2's for employees.

CISP 41 **Programming in C#**

Objects and Properties Plan for ___________Form

Object	Property	Setting
pictureBox1	InitialImage	Project4.Properties.Resources.hexaformLogo
label1	Text	Program by: Paola Socorro, Zixing Qiao, Qianqun Xu
StandardForm	Size StartPosition Text	500, 385 CenterScreen StandardForm
menuStrip1	Size Text	484,24 menuStrip1
Payroll_Form: StandardForm	MainMenuStrip Size	menuStrip1 500,565
employee_NameLabel1	Name	Employee Name
employeeIDLabel	Name	Employee ID
departmentLabel	Name	Department
positionLabel	Name	Position
salary_RateLabel	Name	Salary Rate
medicalLabel	Name	Medical
life_InsuranceLabel	Name	Life Insurance
_401KLabel	Name	401K
fSALabel	Name	FSA
employee_NameComboBox	DataSource DisplayMember	master_TransactionsBindingSource Employee Name
EmployeeIDTextBox	ReadOnly	True
DepartmentTextBox	ReadOnly	True
PositionTextBox	ReadOnly	True
salary_RateTextBox	ReadOnly	True
MedicalTextBox	ReadOnly	True
life_InsuranceTextBox	ReadOnly	True
_401KTextBox	ReadOnly	True
fSATextBox	ReadOnly	True
monthsComboBox	Name	monthsComboBox
processingTextBox	Name ReadOnly	processingTextBox True
processButton	Name Text	processButton &Process
reportButton	Name Text	reportButton &Create W-2

checksButton	**Name** **Text**	**ChecksButton** **&Create Paystubs**
EXITbutton	**Name** **Text**	**EXITbutton** **E&xit**
update401kButton	**Name** **Text** **Enabled**	**update401kButton** **&Update 401K** **False**
masterTransactionsBindingSource	**DataMember** **DataSource**	**Master_Transactions** **project4dataDataSetBindingSource**
master_TransactionsTableAdapter	**ClearBeforeFill**	**True**
jan_TransactionsBindingSource (through dec)	**DataMember** **DataSource**	**Jan_Transactions (through Dec)** **project4dataDataSet**
jan_TransactionsTableAdapter (through dec)	**ClearBeforeFill**	**True**
project4dataDataSet	**DataSetName** **SchemaSerialization**	**project4dataDataSet** **IncludeSchema**
printDocument1	**Name**	**printDocument1**
printPreviewDialog1	**Name**	**printPreviewDialog1**

CISP 41 **Programming in C#**

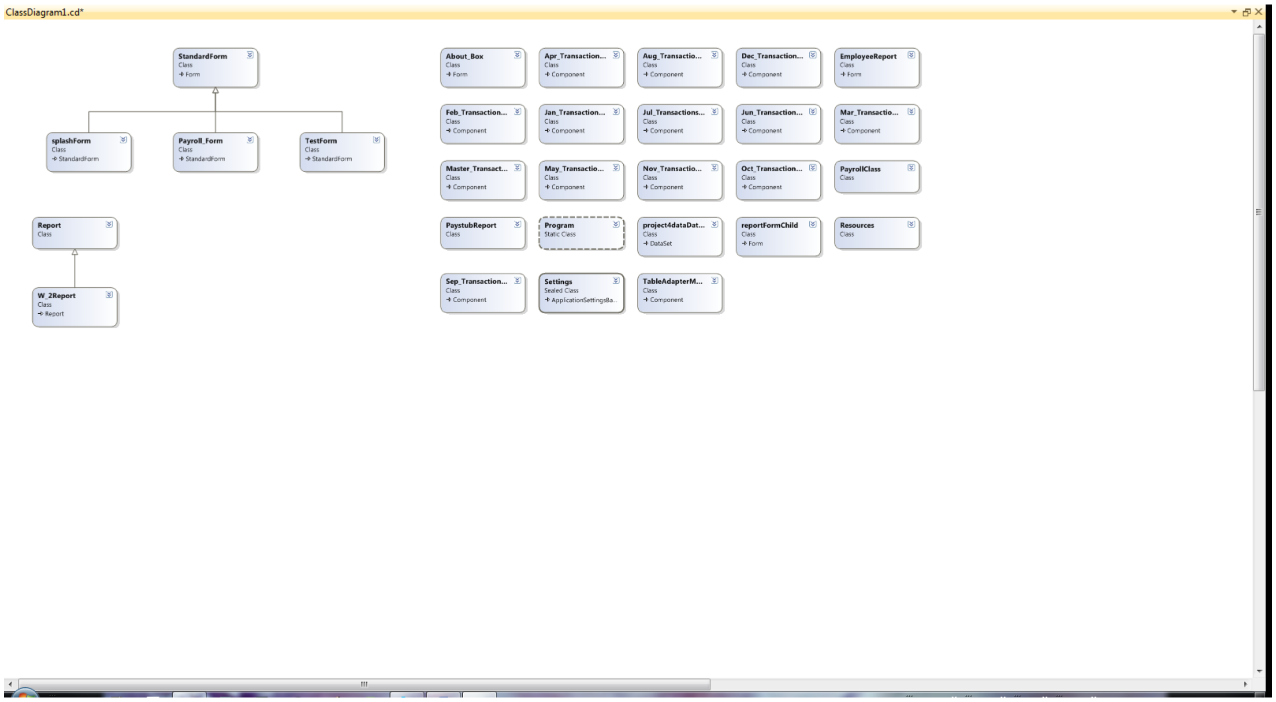
Event Plan for ___________Form

Object	**Event**	**Action - Pseudocode**
processButton	processButton_Click	Try: dataRowReaderMonth() Catch: IndexOutOfRangeException

reportButton
w2ReportToolStripMenuItem_Click
	w2ReportToolStripMenuItem_Click	checkForMonthProcess();
		
		if allMonthsProcessBool==true
		for loop go through employeeID array
		if index in employeeID array is not null
		create w-2 to print preview
		else
		break;

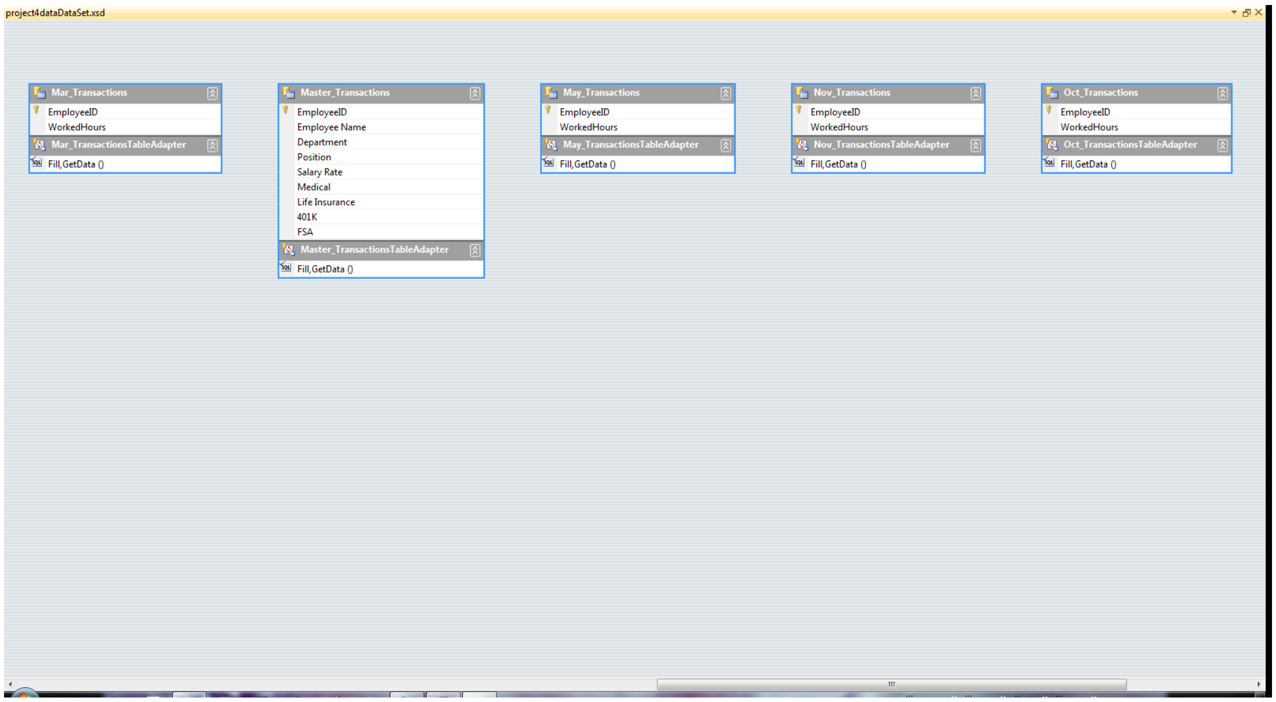
checksButton monthlyPayStubsToolStripMenuItem_Click	monthlyPayStubsToolStripMenuItem_Click	Same as report button Creat paystub report instead.
EXITbutton	EXITbutton_Click	Close the form
update401kButton	None	none
aboutToolStripMenuItem	aboutToolStripMenuItem_Click	Open about form
PayrollClass		Business class that does the payroll calculations and deductions.
Report		Report class that handles creating different reports(base class)
PaystubReport		Handles creating paystubs to print preview
W_2Report:Report		Inherits from Report. Handles creating w-2 to print preview

