**Benefits**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace C# 6

{

classBenefits

{

privateconststring DEFAULT\_HEALTH\_INSURANCE = "Blue Cross";

privateconstdouble MIN\_LIFE\_INSURANCE = 0;

privateconstdouble MAX\_LIFE\_INSURANCE = 1000000;

privateconstint MIN\_VACATION = 0;

privateconstint MAX\_VACATION = 45;

publicstring healthInsuranceCompany = DEFAULT\_HEALTH\_INSURANCE;

publicdouble lifeInsuranceAmount = MIN\_LIFE\_INSURANCE;

publicint vacationDays = MIN\_VACATION;

public Benefits()

{

healthInsuranceCompany = DEFAULT\_HEALTH\_INSURANCE;

lifeInsuranceAmount = MIN\_LIFE\_INSURANCE;

vacationDays = MIN\_VACATION;

}

public Benefits(string health, double life, int vacation)

{

healthInsuranceCompany = health;

lifeInsuranceAmount = life;

vacationDays = vacation;

}

publicoverridestring ToString()

{

string output;

output = "\n\t============ Benefit Information ============";

output += "\nHealth Insurance Company:\t" + healthInsuranceCompany;

output += "\n Life Insurance Amount:\t" + lifeInsuranceAmount.ToString("C2");

output += "\n Vacation Days:\t" + vacationDays;

return output;

}

publicstring Health\_Insurance\_Info

{

get { return healthInsuranceCompany; }

set

{

if (value.Trim() == String.Empty || value.Trim() == "")

healthInsuranceCompany = DEFAULT\_HEALTH\_INSURANCE;

else

healthInsuranceCompany = value;

}

}

publicdouble Life\_Insurance\_Info

{

get { return lifeInsuranceAmount; }

set

{

if (value< MIN\_LIFE\_INSURANCE)

lifeInsuranceAmount = MIN\_LIFE\_INSURANCE;

elseif (value> MAX\_LIFE\_INSURANCE)

lifeInsuranceAmount = MAX\_LIFE\_INSURANCE;

else

lifeInsuranceAmount = value;

}

}

publicint Vacation\_Days\_Info

{

get { return vacationDays; }

set

{

if (value< MIN\_VACATION)

vacationDays = MIN\_VACATION;

elseif (value> MAX\_VACATION)

vacationDays = MAX\_VACATION;

else

vacationDays = value;

}

}

~Benefits() { }

}

}

**Employee**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace C# 6B

{

abstractclassEmployee

{

privateconstint MIN\_DEPENDENTS = 0;

privateconstint MAX\_DEPENDENTS = 10;

privateconstdouble MIN\_SALARY = 20000;

privateconstdouble MAX\_SALARY = 100000;

privateconststring DEFAULT\_NAME = "Not Given";

privateconstchar DEFAULT\_GENDER = 'U';

publicconststring DEFAULT\_TYPE = "Generic Employee";

protectedstring firstName = DEFAULT\_NAME, lastName = DEFAULT\_NAME;

protectedchar gender = DEFAULT\_GENDER;

protectedint dependents = MIN\_DEPENDENTS;

protecteddouble annualSalary = MIN\_SALARY;

protectedstaticint numEmployees = 0;

protected Benefits benefits;

protectedstring employeeType;

publicabstractdouble CalculateNetPay();

public Employee()

{

employeeType = DEFAULT\_TYPE;

FirstName = DEFAULT\_NAME;

LastName = DEFAULT\_NAME;

Gender = DEFAULT\_GENDER;

Dependents = MIN\_DEPENDENTS;

AnnualSalary = MIN\_SALARY;

EmployeeBenefits = new Benefits();

numEmployees++;

}

public Employee(string type)

: this()

{

employeeType = type;

}

public Employee(string type, string fn, string ln, char gen, int dep, double ans, Benefits ben)

{

employeeType = type;

FirstName = fn;

LastName = ln;

Gender = gen;

Dependents = dep;

AnnualSalary = ans;

EmployeeBenefits = ben;

numEmployees++;

}

publicvirtualdouble CalculateWeeklyPay()

{

return annualSalary / 52;

}

publicdouble CalculateWeeklyPay(double modifiedSalary)

{

return modifiedSalary / 52;

}

publicoverridestring ToString()

