# Exercise 1 – Student

using System;

namespace PRN292\_Lab1M\_1\_Student

{

public class Student

{

private int sid { get; set; }

private string name { get; set; }

private string faculty { get; set; }

private double mark;

public Student()

{ // default constructor

sid = 1;

name = "Nguyen Van A";

faculty = "IT";

mark = 0;

}

public Student(int sid, string name, string faculty, double mark)

{// constructor for create new

this.sid = sid;

this.name = name;

this.faculty = faculty;

Mark = mark;

}

public Student(Student st) // constructor for copy

{

this.sid = st.sid;

this.name = st.name;

this.faculty = st.faculty;

this.mark = st.mark;

}

public double Mark { get => mark; set => mark = value; }

public void Show()

{

Console.WriteLine("\tSID: " + sid);

Console.WriteLine("\tName: " + name);

Console.WriteLine("\tFaculty: " + faculty);

Console.WriteLine("\tMark: {0:F2}", mark);

}

}

class Tester

{

static int inputInt(string s)

{

while (true)

{

try

{

Console.Write(s);

int rs = int.Parse(Console.ReadLine());

return rs;

}

catch (Exception e)

{

Console.WriteLine("Input must be integer, try again!");

}

}

}

static double inputDouble(string s)

{

while (true)

{

try

{

Console.Write(s);

double rs = double.Parse(Console.ReadLine());

return rs;

}

catch (Exception e)

{

Console.WriteLine("Invalid must be a number, try again!");

}

}

}

static string inputString(string s)

{

while (true)

{

try

{

Console.Write(s);

string temp = Console.ReadLine().Trim();

if (temp.Length == 0) throw new Exception("Empty string");

return temp;

}

catch (Exception e)

{

Console.WriteLine("Invalid input, try again!");

}

}

}

static void Main(string[] args)

{

Student[] list;

int n;

Console.WriteLine("Number of students: ");

n = inputInt("\tn = ");

if (n > 0)

{

list = new Student[n];

Console.WriteLine("--------- Input Student List---------");

for (int i = 0; i < n; i++)

{

Console.WriteLine("\n\tStudent {0}", (i + 1));

int sid = inputInt("SID = ");

string name = inputString("Name = ");

string fal = inputString("Faculty = ");

double mark = inputDouble("Mark = ");

while (mark < 0 || mark > 10)

{

Console.WriteLine("Mark >= 0 and <=10, try again!");

mark = inputDouble("Mark = ");

}

list[i] = new Student(sid, name, fal, mark);

}

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

Console.WriteLine("\t\tSTUDENT LIST");

for (int i = 0; i < n; i++)

{

Console.WriteLine("\nStudent " + (i + 1));

list[i].Show();

}

}

Console.ReadLine();

}

}

}

# Exercise 2

using System;

namespace PRN292\_Lab1M\_2

{

public class Student

{

private int sid;

private string name;

private string faculty;

private double mark;

public Student()

{ // default constructor

sid = 1;

name = "Nguyen Van A";

faculty = "IT";

mark = 0;

}

public Student(int sid, string name, string faculty, double mark)

{// constructor for create new

this.sid = sid;

this.name = name;

this.faculty = faculty;

SetMark(mark);

}

public Student(Student st) // constructor for copy

{

this.sid = st.sid;

this.name = st.name;

this.faculty = st.faculty;

this.mark = st.mark;

}

public int GetSid()

{

return sid;

}

public void GetSid(int value)

{

sid = value;

}

public string GetName()

{

return name;

}

public void SetName(string value)

{

name = value;

}

public string GetFaculty()

{

return faculty;

}

public void SetFaculty(string value)

{

faculty = value;

}

public double GetMark()

{

return mark;

}

public void SetMark(double value)

{

if(value>=0 && value<=10) mark = value;

}

public void Show()

{

Console.WriteLine("\tSID: " + sid);

Console.WriteLine("\tName: " + name);

Console.WriteLine("\tFaculty: " + faculty);

Console.WriteLine("\tMark: {0:F2}", mark);

}

}

class Tester

{

static int InputInt(string s)

{

while (true)

{

try

{

Console.Write(s);

int rs = int.Parse(Console.ReadLine());

return rs;

}

catch (Exception e)

{

Console.WriteLine("Input must be integer, try again!");

}

}

}

static double InputDouble(string s)

{

while (true)

{

try

{

Console.Write(s);

double rs = double.Parse(Console.ReadLine());

return rs;

}

catch (Exception e)

{

Console.WriteLine("Invalid must be a number, try again!");

}

}

}

static string InputString(string s)

{

while (true)

{

try

{

Console.Write(s);

string temp = Console.ReadLine().Trim();

if (temp.Length == 0) throw new Exception("Empty string");

return temp;