Class diagram



Sample of database created

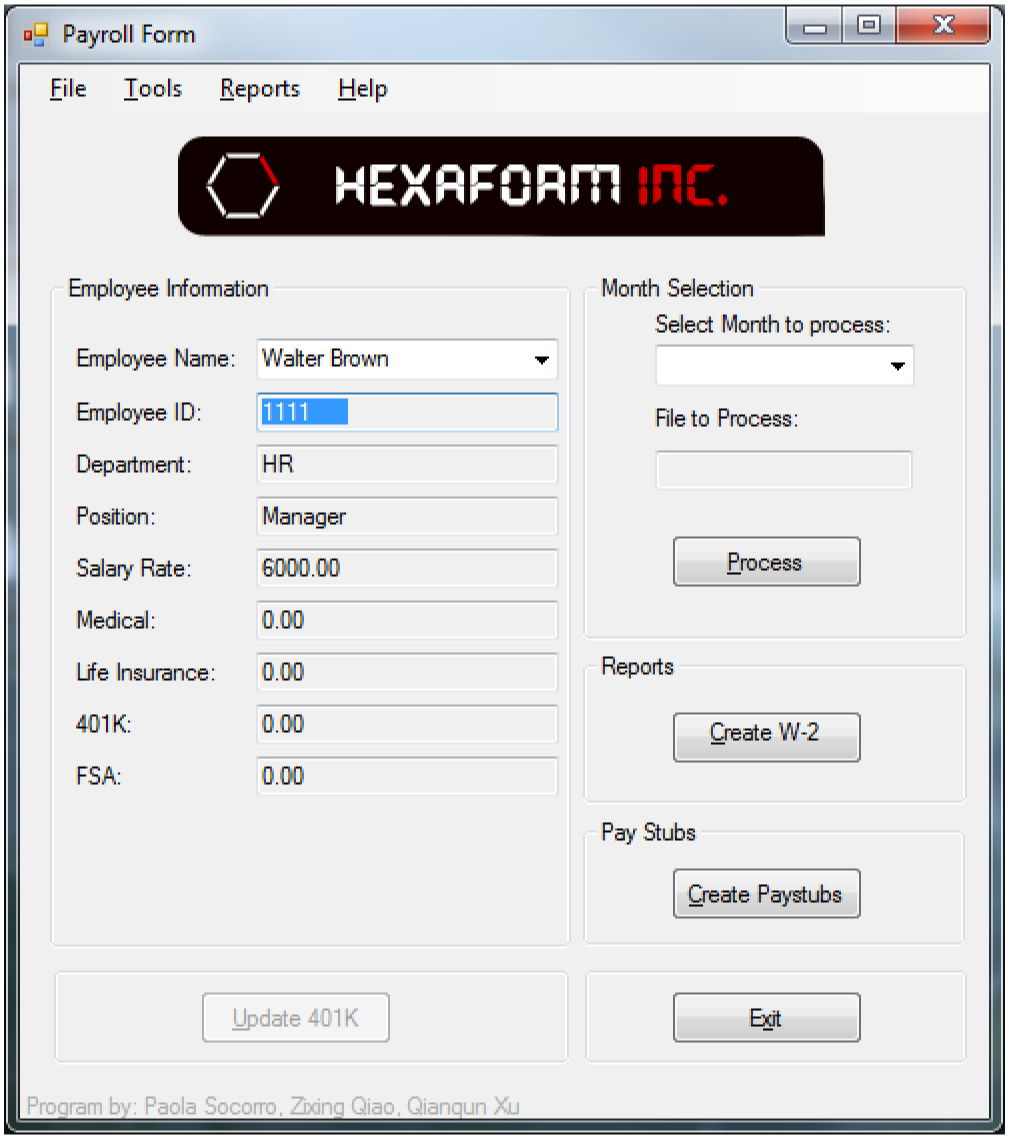
by Zixing Qiao



CISP 41 **Programming in C#**

Test cases and captured screens

Payroll Form:
Contains employee information for individual viewing.
(originally intended to update 401k, add and remove employee. features not implemented yet)

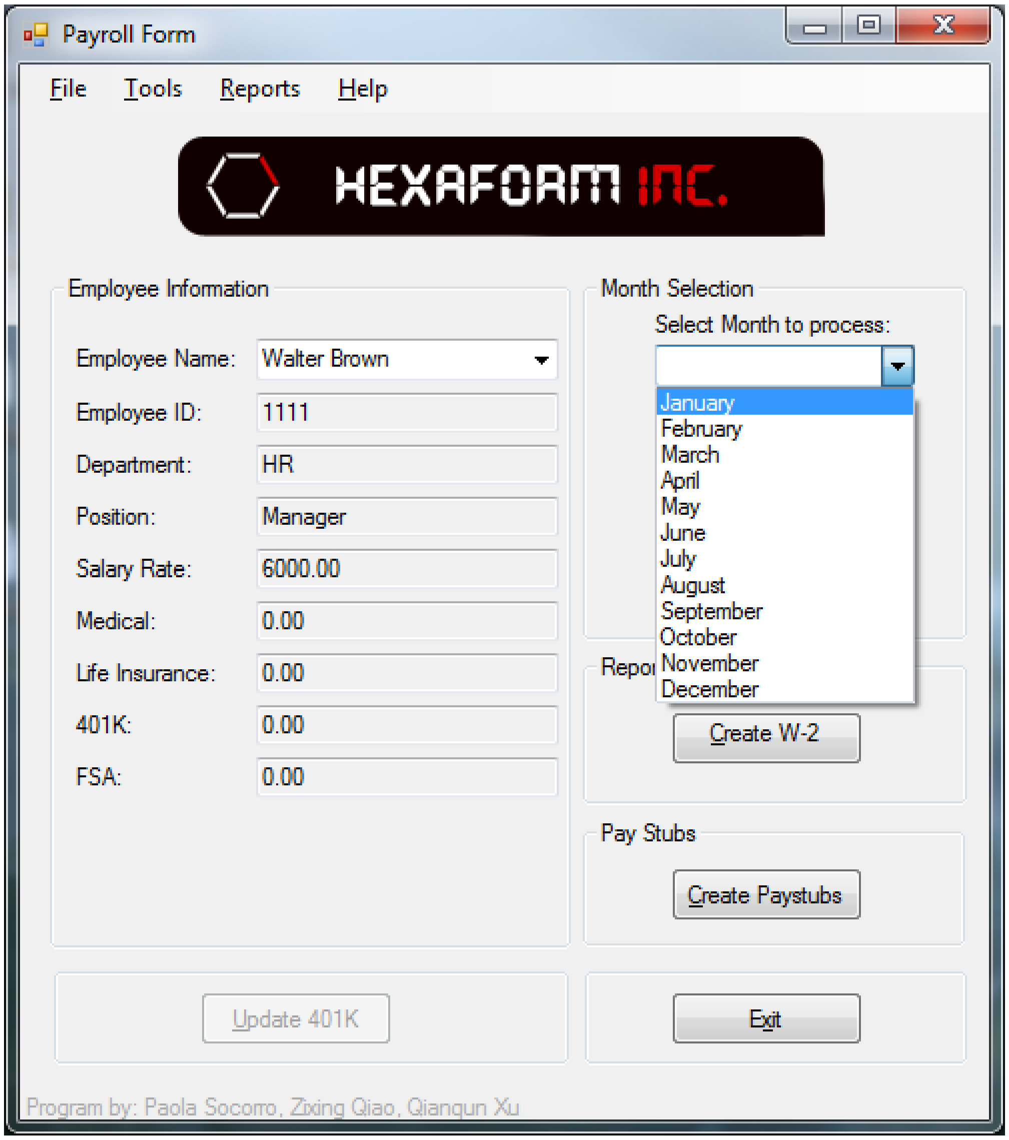


Employee Name: Walter Brown
Employee ID: 1111
Department: HR
Position: Manager
Salary Rate: 6000.00
Medical: 0.00
Life Insurance: 0.00
401K: 0.00
FSA: 0.00

Select Month to process:
File to Process:

Program by: Paola Socorro, Zixing Qiao, Qianqun Xu

Select Month to process from drop down menu.
Process month desired to create paystubs for.