{

string output;

output = "\n============ Employee Information ============";

output += "\n\t Type:\t" + EmployeeType;

output += "\n\t Name:\t" + firstName + " " + lastName;

output += "\n\t Gender:\t" + FullGender;

output += "\n\t Dependents:\t" + Dependents;

output += "\n\t Annual Salary:\t" + AnnualSalary.ToString("C2");

output += "\n\t Weekly Pay:\t" + CalculateWeeklyPay().ToString("C2");

output += "\n\t NetPay:\t" + CalculateNetPay().ToString("C2");

output += EmployeeBenefits.ToString();

return output;

}

publicstring FirstName

{

get { return firstName; }

set

{

if (value.Trim() == String.Empty || value.Trim() == "")

firstName = DEFAULT\_NAME;

else

firstName = value;

}

}

publicstring LastName

{

get { return lastName; }

set

{

if (value.Trim() == String.Empty || value.Trim() == "")

lastName = DEFAULT\_NAME;

else

lastName = value;

}

}

publicchar Gender

{

get { return gender; }

set

{

char input;

input = Char.ToUpper(value);

if (input == 'F' || input == 'M')

gender = input;

else

gender = 'U';

}

}

publicstring FullGender

{

//create a read only property to

//provide the full text of the gender

get

{

string value;

switch (Gender)

{

case'F':

value = "Female";

break;

case'M':

value = "Male";

break;

default:

value = "Unknown";

break;

}

return value;

}

}

publicint Dependents

{

get { return dependents; }

set

{

if (value< MIN\_DEPENDENTS)

dependents = MIN\_DEPENDENTS;

elseif (value> MAX\_DEPENDENTS)

dependents = MAX\_DEPENDENTS;

else

dependents = value;

}

}

publicdouble AnnualSalary

{

get { return annualSalary; }

set

{

if (value< MIN\_SALARY)

annualSalary = MIN\_SALARY;

elseif (value> MAX\_SALARY)

annualSalary = MAX\_SALARY;

else

annualSalary = value;

}

}

publicstaticint NumberEmployees

{

get

{

return numEmployees;

}

}

public Benefits EmployeeBenefits

{

get { return benefits; }

set

{

if (benefits != null)

benefits = new Benefits();

else

benefits = value;

}

}

publicString EmployeeType

{

get { return employeeType; }

set

{

if (String.IsNullOrEmpty(value))

employeeType = DEFAULT\_TYPE;

else

employeeType = value;

}

}

~Employee() { }

}

}

**Hourly**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace C# 6C

{

classHourly : Employee

{

privateconstdouble MIN\_WAGE = 10;

privateconstdouble MAX\_WAGE = 75;

privateconstdouble MIN\_HOURS = 0;

privateconstdouble MAX\_HOURS = 50;

privateconstdouble TAX\_RATE = 0.82;

privateconststring DEFAULT\_CATEGORY = "temporary";

privatedouble wages;

privatedouble hours;

privatestring category;

public Hourly()

: base()

{

Wages = MIN\_WAGE;

Hours = MIN\_HOURS;

Category = "Undefined";

}

public Hourly(string employeeType)

: base(employeeType)

{

Wages = MIN\_WAGE;

Hours = MIN\_HOURS;

Category = DEFAULT\_CATEGORY;

}

public Hourly(string emp, string first, string last, char gen, int dep, double wage, double hour, Benefits ben, string cat)

: base(emp, first, last, gen, dep, 0, ben)

{

Wages = wage;

Hours = hour;

Category = cat;

}

publicoverridedouble CalculateWeeklyPay()

{

return Hours \* Wages;

}

publicoverridestring ToString()

{

string output = base.ToString();

output += "\n\t Hours:\t" + Wages.ToString("C2");

output += "\n\t Wages:\t" + Hours;

output += "\n\t Category: \t" + Category;

return output;

}

publicoverridedouble CalculateNetPay()

{

return CalculateWeeklyPay() \* TAX\_RATE;

}

publicdouble Wages

{

get { return wages; }

set

{

if (value< MIN\_WAGE)

wages = MIN\_WAGE;

elseif (value> MAX\_WAGE)

wages = MAX\_WAGE;

else

wages = value;

base.annualSalary = CalculateWeeklyPay() \* 48;