}

catch (Exception e)

{

Console.WriteLine("Invalid Input, try again!");

}

}

}

static void InputStuList(ref int n,ref Student[] list)

{

Console.WriteLine("Number of students: ");

n = InputInt("\tn = ");

if (n > 0)

{

list = new Student[n];

Console.WriteLine("--------- Input Student List---------");

for (int i = 0; i < n; i++)

{

Console.WriteLine("\n\tStudent {0}", (i + 1));

int sid = InputInt("SID = ");

string name = InputString("Name = ");

string fal = InputString("Faculty = ");

double mark = InputDouble("Mark = ");

while (mark < 0 || mark > 10)

{

Console.WriteLine("Mark >= 0 and <=10, try again!");

mark = InputDouble("Mark = ");

}

list[i] = new Student(sid, name, fal, mark);

}

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

}

}

static void DisplayList(Student[] list,int n)

{

if (n > 0)

{

Console.WriteLine("\t\tSTUDENT LIST");

for (int i = 0; i < n; i++)

{

Console.WriteLine("\nStudent " + (i + 1));

list[i].Show();

}

}

}

static void Main(string[] args)

{

Student[] list=new Student[1];

int n=0;

InputStuList(ref n, ref list);

DisplayList(list, n);

Console.ReadLine();

}

}

}

# Exercise 3

using System;

using System.Collections;

using System.Collections.Generic;

namespace PRN292\_Lab1M\_3

{

public class People

{

protected string name;

protected int age;

protected string Name { get => name; set => name = value; }

protected int Age { get => age; set { if (value > 0 && value < 200) age = value; } }

public People()

{

name = "Nguyen Van A";

age = 1;

}

public People(string name, int age)

{

Name = name;

Age = age;

}

public People(People p)

{

name = p.Name;

age = p.Age;

}

public virtual void Show()

{

Console.WriteLine("\tName: " + name);

Console.WriteLine("\tAge: " + age);

}

}

public class Student : People

{

private int sid;

private string faculty;

private double mark;

public int Sid { get => sid; set => sid = value; }

public string Faculty { get => faculty; set => faculty = value; }

public double Mark { get => mark; set { if (value >= 0 && value <= 10) mark = value; } }

public Student():base()

{ // default constructor

sid = 1;

faculty = "IT";

mark = 0;

}

public Student(int sid, string faculty, double mark, string name,int age):base(name,age)

{// constructor for create new

this.sid = sid;

this.faculty = faculty;

Mark = mark;

}

public Student(Student st):base(st) // constructor for copy

{

sid = st.sid;

faculty = st.faculty;

mark = st.mark;

}

public override void Show()

{

Console.WriteLine("\tSID: " + sid);

base.Show();

Console.WriteLine("\tFaculty: " + faculty);

Console.WriteLine("\tMark: {0:F2}", mark);

}

}

class Tester

{

static int InputInt(string s)

{

while (true)

{

try

{

Console.Write(s);

int rs = int.Parse(Console.ReadLine());

return rs;

}

catch (Exception e)

{

Console.WriteLine("Input must be integer, try again!");

}

}

}

static double InputDouble(string s)

{

while (true)

{

try

{

Console.Write(s);

double rs = double.Parse(Console.ReadLine());

return rs;

}

catch (Exception e)

{

Console.WriteLine("Input must be a number, try again!");

}

}

}

static string InputString(string s)

{

while (true)

{

try

{

Console.Write(s);

string temp = Console.ReadLine().Trim();

if (temp.Length == 0) throw new Exception("Empty string");

return temp;

}

catch (Exception e)

{

Console.WriteLine("Invalid Input, try again!");

}

}

}

static void InputStuList(ref int n, ref List<People> list)

{

Console.WriteLine("Number of students: ");

n = InputInt("\tn = ");

if (n > 0)

{

list = new List<People>();

Console.WriteLine("--------- Input Student List---------");

for (int i = 0; i < n; i++)

{

Console.WriteLine("\n\tStudent {0}", (i + 1));

int sid = InputInt("SID = ");

string name = InputString("Name = ");

int age = InputInt("Age = ");

string fal = InputString("Faculty = ");

double mark = InputDouble("Mark = ");

while (mark < 0 || mark > 10)

{

Console.WriteLine("Mark >= 0 and <=10, try again!");

mark = InputDouble("Mark = ");

}

list.Add(new Student(sid, fal, mark, name, age));

}

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

}

}

static void DisplayList(List<People> list, int n)

{

if (n > 0)

{

Console.WriteLine("=============== STUDENT LIST ===============");

foreach (People p in list)

{

if (p is Student)