Payroll Form

File Tools Reports Help

Employee Information

Employee Name: Walter Brown

Employee ID: 1111

Department: HR

Position: Manager

Salary Rate: 6000.00

Medical: 0.00

Life Insurance: 0.00

401K: 0.00

FSA: 0.00

Month Selection

Select Month to process:

January
February
March
April
May
June
July
August
September
October
November
December

Create W-2

Pay Stubs

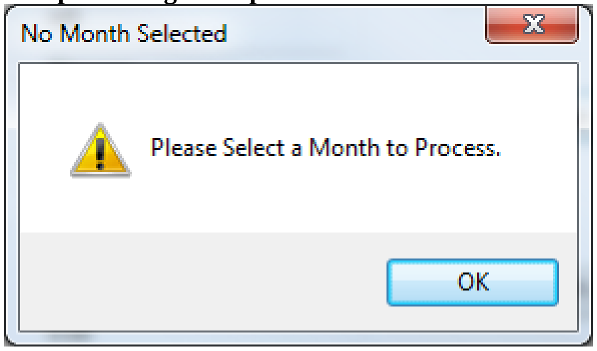
Create Paystubs

Update 401K

Exit

Program by: Paola Socorro, Zixing Qiao, Qianqun Xu

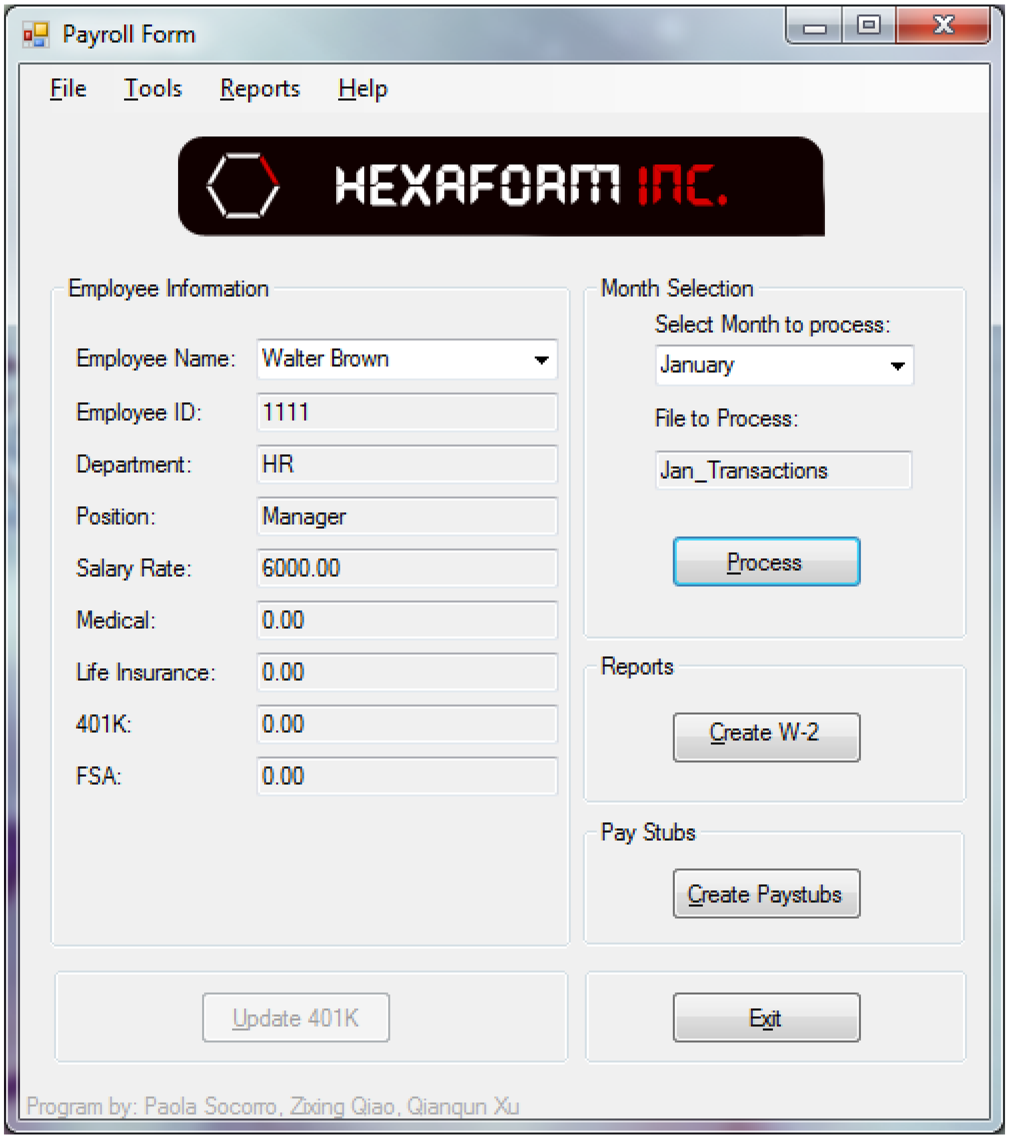
no month selected brings up message box
no processing takes place until month is selected.


No Month Selected

Please Select a Month to Process.

OK

Time to create paystubs for the month processed.



Payroll Form

Employee Information

Employee Name:	Walter Brown
Employee ID:	1111
Department:	HR
Position:	Manager
Salary Rate:	6000.00
Medical:	0.00
Life Insurance:	0.00
401K:	0.00
FSA:	0.00

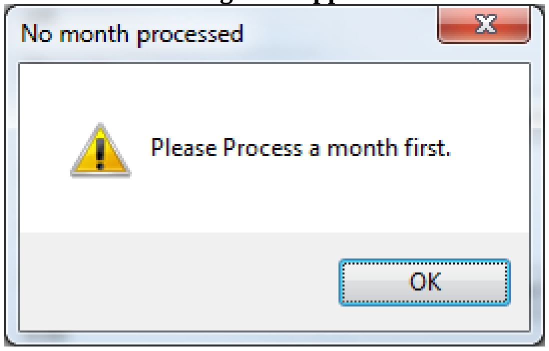
Month Selection

Select Month to process: January

File to Process: Jan_Transactions

Program by: Paola Socorro, Zixing Qiao, Qianqun Xu

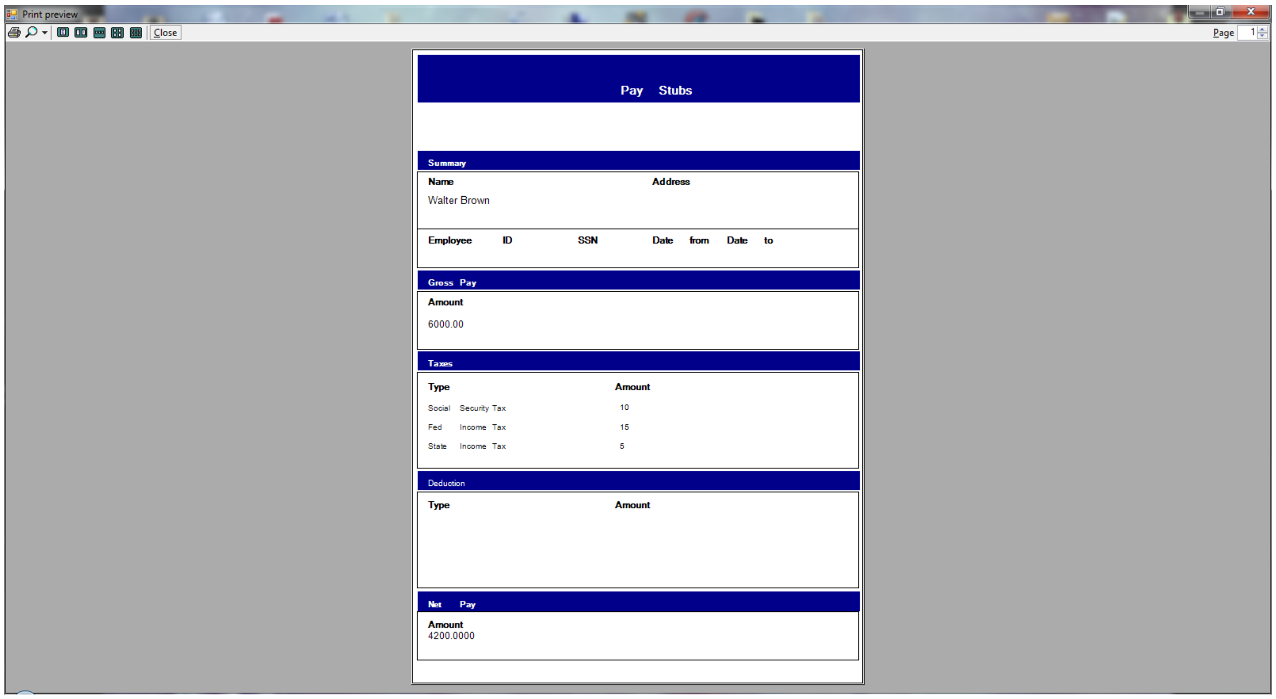
if no month has been processed yet message box appears.