}

}

publicdouble Hours

{

get { return hours; }

set

{

if (value< MIN\_HOURS)

hours = MIN\_HOURS;

elseif (value> MAX\_HOURS)

hours = MAX\_HOURS;

else

hours = value;

base.annualSalary = CalculateWeeklyPay() \* 48;

}

}

publicstring Category

{

get { return category; }

set

{

if (value.Trim() == String.Empty || value.Trim() == "")

category = "Undefined";

else

category = value;

}

}

}

}

**Salaried**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace C# 6D

{

classSalaried : Employee

{

privateconstint MIN\_MANAGEMENT\_LEVEL = 0;

privateconstint MAX\_MANAGEMENT\_LEVEL = 3;

privateconstdouble BONUS\_PERCENT = 0.10;

privateconstdouble TAX\_RATE = 0.73;

privateint managementLevel;

public Salaried()

: base()

{

ManagementLevel = MIN\_MANAGEMENT\_LEVEL;

}

public Salaried(string employeeType)

: base(employeeType)

{

ManagementLevel = MIN\_MANAGEMENT\_LEVEL;

}

public Salaried(string emp, string first, string last, char gen, int dep, double sal, Benefits ben, int manLevel)

: base(emp, first, last, gen, dep, sal, ben)

{

ManagementLevel = manLevel;

}

publicoverridedouble CalculateWeeklyPay()

{

return (annualSalary + ((ManagementLevel \* BONUS\_PERCENT) \* annualSalary)) / 52;

}

publicoverridestring ToString()

{

string output = base.ToString();

output += "\n\t Level:" + ManagementLevel;

return output;

}

publicoverridedouble CalculateNetPay()

{

return CalculateWeeklyPay() \* TAX\_RATE;

}

publicint ManagementLevel

{

get { return managementLevel; }

set

{

if (value< MIN\_MANAGEMENT\_LEVEL)

managementLevel = MIN\_MANAGEMENT\_LEVEL;

elseif (value> MAX\_MANAGEMENT\_LEVEL)

managementLevel = MAX\_MANAGEMENT\_LEVEL;

else

managementLevel = value;

}

}

}

}

**Employee Input**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace C# 6E

{

classEmployeeInput

{

publicstaticvoid CollectEmployeeInformation(Employee theEmployee)

{

theEmployee.FirstName = InputUtilities.getStringInputValue("First Name");

theEmployee.LastName = InputUtilities.getStringInputValue("Last Name");

theEmployee.Gender = InputUtilities.getCharInputValue("Gender");

theEmployee.Dependents = InputUtilities.getIntegerInputValue("Number of Dependents");

theEmployee.EmployeeBenefits.healthInsuranceCompany = InputUtilities.getStringInputValue("Health Insurance Company");

theEmployee.EmployeeBenefits.lifeInsuranceAmount = InputUtilities.getDoubleInputValue("Life Insurance Amount");

theEmployee.EmployeeBenefits.vacationDays = InputUtilities.getIntegerInputValue("Vacation Days");

}

publicstaticvoid CollectorHourlyInformation(Hourly theEmployee)

{

theEmployee.Wages = InputUtilities.getDoubleInputValue("Hourly Wage");

theEmployee.Hours = InputUtilities.getDoubleInputValue("Hours Worked");

theEmployee.Category = InputUtilities.getStringInputValue("Category");

}

publicstaticvoid CollectorSalariedInformation(Salaried theEmployee)

{

theEmployee.ManagementLevel = InputUtilities.getIntegerInputValue("Management Level");

theEmployee.AnnualSalary = InputUtilities.getDoubleInputValue("Annual Salary");

}

}

}

**Employee Output**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace C# 6F

{

classEmployeeOutput

{

publicstaticvoid DisplayEmployeeInformation(Employee theEmployee)

{

ApplicationUtilities.DisplayDivider(theEmployee.FirstName + " " + theEmployee.LastName);

Console.Write(theEmployee.ToString());

}

publicstaticvoid DisplayNumberObjects()

{

ApplicationUtilities.DisplayDivider("Number of Employees");

ApplicationUtilities.DisplayDivider("Number of Employee Objects: " + Employee.NumberEmployees + " ");

}

}

}

**Input Utilities**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace C# 6G

{

classInputUtilities

{

publicstaticstring GetInput(string inputType)

{

string strInput = String.Empty;