{

Console.WriteLine("- Student " + (list.IndexOf(p) + 1));

}

else

{

Console.WriteLine("- People " + (list.IndexOf(p) + 1));

}

p.Show();

}

}

else Console.WriteLine("Empty List!");

}

static void Main(string[] args)

{

List<People> list=new List<People>();

int n = 0;

InputStuList(ref n, ref list);

DisplayList(list, n);

Console.Write("\nPress any key to exit!");

Console.ReadLine();

}

}

}

# Exercise 4

using System;

using System.Collections.Generic;

using System.Linq;

namespace PRN292\_Lab1M\_4

{

    class Employee

    {

        protected string name;

        protected int birth;

        protected string degree;

        public string Name { get => name; set => name = value; }

        public string Degree { get => degree; set => degree = value; }

        public int Birth {

            get => birth;

            set {

                if(value>1920 && value<int.Parse(DateTime.Now.ToString("yyyy")))

                    birth = value;

            }

        }

        public Employee() { name = ""; degree = ""; }

        public Employee(string name, string degree, int birth)

        {

            Name = name;

            Degree = degree;

            Birth = birth;

        }

        public Employee(Employee e)

        {

            name = e.name;

            degree = e.degree;

            birth = e.birth;

        }

        public virtual void Input()

        {

            name = Program.InputString("\tEnter name: ");

            Birth = Program.InputInt("\tEnter year of birth: ");

            degree = Program.InputString("\tEnter degree: ");

        }

        public virtual void Display()

        {

            Console.WriteLine($"Name = {name}; Year of birth = {birth}; Degree = {degree}");

        }

    }

    class Manager : Employee

    {

        protected string position;

        protected int noOfWorkedDay;

        protected double salaryDegree;

        public int NoOfWorkedDay {

            get => noOfWorkedDay;

            set

            {

                if (value >= 0) noOfWorkedDay = value;

            }

        }

        public double SalaryDegree {

            get => salaryDegree;

            set {

                if (value >= 0) salaryDegree = value;

            }

        }

        public string Position { get => position; set => position = value; }

        public Manager() {}

        public Manager(string name, string degree, int birth, int noOfWorkedDay, double salaryDegree, string position) : base(name,degree,birth)

        {

            NoOfWorkedDay = noOfWorkedDay;

            SalaryDegree = salaryDegree;

            Position = position;

        }

        public Manager(Manager m):base(m)

        {

            noOfWorkedDay = m.NoOfWorkedDay;

            salaryDegree = m.SalaryDegree;

            position = m.Position;

        }

        public override void Display()

        {

            Console.WriteLine($"Name = {name}; Year of birth = {birth}; Degree = {degree}; Position = {position};Salary={GetSalary()}");

        }

        public double GetSalary()

        {

            return salaryDegree \* noOfWorkedDay;

        }

        public override void Input()

        {

            base.Input();

            position = Program.InputString("\tPostion: ");

            NoOfWorkedDay = Program.InputInt("\tNo of worked days in month: ");

            SalaryDegree = Program.InputInt("\tSalary degree: ");

        }

    }

    class Scientis : Manager

    {

        private int noOfArticle;

        public int NoOfArticle {

            get => noOfArticle;

            set {

                if (value >= 0) noOfArticle = value;

            }

        }

        public Scientis() { }

        public Scientis(string name, string degree, int birth, int noOfWorkedDay, double salaryDegree, string position, int noOfArticle) : base(name, degree, birth,noOfWorkedDay,salaryDegree,position)

        {

            NoOfArticle = noOfArticle;

        }

        public Scientis(Scientis sc) : base(sc)

        {

            noOfArticle = sc.noOfArticle;

        }

        public override void Display()

        {

            Console.WriteLine($"Name = {name}; Year of birth = {birth}; Degree = {degree}; Position = {position};Salary={GetSalary()}; No Of Articles = {noOfArticle}");

        }

        public override void Input()

        {

            base.Input();

            NoOfArticle = Program.InputInt("\tNumber of articles: ");

        }

    }

    class Staff : Employee

    {

        private double salary;

        public double Salary {

            get => salary;

            set

            {

                if (value >= 0) salary = value;

            }

        }

        public Staff()

        {

        }

        public Staff(string name, string degree, int birth, double salary) : base(name,degree,birth)

        {

            Salary = salary;

        }

        public Staff(Staff st):base(st)

        {

            salary = st.Salary;

        }

        public override void Input()

        {

            base.Input();

            salary = Program.InputDouble("\tEnter salary: ");

        }

        public override void Display()

        {

            Console.WriteLine($"Name = {name}; Year of birth = {birth}; Degree = {degree};Salary={salary}");

        }

    }

    class Program

    {

        public static int InputInt(string s)