No month processed

Please Process a month first.

Paystubs are created and go directly to print preview one print preview per paystub.



Pay Stubs

Summary

Name	Address
Walter Brown	

Employee ID	SSN	Date from	Date to

Gross Pay

Amount

6000.00

Taxes

Type	Amount
Social Security Tax	10
Fed Income Tax	15
State Income Tax	5

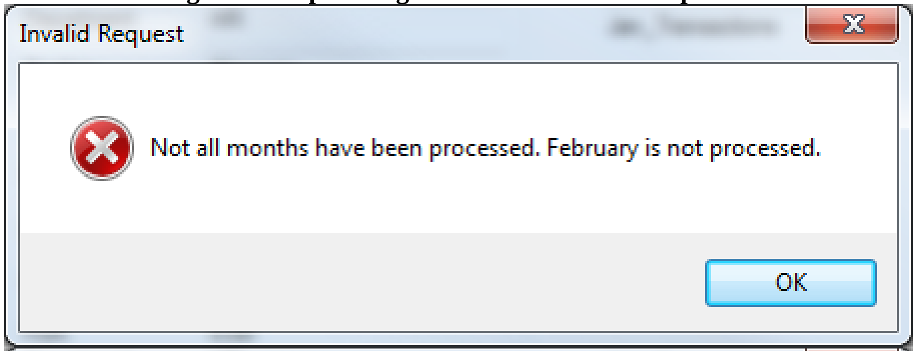
Deduction

Type	Amount

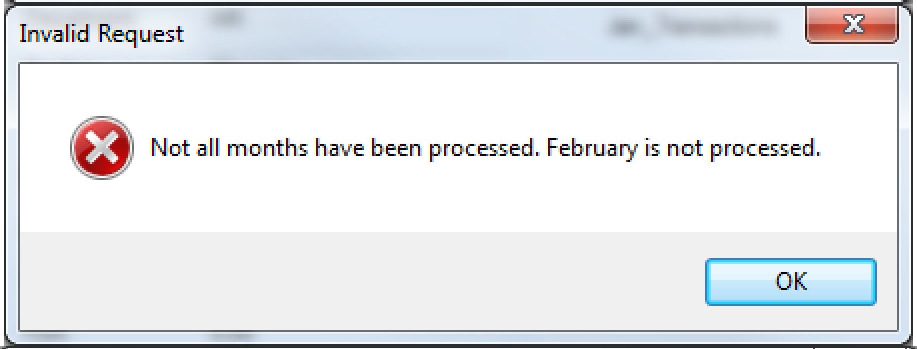
Net Pay

Amount
4200.0000

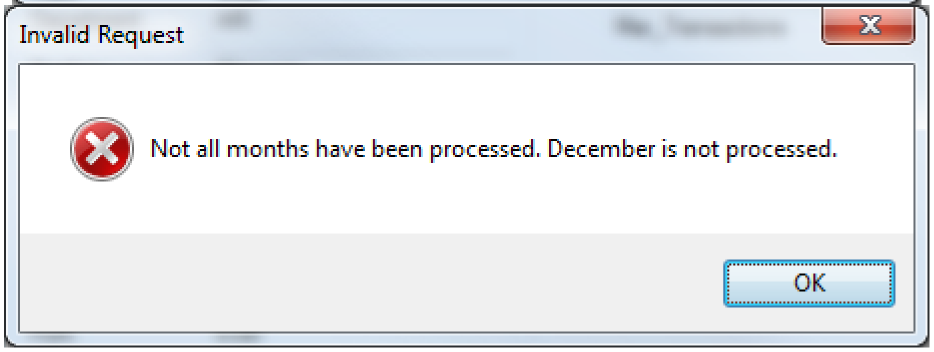
If user attempts to create w-2 now an error message comes up message comes up as long as there is a month not processed.



Not all months have been processed. February is not processed.

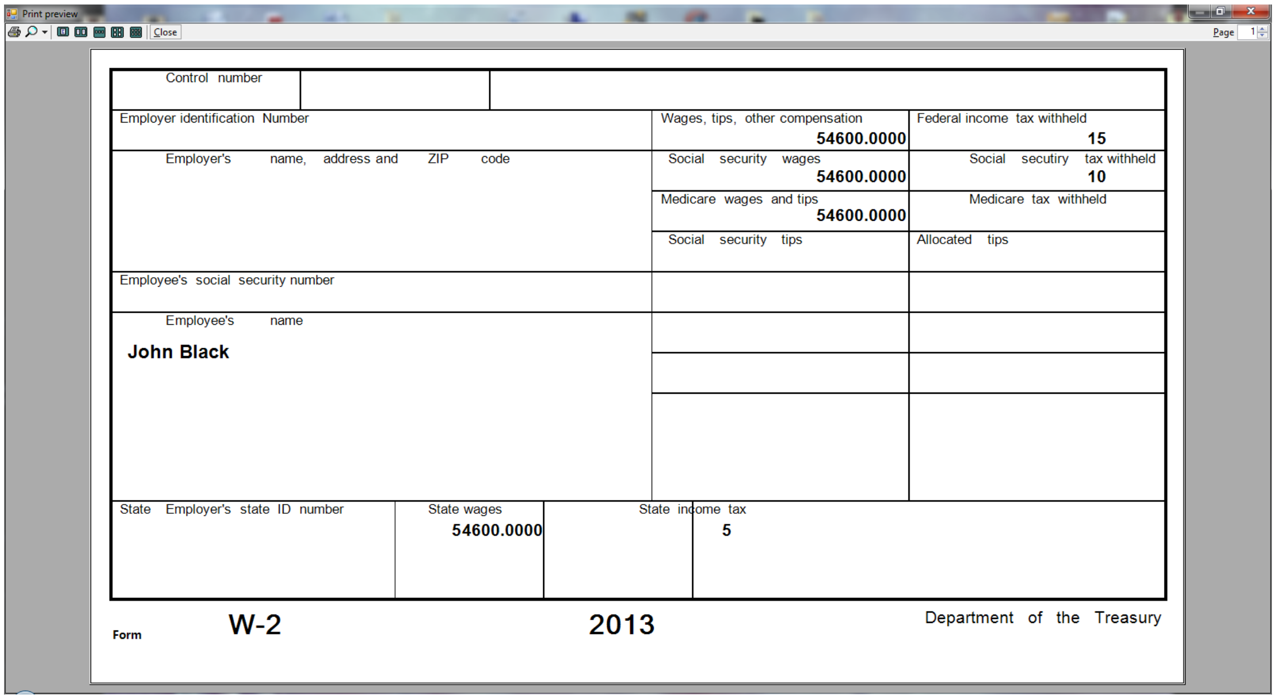


Not all months have been processed. February is not processed.



Not all months have been processed. December is not processed.

Once all months are processed
w-2 is created to the print preview.
one per employee.



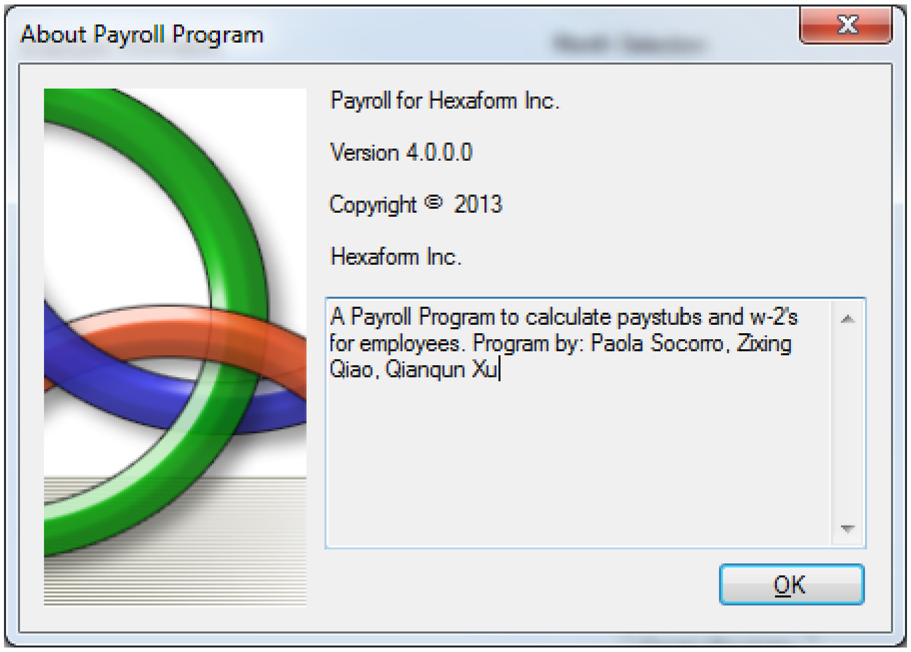
Control number		
Employer identification Number	Wages, tips, other compensation 54600.0000	Federal income tax withheld 15
Employer's name, address and ZIP code	Social security wages 54600.0000	Social secutiry tax withheld 10
	Medicare wages and tips 54600.0000	Medicare tax withheld
	Social security tips	Allocated tips
Employee's social security number		
Employee's name **John Black**		

State Employer's state ID number	State wages 54600.0000	State income tax 5	

Form **W-2** **2013** Department of the Treasury

Both reports can be accessed through the menu.

our about form


About Payroll Program

Payroll for Hexaform Inc.

