PRN211

SLOT 1,2,3 EXERCISES

14/01/2022

Full name: Dang Huu Quang

Student code: DE150199

Class: SE1507

File: OOP Exercises for Slot 123.docx

Exercise 1: Dice

Class Dice:

public class Dice  
**{** public int Side { get; set; }  
  
 public Dice(int *sides*)  
 {  
 Side = *sides*;  
 }  
  
 public int Roll()  
 {  
 return (new Random()).Next(1, Side);  
 }  
**}**

Class Program:

internal class Program  
**{** public static void *Main*(string[] *args*)  
 {  
 int sides;  
 int guessingSide;  
   
 InputSides:  
 Console.*Write*("Input number of sides of Dice: ");  
 if (!int.*TryParse*(Console.*ReadLine*(), out sides))  
 {  
 Console.*Write*("Input only the number! ");  
 goto InputSides;  
 }  
  
 InputGuessingSide:  
 Console.*Write*("Input guessing number: ");  
 if (!int.*TryParse*(Console.*ReadLine*(), out guessingSide))  
 {  
 Console.*Write*("Input only the number! ");  
 goto InputGuessingSide;  
 }  
   
 Dice dice = new Dice(sides);  
 int randomSide = dice.Roll();  
 if (randomSide == guessingSide) Console.*WriteLine*("You guessed right!");  
 else  
 {  
 Console.*WriteLine*("You guessed wrong!");  
 Console.*WriteLine*($"The random sides is {randomSide}!");  
 }  
 }  
**}**

Exercise 2: Create classes with the following components

Class Person

public class Person  
**{** public int Age { get; set; }  
 public string Name { get; set; }  
  
 public Person()  
 {  
 }  
 public Person(int *age*, string *name*)  
 {  
 Age = *age*;  
 Name = *name*;  
 }  
  
 public void DisplayPerson() {}  
 public void Input() {}  
  
 public override string ToString()  
 {  
 return $"Name: {Name}, Age: {Age}";  
 }  
**}**

Class Student

public class Student : Person  
**{** public double Gpa { get; set; }  
  
 public Student()  
 {  
 }  
  
 public Student(int *age*, string *name*, double *gpa*) : base(*age*, *name*)  
 {  
 Gpa = *gpa*;  
 }  
  
 public void DisplayStudent() {}  
 public void Input() {}  
  
 public override string ToString()  
 {  
 return base.ToString() + $", GPA: {Gpa}";  
 }  
**}**

Class Program

internal class Program  
**{** public static void *Main*(string[] *args*)  
 {  
 }  
**}**

Exercise 3:

Class Song

public class Song  
**{** public string TypeList {get; set; }  
 public string Name {get; set; }  
 public string Time {get; set; }  
  
 public Song(string *typeList*, string *name*, string *time*)  
 {  
 TypeList = *typeList*;  
 Name = *name*;  
 Time = *time*;  
 }  
**}**

Class Program

public static void *Main*(string[] *args*)  
 {  
 int numSongs;  
 string typeList;  
 List<Song> list = new List<Song>();  
  
 Console.*Write*("Input number of songs: ");  
 int.*TryParse*(Console.*ReadLine*(), out numSongs);  
  
 for (int i = 0; i < numSongs; i++)  
 {  
 string[] data = Console.*ReadLine*()?.Split('\_');  
 if (data != null) list.Add(new Song(data[0], data[1], data[2]));  
 }  
   
 Console.*Write*("Input type list: ");  
 typeList = Console.*ReadLine*();  
  
 switch (typeList)  
 {  
 case "all":  
 {  
 foreach (var song in list) Console.*WriteLine*(song.Name);  
 break;  
 }  
 default:  
 {  
 foreach (var song in list)  
 {  
 if (song.Name == typeList) Console.*WriteLine*(song.Name);  
 }  
 break;  
 }  
 }  
 }  
**}**

Exercise 4: Make a Fraction class in C#

Class Fraction

public class Fraction  
**{** public int Numerator { get; set; }  
 public int Denominator { get; set; }  
  
 public Fraction(int *numerator*, int *denominator*)  
 {  
 Numerator = *numerator*;  
 Denominator = *denominator*;  
 }  
**}**

Class Calculation

public class Calculation  
**{** public Fraction First { get; set; }  
 public Fraction Second { get; set; }  
 public Fraction Result { get; set; }  
  
 public Calculation(Fraction *first*, Fraction *second*)  
 {  
 First = *first*;  
 Second = *second*;  
 }  
  
 private int GCD(Fraction *fraction*)  
 {  
 int numerator = *fraction*.Numerator;  
 int denominator = *fraction*.Denominator;  
  
 while (denominator != 0)  
 {  
 int temp = denominator;  
 denominator = numerator % denominator;  
 numerator = temp;  
 }  
  
 return numerator;  
 }  
  
 public void Calculate(char *@operator*)  
 {  
 int numerator = 0, denominator = 0;  
 switch (*@operator*)  
 {  
 case '+':  
 {  
 numerator = First.Numerator \* Second.Denominator + Second.Numerator \* First.Denominator;  
 denominator = First.Denominator \* Second.Denominator;  
 break;  
 }  
 case '-':  
 {  
 numerator = First.Numerator \* Second.Denominator - Second.Numerator \* First.Denominator;  
 denominator = First.Denominator \* Second.Denominator;  
 break;  
 }  
 case '\*':  
 {  
 numerator = First.Numerator \* Second.Numerator;  
 denominator = First.Denominator \* Second.Denominator;  
 break;  
 }  
 case '/':  
 {  
 numerator = First.Numerator \* Second.Denominator;  
 denominator = First.Denominator \* Second.Numerator;  
 break;  
 }  
 }  
   