Console.Write("Enter the " + inputType + ": ");

strInput = Console.ReadLine();

return strInput;

}

publicstaticstring getStringInputValue(string inputType)

{

string value = String.Empty;

bool valid = false;

string inputString = String.Empty;

do

{

inputString = GetInput(inputType);

if (!String.IsNullOrEmpty(inputString))

{

value = inputString;

valid = true;

}

else

{

value = "Invalid input";

valid = false;

}

if (!valid)

Console.WriteLine("Invalid " + inputType + " try again!");

} while (!valid);

return value;

}

publicstaticint getIntegerInputValue(string inputType)

{

bool valid = false;

int value = 0;

string inputString = String.Empty;

do

{

inputString = GetInput(inputType);

if (!(String.IsNullOrEmpty(inputString)))

{

valid = Int32.TryParse(inputString, out value);

}

if (!valid)

Console.WriteLine("Invalid " + inputType + " try again!");

} while (!valid);

return value;

}

publicstaticdouble getDoubleInputValue(string inputType)

{

bool valid = false;

double value = 0;

string inputString = String.Empty;

do

{

inputString = GetInput(inputType);

if (!(String.IsNullOrEmpty(inputString)))

{

valid = Double.TryParse(inputString, out value);

}

if (!valid)

Console.WriteLine("Invalid " + inputType + " try again!");

} while (!valid);

return value;

}

publicstaticchar getCharInputValue(string inputType)

{

bool valid = false;

char value = 'u';

string inputString = String.Empty;

do

{

inputString = GetInput(inputType);

if (!(String.IsNullOrEmpty(inputString)))

{

valid = Char.TryParse(inputString, out value);

}

if (!valid)

Console.WriteLine("Invalid " + inputType + " try again!");

} while (!valid);

return value;

}

}

}

**Program**

using System;

namespace C# 6H

{

classProgram

{

staticvoid Main(string[] args)

{

ApplicationUtilities.DisplayApplicationInformation();

ApplicationUtilities.DisplayDivider("Start Program");

Employee[] el = new Employee[2];

el[0] = new Hourly("Hourly");

el[1] = new Salaried("Salaried");

for (int i = 0; i < el.Length; i++)

{

if (el[i] is Hourly)

{

ApplicationUtilities.DisplayDivider("Collect Hourly Information");

EmployeeInput.CollectEmployeeInformation(el[i]);

EmployeeInput.CollectorHourlyInformation((Hourly)el[i]);

EmployeeOutput.DisplayEmployeeInformation((Hourly)el[i]);

}

elseif (el[i] is Salaried)

{

ApplicationUtilities.DisplayDivider("Collect Salaried Information");

EmployeeInput.CollectEmployeeInformation(el[i]);

EmployeeInput.CollectorSalariedInformation((Salaried)el[i]);

EmployeeOutput.DisplayEmployeeInformation((Salaried)el[i]);

}

else

{

ApplicationUtilities.DisplayDivider("Collect Generic Information");

EmployeeInput.CollectEmployeeInformation(el[i]);

EmployeeOutput.DisplayEmployeeInformation(el[i]);

}

ApplicationUtilities.PauseExecution();

}

EmployeeOutput.DisplayNumberObjects();

ApplicationUtilities.TerminateApplication();

}

}

}

**Application Utilities**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace C# 6I

{

classApplicationUtilities

{

publicstaticvoid DisplayApplicationInformation()

{

Console.WriteLine("Abstract Classes");

Console.WriteLine("CIS247A, Week 6 Lab");

Console.WriteLine("Name: Kevin Nguyen");

Console.WriteLine("This program implements the Employee inheritance hierarchy and generalized \nmethods that demonstrates abstract classes, dynamic binding, and polymorphism.");

Console.WriteLine();

}

publicstaticvoid DisplayDivider(string outputTitle)

{

Console.WriteLine("\n\*\*\*\*\*\*\*\*\* " + outputTitle + " \*\*\*\*\*\*\*\*\*\n");

}

publicstaticvoid TerminateApplication()

{

DisplayDivider("Program Termination");

Console.Write("Thank you. Press any key to terminate the program...");

Console.ReadLine();

}

publicstaticvoid PauseExecution()

{

Console.Write("\nProgram paused, press any key to continue...");

Console.ReadLine();

Console.WriteLine();

}

}

}