Version 4.0.0.0

Copyright © 2013

Hexaform Inc.

A Payroll Program to calculate paystubs and w-2's for employees. Program by: Paola Socorro, Zixing Qiao, Qianqun Xu

CISP 41 Programming in C#

Source code

Main Form - by Paola Socorro

```
/*  Program:     Payroll Project 4
    Author:      Paola Socorro (for this PayrollClass)
    Class:      CISP 41
    Date:       May 18 2013
    Description: The business part of the payroll program that handles user input, as well
as calculations.

    I certify that the code below is my own work.

   Exception(s): Database - made by Zixing Qiao. ReportClass, Paystub report, and w_report
derived class - made by Qianqun Xu.
 *                  Payroll Class, User Interface, Forms, Payroll_Form code, splashcreen,
graphics - made by Paola Socorro.
 *
 *   Note(s): Small changes made to Database to correct spelling errors,
 *           Originally also made code to read and gather data from datafiles, code was
replaced by the use of a database. - both by PSocorro.

*/


using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace Project4
{
    public partial class Payroll_Form : StandardForm
    {
        //
        //CODE by Paola Socorro
        //

        PayrollClass aPayroll = new PayrollClass();

        const int MAXIMUM_EMPLOYEEINT =20;

        string[] months = { "January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December" };
        string[] monthlyData = { "Jan_Transactions", "Feb_Transactions", "Mar_Transactions",
"Apr_Transactions", "May_Transactions", "Jun_Transactions", "Jul_Transactions",
"Aug_Transactions", "Sep_Transactions", "Oct_Transactions", "Nov_Transactions",
"Dec_Transactions" };
        string[] isMonthProcess = { "no", "no", "no", "no", "no", "no", "no", "no", "no",
"no", "no", "no" };
        string monthToProcess;
        bool allMonthsProcessedBool=false;
        string[] employeeIdArray= new string[MAXIMUM_EMPLOYEEINT];
        decimal[] eNetPay = new decimal[MAXIMUM_EMPLOYEEINT];
        decimal[] eGrossPay = new decimal[MAXIMUM_EMPLOYEEINT];
        decimal[] eDeductedTax = new decimal[MAXIMUM_EMPLOYEEINT];
        decimal[] w2TotalPay = new decimal[MAXIMUM_EMPLOYEEINT];

        decimal socialSecurityTaxDec = 10;
        decimal fedIncomeTaxDec = 15;
        decimal stateIncomeTaxDec = 5;



        string idNumber;
        string nameLastname;
        string departmentString;
        string positionString;
        decimal rateDec;
        decimal monthlySalaryDec; //just for salary employees
        decimal benefitsDec; //medical and dental benefits both into one.
        decimal lifeInsur;
        decimal savings401K; // not to be confused with FSA
        decimal fsa;

        decimal hoursWorked;



        public Payroll_Form()
        {
            InitializeComponent();
        }

        private void EXITbutton_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        //
        //Loads the months array into the combobox.
        //
        private void Payroll_Form_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the
'project4dataDataSet1.Dec_Transactions' table. You can move, or remove it, as needed.

this.dec_TransactionsTableAdapter.Fill(this.project4dataDataSet.Dec_Transactions);
            // TODO: This line of code loads data into the
'project4dataDataSet1.Nov_Transactions' table. You can move, or remove it, as needed.

this.nov_TransactionsTableAdapter.Fill(this.project4dataDataSet.Nov_Transactions);
            // TODO: This line of code loads data into the
'project4dataDataSet1.Oct_Transactions' table. You can move, or remove it, as needed.

this.oct_TransactionsTableAdapter.Fill(this.project4dataDataSet.Oct_Transactions);
            // TODO: This line of code loads data into the
'project4dataDataSet1.Sep_Transactions' table. You can move, or remove it, as needed.

this.sep_TransactionsTableAdapter.Fill(this.project4dataDataSet.Sep_Transactions);
            // TODO: This line of code loads data into the
'project4dataDataSet1.Aug_Transactions' table. You can move, or remove it, as needed.

this.aug_TransactionsTableAdapter.Fill(this.project4dataDataSet.Aug_Transactions);
            // TODO: This line of code loads data into the
'project4dataDataSet1.Jul_Transactions' table. You can move, or remove it, as needed.
```

```
this.jul_TransactionsTableAdapter.Fill(this.project4dataDataSet.Jul_Transactions);
            // TODO: This line of code loads data into the
'project4dataDataSet1.Jun_Transactions' table. You can move, or remove it, as needed.

this.jun_TransactionsTableAdapter.Fill(this.project4dataDataSet.Jun_Transactions);
            // TODO: This line of code loads data into the
'project4dataDataSet1.May_Transactions' table. You can move, or remove it, as needed.

this.may_TransactionsTableAdapter.Fill(this.project4dataDataSet.May_Transactions);
            // TODO: This line of code loads data into the
'project4dataDataSet1.Apr_Transactions' table. You can move, or remove it, as needed.

this.apr_TransactionsTableAdapter.Fill(this.project4dataDataSet.Apr_Transactions);
            // TODO: This line of code loads data into the
'project4dataDataSet1.Mar_Transactions' table. You can move, or remove it, as needed.

this.mar_TransactionsTableAdapter.Fill(this.project4dataDataSet.Mar_Transactions);
            // TODO: This line of code loads data into the
'project4dataDataSet1.Feb_Transactions' table. You can move, or remove it, as needed.

this.feb_TransactionsTableAdapter.Fill(this.project4dataDataSet.Feb_Transactions);
            // TODO: This line of code loads data into the
'project4dataDataSet.Master_Transactions' table. You can move, or remove it, as needed.

this.master_TransactionsTableAdapter.Fill(this.project4dataDataSet.Master_Transactions);
            // TODO: This line of code loads data into the
'project4dataDataSet.Master_Transactions' table. You can move, or remove it, as needed.

this.master_TransactionsTableAdapter.Fill(this.project4dataDataSet.Master_Transactions);


this.jan_TransactionsTableAdapter.Fill(this.project4dataDataSet.Jan_Transactions);
            //
            //FILL employeeIdArray at runtime.
            //
            populateIdArray();
            Console.WriteLine("array filled");
            for (int i = 0; i < months.Length; i++)
            {
                monthsComboBox.Items.Add(months[i]);
            }

        }
        //
        //DISPLAYS file to be processed to the user. Assigns selected index to variable.
        //
        private void monthsComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            processingTextBox.Text = monthlyData[monthsComboBox.SelectedIndex];
            monthToProcess = monthlyData[monthsComboBox.SelectedIndex];
            //Console.WriteLine("To process: " +
monthlyData[monthsComboBox.SelectedIndex].ToString());
        }
        //
        //READ DATA FROM MASTER TABLE.
        //Places each row of data into variables.
        //
        private void populateIdArray()
        {
            int counter =0;
            foreach (DataRow row in project4dataDataSet.Master_Transactions.Rows)
            {
                string idNum = row["EmployeeID"].ToString();
                idNum = idNum.Replace(" ", null);//removing spaces in string
                employeeIdArray[counter] = idNum;
                counter++;
            }
        }
        //
        //Reads employee information from Master_transactions.
        //Places data in apropriate variables.
        //
        private void dataRowReaderMaster()
        {
            foreach (DataRow row in project4dataDataSet.Master_Transactions.Rows)
            {
                string idNum = row["EmployeeID"].ToString();
                idNum = idNum.Replace(" ", null);//removing spaces in string
                if (idNum == idNumber)
                {
                    nameLastname = row["Employee Name"].ToString();
                    //nameLastname = nameLastname.Replace(" ", null);
                    departmentString = row["Department"].ToString();
                    positionString = row["Position"].ToString();
                    positionString = positionString.Replace(" ", null);//removing spaces in
string

                    rateDec = decimal.Parse(row["Salary Rate"].ToString());
                    benefitsDec = decimal.Parse(row["Medical"].ToString());
                    lifeInsur = decimal.Parse(row["Life Insurance"].ToString());
                    savings401K = decimal.Parse(row["401k"].ToString());
                    fsa = decimal.Parse(row["FSA"].ToString());
                }
            }

            //
            //Compares position to check if its Manager or Engineer.
            //Manger is used, due to error in database values for one employee. Cindy Red.
            //
            if (positionString == "Manager" || positionString == "Engineer" ||
positionString == "Manger")
            {
                monthlySalaryDec = rateDec;
                rateDec = 0;
               // Console.WriteLine("done" + monthlySalaryDec.ToString() + " set");
            }

        }

        private void dataRowReaderMonth()
        {

            //processingTextBox.Text = monthlyData[monthsComboBox.SelectedIndex];
            //monthToProcess = monthlyData[monthsComboBox.SelectedIndex];

            int counter =0;
            //int monthCounter = 0;
            string tName; // name of the stable.
            //string month = monthlyData[counter];
            int tableCount= monthsComboBox.SelectedIndex;

            foreach (DataTable table in project4dataDataSet.Tables)
            {
                if (table.TableName.ToString() == monthlyData[tableCount])
                {
                    tName = table.TableName;
                    Console.WriteLine(monthlyData[tableCount].ToString() + "\n");
                    foreach (DataRow row in table.Rows)
```

```
                        {
                            idNumber = row["EmployeeID"].ToString();
                            idNumber = idNumber.Replace(" ", null);//removing spaces in string
                            hoursWorked = decimal.Parse(row["WorkedHours"].ToString());
                            dataRowReaderMaster();
                            calcEmployeePay(counter);
                            //Console.WriteLine("Data Read for month of "+
months[counter].ToString());
                            Console.WriteLine("One Employee done: " + nameLastname.ToString() +
" id: " + idNumber.ToString() + " hours worked: " + hoursWorked.ToString());
                            counter++;

                        }
                        isMonthProcess[tableCount] = "yes";
                    }

                }


            }


            private void findEmployeeInData()
            {
                int counter = 0;
                string tName; // name of the stable.
                int tableCount = monthsComboBox.SelectedIndex;

                foreach (DataTable table in project4dataDataSet.Tables)
                {
                    if (table.TableName.ToString() == monthlyData[tableCount])
                    {
                        tName = table.TableName;
                        Console.WriteLine(monthlyData[tableCount].ToString() + "\n");
                        foreach (DataRow row in table.Rows)
                        {
                            idNumber = row["EmployeeID"].ToString();
                            idNumber = idNumber.Replace(" ", null);//removing spaces in string
                            dataRowReaderMaster();
                            counter++;

                        }

                    }

                }
            }//Not used ran out of time. It was supposed to find an employee in the database, to
output w-2 and paystubs correctly, and per person.

            private void calcEmployeePay(int step)
            {
                if (positionString == "Manager" || positionString == "Engineer" ||
positionString == "Manger")
                {

                    eGrossPay[step] = aPayroll.grossPaySalary(monthlySalaryDec);


                }
                else
                {
                    eGrossPay[step] = aPayroll.grossPay(rateDec, hoursWorked);

                }
                aPayroll.deductionPay(benefitsDec, lifeInsur, savings401K, fsa);
                eDeductedTax[step] = aPayroll.mandatoryTaxDeductions();
                eNetPay[step] = aPayroll.netPayCalc();
                w2TotalPay[step] += eNetPay[step];
                //Console.WriteLine("inside calcEmployeePay");
            }

            private void processButton_Click(object sender, EventArgs e)
            {
                //dataRowReaderMaster();
                //dataRowReaderMonth();
                try
                {
                    dataRowReaderMonth();
                    Console.WriteLine("done");
                }
                catch (IndexOutOfRangeException)
                {
                    MessageBox.Show("Please Select a Month to Process.","No Month
Selected",MessageBoxButtons.OK,MessageBoxIcon.Exclamation);

                }

            }


            private void testFormToolStripMenuItem_Click(object sender, EventArgs e)
            {
                TestForm myTestForm = new TestForm();
                myTestForm.ShowDialog();
            }

            private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
            {
                About_Box myAboutForm = new About_Box();
                myAboutForm.ShowDialog();
            }

            private void w2ReportToolStripMenuItem_Click(object sender, EventArgs e)
            {
                checkForMonthProcess();
                if (allMonthsProcessedBool == true)
                {
                    for (int i = 0; i < employeeIdArray.Length; i++)
                    {
                        if (employeeIdArray[i] != null)
                        {
                            W_2Report aW_2 = new W_2Report(nameLastname,w2TotalPay[i],
socialSecurityTaxDec, fedIncomeTaxDec, stateIncomeTaxDec);
                        }
                        else
                        {
                            break;
                        }
                    }
                }

            }

            private void monthlyPayStubsToolStripMenuItem_Click(object sender, EventArgs e)
            {
                if (monthsComboBox.SelectedIndex > -1)
                {
                    for (int i = 0; i < employeeIdArray.Length; i++)
                    {
                        if (employeeIdArray[i] != null)
                        {
                            idNumber = employeeIdArray[i];
                            dataRowReaderMaster();
                            PaystubReport aPSReport = new PaystubReport(nameLastname,
    eGrossPay[i], socialSecurityTaxDec, fedIncomeTaxDec, stateIncomeTaxDec, eNetPay[i]);
                        }
                        else
                        {
                            break;
                        }
                    }
                }
                else
                {
                    MessageBox.Show("Please Process a month first.", "No month processed",
    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                }

            }



            private void checkForMonthProcess()
            {
                for (int step = 0; step < isMonthProcess.Length; step++)
                {
                    allMonthsProcessedBool = true;
                    if (isMonthProcess[step] == "no")
                    {
                        allMonthsProcessedBool = false;
                        MessageBox.Show("Not all months have been processed. " +
    months[step].ToString() + " is not processed.", "Invalid Request", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
                        break;

                    }
                }

            }

        }
    }
```

