UML Design:

A screenshot of a cell phone

Description automatically generated

Code Execution:

A close up of text on a black surface

Description automatically generated

public class Employee

{

private string \_first = "First";

private string \_last = "Last";

private int \_employeeNum = 00000;

private double \_sales = 0.00;

public string FirstName

{

get => \_first;

set

{

\_first = char.ToUpper(value[0]) + value.Substring(1);

}

}

public string LastName

{

get => \_last;

set

{

\_last = char.ToUpper(value[0]) + value.Substring(1);

}

}

public int EmployeeNumber

{

get => \_employeeNum;

set

{

if (value > 0 && value < 99999)

{

\_employeeNum = Convert.ToInt32(value.ToString("D5"));

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"[-] Error. Employee {\_first}'s ID must be 5 digits.");

Console.ResetColor();

}

}

}

public double TotalSales

{

get => \_sales;

set

{

if (value > 0)

{

\_sales = value;

}

}

}

public Employee() { }

public Employee(string first, string last, int number, double sales)

{

this.EmployeeNumber = number;

this.FirstName = first;

this.LastName = last;

this.TotalSales = sales;

}

public void Deductions()

{

double deducCut = Math.Round(\_sales \* 0.09, 2);

double deducFed = Math.Round(deducCut \* 0.18, 2);

double deducRet = Math.Round(deducCut \* 0.1, 2);

double deducSoc = Math.Round(deducCut \* 0.06, 2);

double net = deducCut - deducFed - deducRet - deducSoc;

employeeInfo(net, deducCut, deducFed, deducRet, deducSoc);

}

public void displayDeductions(double deducFed, double deducRet, double deducSoc)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[-] Deductions: ");

Console.WriteLine($"- Federal Tax (18%):\t{deducFed:C2}");

Console.WriteLine($"- Retirement (10%):\t{deducRet:C2}");

Console.WriteLine($"- Social Sec (6%):\t{deducSoc:C2}");

Console.ResetColor();

}

public void employeeInfo(double net, double deducCut, double deducFed, double deducRet, double deducSoc)

{

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("[+] Employee Information:");

Console.WriteLine($"--> First Name:\t\t{\_first}");

Console.WriteLine($"--> Last Name:\t\t{\_last}");

Console.WriteLine($"--> ID: \t\t#{\_employeeNum:D5}");

Console.WriteLine($"--> Total Sales: \t{\_sales:C2}");

Console.WriteLine($"--> Employee Cut (9%):\t{deducCut:C2}");

displayDeductions(deducFed, deducRet, deducSoc);

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine($"--> Net Salary: \t{net:C2}");

Console.ResetColor();

Console.WriteLine("------------------------------------");

}

}

class Driver

{

static void Main(string[] args)

{

Console.WriteLine(" ------------------------------------------");

Console.WriteLine(" | Net Pay Calculator |");

Console.WriteLine(" ------------------------------------------");

bool again = false;

do

{

Console.Write("Employee Full Name? ");

string name = Console.ReadLine();

string[] nameSep = name.Split(' ');

Console.Write("Employee ID? #");

int employeeID;

while (!Int32.TryParse(Console.ReadLine(), out employeeID) || employeeID < 0 || employeeID > 99999)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[-] Error! ID must be 5 digit number.");

Console.ResetColor();

Console.Write("Try again > ");

}

double sales;

Console.Write("Total Sales? $");

while (!Double.TryParse(Console.ReadLine(), out sales) || sales < 0)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Error! Sales must be positive number.");

Console.ResetColor();

Console.Write("Try again > ");

}

Employee emp = new Employee(nameSep[0], nameSep[nameSep.Length - 1], employeeID, sales);

emp.calculateDeductions();

Console.Write("Again? (y/n) ");

char againInput = Convert.ToChar(Console.ReadLine());

if (againInput == 'y')

{

again = true;

Console.WriteLine("------------------------------------");

}

else

{

again = false;

Console.WriteLine("Goodbye.");

Environment.Exit(0);

}

} while (again);

}

}