        {

            while (true)

            {

                try

                {

                    Console.Write(s);

                    int rs = int.Parse(Console.ReadLine());

                    return rs;

                }

                catch (Exception e)

                {

                    Console.WriteLine("Input must be integer, try again!");

                }

            }

        }

        public static double InputDouble(string s)

        {

            while (true)

            {

                try

                {

                    Console.Write(s);

                    double rs = double.Parse(Console.ReadLine());

                    return rs;

                }

                catch (Exception e)

                {

                    Console.WriteLine("Input must be a number, try again!");

                }

            }

        }

        public static string InputString(string s)

        {

            while (true)

            {

                try

                {

                    Console.Write(s);

                    string temp = Console.ReadLine().Trim();

                    if (temp.Length == 0) throw new Exception("Empty string");

                    return temp;

                }

                catch (Exception e)

                {

                    Console.WriteLine("Invalid Input, try again!");

                }

            }

        }

        static void InputList(ref List<Employee> list,int type)

        {

            // 1 - Staff, 2 - Manager, 3 - Scientis

            int n;

            if(type == 1)

            {

                n = InputInt("Number of staffs: ");

                for (int i = 0; i < n; i++)

                {

                    Staff st = new Staff();

                    Console.WriteLine("\nStaff " + (i + 1));

                    st.Input();

                    list.Add(st);

                }

            }

            else if(type == 2)

            {

                n = InputInt("Number of managers: ");

                for (int i = 0; i < n; i++)

                {

                    Manager m = new Manager();

                    Console.WriteLine("\nManager " + (i + 1));

                    m.Input();

                    list.Add(m);

                }

            }

            else if(type == 3)

            {

                n = InputInt("Number of scientises: ");

                for (int i = 0; i < n; i++)

                {

                    Scientis sc = new Scientis();

                    Console.WriteLine("\nScientis " + (i + 1));

                    sc.Input();

                    list.Add(sc);

                }

            }

            Console.WriteLine("Added new employees successful!");

        }

        static void Show(List<Employee> list)

        {

            double salaryOfManager=0;

            double salaryOfScientis=0;

            double salaryOfStaff=0;

            if (list.Count() == 0)

            {

                Console.WriteLine("\nEmpty list!");

            }

            else

            {

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

                Console.WriteLine("\t\t\t\t\tEMPLOYEE LIST");

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

            }

            foreach(Employee e in list)

            {

                if(e is Staff)

                {

                    Staff st = (Staff)list[list.IndexOf(e)];

                    salaryOfStaff += st.Salary;

                    Console.Write((list.IndexOf(e) + 1) + ". ");

                    st.Display();

                }else if(e is Scientis)

                {

                    Scientis sc = (Scientis)list[list.IndexOf(e)];

                    salaryOfScientis += sc.GetSalary();

                    Console.Write((list.IndexOf(e) + 1) + ". ");

                    sc.Display();

                }else if(e is Manager)

                {

                    Manager m = (Manager)list[list.IndexOf(e)];

                    salaryOfManager += m.GetSalary();

                    Console.Write((list.IndexOf(e) + 1) + ". ");

                    m.Display();

                }

            }

            if (list.Count > 0)

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

            Console.WriteLine("\n\nTotal salary of managers: " + salaryOfManager);

            Console.WriteLine("Total salary of scientises: " + salaryOfScientis);

            Console.WriteLine("Total salary of staffs: " + salaryOfStaff);

        }

        static void Main(string[] args)

        {

            List<Employee> list = new List<Employee>();

            while (true)

            {

                Console.WriteLine("\n\n====================== EMPLOYEE MANAGE ======================");

                Console.WriteLine("\t1 - Create staff list");

                Console.WriteLine("\t2 - Create manager list");

                Console.WriteLine("\t3 - Create scientis list");

                Console.WriteLine("\t4 - View employee list");

                Console.WriteLine("\t5 - Remove all");

                Console.WriteLine("\t6 - Clear console");

                Console.WriteLine("\tOther - Exit");

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

                int choice = InputInt("Select your choose: ");

                switch (choice)

                {

                    case 1:

                        InputList(ref list, 1);

                        break;

                    case 2:

                        InputList(ref list, 2);

                        break;

                    case 3:

                        InputList(ref list, 3);

                        break;

                    case 4:

                        Show(list);

                        Console.ReadLine();

                        break;

                    case 5:

                        list.Clear();

                        Console.WriteLine("Removed all elements from list.");

                        break;

                    case 6:

                        Console.Clear();

                        Console.WriteLine("Press any keys to continue!");

                        Console.ReadLine();

                        Console.Clear();

                        break;

                    default:

                        return;

                }

            }

        }

    }

}