Payroll Class - Paola Socorro

```
/*  Program:     Payroll Project 4
    Author:      Paola Socorro (for this PayrollClass)
    Class:      CISP 41
    Date:       May 18 2013
    Description: This Class is will calculate payroll for salaried employees and hourly
employees.

    I certify that the code below is my own work.

   Exception(s): N/A

*/



using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Project4
{
    class PayrollClass
    {
        //MANDATORY
        public const decimal SOCIAL_SECURITY_TAX = .10M;
        public const decimal FEDERAL_INCOME_TAX = .15M;
        public const decimal STATE_INCOME_TAX = .05M;




        //GROSS AND NET PAY
        protected decimal grossPayDecimal;
        protected decimal netPayDecimal;
        protected decimal deductedTaxes;

        public PayrollClass()
        {

        }


        public decimal grossPay(decimal rate, decimal hours)
        {
            grossPayDecimal = rate * hours;
            return grossPayDecimal;
        }

        public decimal grossPaySalary(decimal salary)
        {
            grossPayDecimal = salary;
            return grossPayDecimal;
        }

        //TO BE CALCULATED BEFORE TAXES.
        public decimal deductionPay(decimal benefitsMedDen, decimal lifeInsurance, decimal
save401K, decimal fsa)
        {
            netPayDecimal = grossPayDecimal - (benefitsMedDen+ lifeInsurance + save401K +
fsa);
            return netPayDecimal;
        }

        //TO BE CALCULATED AFTER deductionPay()
        public decimal mandatoryTaxDeductions()
        {
            deductedTaxes= netPayDecimal * (SOCIAL_SECURITY_TAX + FEDERAL_INCOME_TAX +
STATE_INCOME_TAX);
            return deductedTaxes;
        }

        public decimal netPayCalc()
        {
            netPayDecimal = netPayDecimal - deductedTaxes;
            return netPayDecimal;
        }

        //Stop contribution of 401k to  15,000
        public decimal stop401k(decimal contribution401k)
        {
            return contribution401k;
        }

    }
}
```

Report Class by Qianqun Xu

```
/*  Program:  Report Class
    Author:   Qianqun Xu
    Class:   CISP 41
    Date:
    Description: Handles reports program

    I certify that the code below is my own work.

   Exception(s): N/A

*/


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Project4
{
    class Report
    {

        protected string nameString;
        protected decimal wagesDecimal, socialSecurityTaxDecimal, fedIncomeTaxDecimal,
stateIncomeTaxDecimal;

        public string Name
        {
            get { return nameString; }
            set { nameString = value; }
        }
        public decimal Wages
        {
            get { return wagesDecimal; }
            set { wagesDecimal = value; }
        }
        public decimal SocialSecurityTax
        {
            get { return socialSecurityTaxDecimal; }
            set { socialSecurityTaxDecimal = value; }
        }
        public decimal FedIncomeTax
        {
            get { return fedIncomeTaxDecimal; }
            set { fedIncomeTaxDecimal = value; }
        }
        public decimal StateIncomeTax
        {
            get { return stateIncomeTaxDecimal; }
            set { stateIncomeTaxDecimal = value; }
        }


        public Report(string nameString, decimal wagesDecimal, decimal
socialSecurityTaxDecimal,decimal fedIncomeTaxDecimal, decimal stateIncomeTaxDecimal)
        {
            Name = nameString; Wages = wagesDecimal;
            SocialSecurityTax = socialSecurityTaxDecimal;
            FedIncomeTax = fedIncomeTaxDecimal;
            StateIncomeTax = stateIncomeTaxDecimal;
            printReport();
        }

        public virtual void printReport()
        {

        }



    }
}

/*  Program:  Paystub Report Class
    Author:   Qianqun Xu
    Class:   CISP 41
    Date:
    Description: Handles reports pay stubs in derived classess.

    I certify that the code below is my own work.

   Exception(s): N/A

*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using System.Drawing.Printing;
using System.Windows.Forms;
using System.ComponentModel;


namespace Project4
{
    class PaystubReport
    {
        private string nameString;
        private int monthInterger;
        private decimal grossPayDecimal, socialSecurityTaxDecimal, fedIncomeTaxDecimal,
stateIncomeTaxDecimal, netPayDecimal;

        public string Name
        {
            get { return nameString; }
            set { nameString = value; }
        }
        public int Month
        {
            get { return monthInterger; }
            set { monthInterger = value; }
        }
        public decimal GrossPay
        {
            get { return grossPayDecimal; }
            set { grossPayDecimal = value; }
        }
        public decimal SocialSecurityTax
        {
            get { return socialSecurityTaxDecimal; }
            set { socialSecurityTaxDecimal = value; }
        }
        public decimal FedIncomeTax
        {
            get { return fedIncomeTaxDecimal; }
            set { fedIncomeTaxDecimal = value; }
        }
        public decimal StateIncomeTax
        {
            get { return stateIncomeTaxDecimal; }
            set { stateIncomeTaxDecimal = value; }
        }
        public decimal NetPay
        {
            get { return netPayDecimal; }
            set { netPayDecimal = value; }
        }


        public PaystubReport(string nameString, decimal grossPayDecimal, decimal
socialSecurityTaxDecimal,decimal fedIncomeTaxDecimal, decimal stateIncomeTaxDecimal, decimal
netPayDecimal)
        {

            Name = nameString;
            GrossPay = grossPayDecimal;
            SocialSecurityTax = socialSecurityTaxDecimal;
            FedIncomeTax = fedIncomeTaxDecimal;
            StateIncomeTax = stateIncomeTaxDecimal;
            NetPay = netPayDecimal;
            printReport();
        }

        public void printReport()
        {

            PrintPreviewDialog PrintPreviewDialog1 = new PrintPreviewDialog();
            PaperSize paperSize = new PaperSize("DataOrder", 470, 660);
            PrintDocument Report = new PrintDocument();
            Report.DefaultPageSettings.PaperSize = paperSize;
            PrintPreviewDialog1.Document = Report;
            Report.PrintPage += new PrintPageEventHandler(Report_PrintPage);
            PrintPreviewDialog1.FormBorderStyle = FormBorderStyle.Fixed3D;
            PrintPreviewDialog1.ShowDialog();
        }

        public   void   Report_PrintPage(object
sender,System.Drawing.Printing.PrintPageEventArgs  e)
        {

            float  hozPosFloat  =  15.0f;
            float  verPosFoat  =  15.0f;
            float  leftbianJu  =  5;
            float  topbianJu  = 5;


            Pen line  = new  Pen(Color.Black,  1.0f);


            //Draw  frame  and  blue  patches
            e.Graphics.FillRectangle(Brushes.DarkBlue,  leftbianJu,  topbianJu,  461,  50);
            e.Graphics.FillRectangle(Brushes.DarkBlue,  leftbianJu,  topbianJu  + 100,
461,20);


            e.Graphics.DrawRectangle(line,  leftbianJu,  topbianJu  + 122,  460,  60);
            e.Graphics.DrawRectangle(line,  leftbianJu,  topbianJu  + 122,  460,  100);
            e.Graphics.FillRectangle(Brushes.DarkBlue,  leftbianJu,  topbianJu  + 225,
461,20);


            e.Graphics.DrawRectangle(line,  leftbianJu,  topbianJu  + 247,  460,  60);
            e.Graphics.FillRectangle(Brushes.DarkBlue,  leftbianJu,  topbianJu  + 309,
461,20);


            e.Graphics.DrawRectangle(line,  leftbianJu,  topbianJu  + 331,  460,  100);
            e.Graphics.FillRectangle(Brushes.DarkBlue,  leftbianJu,  topbianJu  + 434,
461,20);


            e.Graphics.DrawRectangle(line,  leftbianJu,  topbianJu  + 456,  460,  100);
            e.Graphics.FillRectangle(Brushes.DarkBlue,  leftbianJu,  topbianJu  + 560,
461,20);


            e.Graphics.DrawRectangle(line,  leftbianJu,  topbianJu  + 581,  460,  50);

            //Paystubs,  2013,Fill  the  form  with  instructions
            e.Graphics.DrawString("Pay    Stubs",    new    Font("Kozuka    Mincho    Pro
B",    9, FontStyle.Bold),
            Brushes.White,  hozPosFloat  + 200,  verPosFoat  + 20);
            e.Graphics.DrawString("Summary",  new Font("Kozuka  Mincho Pro B", 6,
FontStyle.Bold),
            Brushes.White,  hozPosFloat,  112);

            e.Graphics.DrawString("Name\t\t\t\t\t\tAddress",  new  Font("Kozuka  Mincho  Pro
B",  7,  FontStyle.Bold),
            Brushes.Black,  hozPosFloat,  131);
            e.Graphics.DrawString("Employee  ID\t\tSSN\t\tDate from\tDate to",   new
            Font("Kozuka  Mincho  Pro  B",  7,  FontStyle.Bold), Brushes.Black,
hozPosFloat,  192);
            e.Graphics.DrawString("Gross    Pay",    new    Font("Kozuka    Mincho    Pro
B",    6, FontStyle.Bold),
            Brushes.White,  hozPosFloat,  237);
            e.Graphics.DrawString("Amount",  new Font("Kozuka  Mincho Pro B", 7,
FontStyle.Bold),
            Brushes.Black,  hozPosFloat,  256);
            e.Graphics.DrawString("Taxes",   new Font("Kozuka  Mincho Pro B", 6,
FontStyle.Bold),
            Brushes.White,  hozPosFloat,  321);
            e.Graphics.DrawString("Type\t\t\t\t\tAmount", new Font("Kozuka Mincho Pro B",
            7,  FontStyle.Bold),
            Brushes.Black,  hozPosFloat,  345);
            e.Graphics.DrawString("Social Security  Tax",  new Font("Arial", 6,
FontStyle.Regular),
            Brushes.Black,  hozPosFloat,  368);
            e.Graphics.DrawString("Fed Income Tax",  new Font("Arial", 6, FontStyle.Regular),
            Brushes.Black,  hozPosFloat,  388);
            e.Graphics.DrawString("State  Income Tax",  new Font("Arial", 6,
FontStyle.Regular),
            Brushes.Black,  hozPosFloat,  408);
            e.Graphics.DrawString("Deduction",    new    Font("Kozuka    Mincho    Pro
B",    6, FontStyle.Regular),
            Brushes.White,  hozPosFloat,  446);
            e.Graphics.DrawString("Type\t\t\t\t\tAmount", new Font("Kozuka Mincho Pro B",
            7,  FontStyle.Bold),
            Brushes.Black,  hozPosFloat,  468);
            e.Graphics.DrawString("Net Pay",  new Font("Kozuka  Mincho Pro B", 6,
FontStyle.Bold),
            Brushes.White,  hozPosFloat,  572);
            e.Graphics.DrawString("Amount",  new Font("Kozuka  Mincho Pro B", 7,
FontStyle.Bold),
            Brushes.Black,  hozPosFloat,  592);


            //Fill  the  blank  with  data
            e.Graphics.DrawString(nameString,  new  Font("Arial",  8,  FontStyle.Regular),
Brushes.Black,  hozPosFloat,  150);

            e.Graphics.DrawString(grossPayDecimal.ToString(),new Font("Arial", 7,
FontStyle.Regular), Brushes.Black, hozPosFloat, 280);

            e.Graphics.DrawString(socialSecurityTaxDecimal.ToString()  ,new Font("Arial", 6,
FontStyle.Regular), Brushes.Black, hozPosFloat + 200,368);


            e.Graphics.DrawString(fedIncomeTaxDecimal.ToString(),new Font("Arial", 6,
FontStyle.Regular), Brushes.Black, hozPosFloat + 200,388);


            e.Graphics.DrawString(stateIncomeTaxDecimal.ToString(),new Font("Arial", 6,
FontStyle.Regular), Brushes.Black, hozPosFloat + 200,408);


            e.Graphics.DrawString(netPayDecimal.ToString(),new Font("Arial", 7,
FontStyle.Regular), Brushes.Black, hozPosFloat, 605);
            }

    }
}

/*  Program:  Derived W-2:Report Class
    Author:   Qianqun Xu
    Class:   CISP 41
    Date:
    Description: Handles reports for W-2  in derived classess.

    I certify that the code below is my own work.

   Exception(s): N/A

*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using System.Drawing.Printing;
using System.Windows.Forms;
using System.ComponentModel;


namespace Project4
{
    class W_2Report:Report
    {
        public W_2Report(string nameString, decimal wagesDecimal, decimal
socialSecurityTaxDecimal,decimal fedIncomeTaxDecimal, decimal stateIncomeTaxDecimal)
            :
            base(nameString, wagesDecimal, socialSecurityTaxDecimal, fedIncomeTaxDecimal,
stateIncomeTaxDecimal)
        {
        }


        public override void printReport()
        {

            PrintPreviewDialog PrintPreviewDialog1 = new PrintPreviewDialog();
            PaperSize paperSize = new PaperSize("DataOrder", 810, 470);
            PrintDocument Report = new PrintDocument();
            Report.DefaultPageSettings.PaperSize = paperSize;
            PrintPreviewDialog1.Document = Report;
            Report.PrintPage += new PrintPageEventHandler(Report_PrintPage);
            PrintPreviewDialog1.FormBorderStyle = FormBorderStyle.Fixed3D;
            PrintPreviewDialog1.ShowDialog();
        }

        public   void   Report_PrintPage(object
sender,System.Drawing.Printing.PrintPageEventArgs  e)
        {

            //draw  a  form

            float  hozPosFloat  =  15.0f; float  verPosFoat  =  15.0f; float  leftbianJu  =
15; float  topbianJu  = 15;
            float  tableWidth  = 780;
            float  tableHeight  = 393;


            Pen line  = new  Pen(Color.Black,  1.0f);
            Pen wideLine  = new  Pen(Color.Black,  2.0f);


            //Draw  a  Rectangle,  the  frame  of  the  form


            e.Graphics.DrawRectangle(wideLine,   leftbianJu,   topbianJu, tableWidth,
tableHeight);





            //Draw  horizontal  lines  inside  the  table
            e.Graphics.DrawLine(line, leftbianJu, topbianJu + 30, leftbianJu + tableWidth,
topbianJu  + 30);
            e.Graphics.DrawLine(line, leftbianJu, topbianJu + 60, leftbianJu + tableWidth,
topbianJu  + 60);
            e.Graphics.DrawLine(line, leftbianJu, topbianJu + 150, leftbianJu + tableWidth,
topbianJu  + 150);
            e.Graphics.DrawLine(line, leftbianJu, topbianJu + 180, leftbianJu + tableWidth,
topbianJu  + 180);
            e.Graphics.DrawLine(line, leftbianJu, topbianJu + 320, leftbianJu + tableWidth,
topbianJu  + 320);
            e.Graphics.DrawLine(line,   leftbianJu   +  400,   topbianJu   +  90,
leftbianJu   +
            tableWidth,  topbianJu  + 90);
            e.Graphics.DrawLine(line,   leftbianJu   +  400,   topbianJu   +  120,
leftbianJu   +
            tableWidth,  topbianJu  + 120);
            e.Graphics.DrawLine(line,   leftbianJu   +  400,   topbianJu   +  210,
leftbianJu   +
            tableWidth,  topbianJu  + 210);
            e.Graphics.DrawLine(line,   leftbianJu   +  400,   topbianJu   +  240,
leftbianJu   +
            tableWidth,  topbianJu  + 240);


            //Draw  vertical  lines  inside  the  table
            e.Graphics.DrawLine(line,   leftbianJu   +  140,   topbianJu,   leftbianJu   +
140, topbianJu  + 30);
            e.Graphics.DrawLine(line,   leftbianJu   +  280,   topbianJu,   leftbianJu   +
280, topbianJu  + 30);
            e.Graphics.DrawLine(line,  leftbianJu  + 400,  topbianJu  + 30,  leftbianJu  +
400,

            topbianJu  + 320);
            e.Graphics.DrawLine(line,  leftbianJu  + 590,  topbianJu  + 30,  leftbianJu  +
590, topbianJu  + 320);
            e.Graphics.DrawLine(line, leftbianJu + 210, topbianJu + 320, leftbianJu + 210,
topbianJu  + tableHeight);
            e.Graphics.DrawLine(line, leftbianJu + 320, topbianJu + 320, leftbianJu + 320,
topbianJu  + tableHeight);
            e.Graphics.DrawLine(line, leftbianJu + 430, topbianJu + 320, leftbianJu + 430,
topbianJu  + tableHeight);


            //W-2,  2013
            e.Graphics.DrawString("Form", new Font("Microsoft YaHei", 6, FontStyle.Bold),
Brushes.Black,  hozPosFloat,  verPosFoat  + 414);
            e.Graphics.DrawString("   W-2",  new Font("Microsoft   YaHei",   15,
FontStyle.Bold),
            Brushes.Black,  hozPosFloat,  verPosFoat  + 400);
            e.Graphics.DrawString("2013", new Font("Kozuka  Mincho Pro B", 15,
FontStyle.Bold),
            Brushes.Black,  hozPosFloat  + 350,  verPosFoat  + 400);
            e.Graphics.DrawString("Department   of   the   Treasury",   new   Font("Arial",
9, FontStyle.Regular),
            Brushes.Black,  hozPosFloat  + 600,  verPosFoat  + 400);


            //Fill  the  form  with  instructions
            e.Graphics.DrawString("   Control   number",  new Font("Arial", 7,
FontStyle.Regular),
            Brushes.Black,  hozPosFloat,  verPosFoat  + 1);
            e.Graphics.DrawString("  Employer identification  Number",  new Font("Arial",
7, FontStyle.Regular),
            Brushes.Black,  hozPosFloat,  verPosFoat  + 31);
            e.Graphics.DrawString("   Employer's name,  address   and   ZIP code", new
            Font("Arial",  7,  FontStyle.Regular),
            Brushes.Black,  hozPosFloat,  verPosFoat  + 61);
            e.Graphics.DrawString("  Employee's  social  security number", new
Font("Arial",
            7,  FontStyle.Regular),
            Brushes.Black,  hozPosFloat,  verPosFoat  + 151);
            e.Graphics.DrawString("   Employee's name", new Font("Arial", 7,
FontStyle.Regular),
            Brushes.Black,  hozPosFloat,  verPosFoat  + 181);
            e.Graphics.DrawString("  Wages, tips,  other compensation\t\tFederal income  tax
withheld",  new  Font("Arial",  7,  FontStyle.Regular),
            Brushes.Black,  hozPosFloat  + 400,  verPosFoat  + 31);
            e.Graphics.DrawString("   Social   security   wages\t\t\tSocial   secutiry
tax withheld",  new  Font("Arial",  7,  FontStyle.Regular),
            Brushes.Black,  hozPosFloat  + 400,  verPosFoat  + 61);
            e.Graphics.DrawString("  Medicare  wages  and tips\t\t\tMedicare  tax
withheld", new  Font("Arial",  7,  FontStyle.Regular),
            Brushes.Black,  hozPosFloat  + 400,  verPosFoat  + 91);
            e.Graphics.DrawString("   Social   security   tips\t\t\tAllocated   tips",
new
            Font("Arial",  7,  FontStyle.Regular),
            Brushes.Black,  hozPosFloat  + 400,  verPosFoat  + 121);
            e.Graphics.DrawString("  State   Employer's  state  ID  number\t State wages\t\t
   State  income  tax",  new  Font("Arial",  7,  FontStyle.Regular),
            Brushes.Black,  hozPosFloat,  verPosFoat  + 321);


            //Fill  the  blank  with  data
            e.Graphics.DrawString(Name,  new  Font("Arial",  10,  FontStyle.Bold),
Brushes.Black,  hozPosFloat  + 10,  verPosFoat  + 201);
e.Graphics.DrawString(Wages.ToString()  + "\n\n"  + Wages.ToString()  +  "\n\n"  +
            Wages.ToString(),
            new  Font("Arial",   9,  FontStyle.Bold),Brushes.Black,  hozPosFloat   +  520,
verPosFoat  + 44);
            e.Graphics.DrawString(FedIncomeTax.ToString()  +   "\n\n" +
SocialSecurityTax.ToString(),
            new  Font("Arial",  9,  FontStyle.Bold),  Brushes.Black,  hozPosFloat  +  720,
verPosFoat  + 44);
            e.Graphics.DrawString(Wages.ToString()   +   "\t\t  "   +
StateIncomeTax.ToString(),
            new  Font("Arial",  9,  FontStyle.Bold),  Brushes.Black,  hozPosFloat  +  250,
verPosFoat  + 335);

            }

    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace Project4
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new splashForm());//psocorro
            Application.Run(new Payroll_Form());
        }
    }
}
```