 Result = new Fraction(numerator, denominator);  
 }  
  
 public Fraction Normalized(Fraction *fraction*)  
 {  
 int x = GCD(*fraction*);  
 return new Fraction((*fraction*.Numerator /= x), (*fraction*.Denominator /= x));  
 }  
**}**

Class Program

internal class Program  
**{** public static void *Main*(string[] *args*)  
 {  
 Fraction first, second;  
 string[] data;  
 char @operator;  
   
 Console.*Write*("Input the first fraction: ");  
 data = Console.*ReadLine*()?.Split('/');  
 first = new Fraction(int.*Parse*(data[0]), int.*Parse*(data[1]));  
   
 Console.*Write*("Input the second fraction: ");  
 data = Console.*ReadLine*()?.Split('/');  
 second = new Fraction(int.*Parse*(data[0]), int.*Parse*(data[1]));  
   
 Console.*Write*("Input operator: ");  
 @operator = Console.*ReadKey*().KeyChar;  
  
 Calculation calculation = new Calculation(first, second);  
 calculation.Calculate(@operator);  
 Console.*WriteLine*($"\nThe result is: {calculation.Result.Numerator}/{calculation.Result.Denominator}");  
 Console.*ReadLine*();  
 }  
**}**

File: Lab1\_FPT\_C#\_OOP.doc

Exercise 1:

Class Point

class Point  
**{** public int X { get; set; }  
 public int Y { get; set; }  
  
 public Point(int *x*, int *y*)  
 {  
 X = *x*;  
 Y = *y*;  
 }  
  
 public override string ToString()  
 {  
 return $"({X},{Y})";  
 }  
**}**

Abstract class Shape

abstract class Shape  
**{** public Point StartPoint { get; set; }  
  
 protected Shape(Point *point*)  
 {  
 StartPoint = new Point(*point*.X, *point*.Y);  
 }  
  
 public void Move(Point *point*) {}  
   
 public abstract void Show();  
  
 public override string ToString()  
 {  
 return $"({StartPoint.X},{StartPoint.Y})";  
 }  
**}**

Class Line

class Line : Shape  
**{** public Point EndPoint { get; set; }  
  
 public Line(Point *startPoint*, Point *endPoint*) : base(*startPoint*)  
 {  
 EndPoint = new Point(*endPoint*.X, *endPoint*.Y);  
 }  
  
 public override void Show()  
 {  
 Console.*WriteLine*($"Line from ({StartPoint.X},{StartPoint.Y}) to ({EndPoint.X},{EndPoint.Y})");  
 }  
  
 public override string ToString()  
 {  
 return base.ToString() +  
 $" -> ({EndPoint.X},{EndPoint.Y})";  
 }  
**}**

Class Circle

class Circle : Shape  
**{** public double Radius { get; set; }  
  
 public Circle(Point *point*, double *radius*) : base(*point*)  
 {  
 Radius = *radius*;  
 }  
  
 public override void Show()  
 {  
 Console.*WriteLine*($"Circle with center at ({StartPoint.X},{StartPoint.Y}) and radius = {Radius}");  
 }  
   
 public override string ToString()  
 {  
 return base.ToString() +  
 $", r:{Radius}";  
 }  
}

Class Rectangle

class Rectangle : Shape  
**{** public Point Point2 { get; set; }  
 public Point Point3 { get; set; }  
  
 public Rectangle(Point *point1*, Point *point2*, Point *point3*) : base(*point1*)  
 {  
 Point2 = *point2*;  
 Point3 = *point3*;  
 }  
  
 public override void Show()  
 {  
 Console.*WriteLine*($"The rectangle is created with three points:\n" +  
 $"({StartPoint.X},{StartPoint.Y})\n" +  
 $"({Point2.X},{Point2.Y})\n" +  
 $"({Point3.X},{Point3.Y})");  
 }  
   
 public override string ToString()  
 {  
 return base.ToString() +  
 $" -> ({Point2.X},{Point2.Y}) -> ({Point3.X},{Point3.Y})";  
 }  
}

Class Polyline

class PolyLine : Shape  
**{** public Point[] Chain { get; set; }  
  
 public PolyLine(Point *point*, Point[] *chain*) : base(*point*)  
 {  
 Chain = new Point[*chain*.Length];  
 for (int i = 0; i < Chain.Length; i++)  
 {  
 Chain[i] = new Point(*chain*[i].X, *chain*[i].Y);  
 }  
 }  
  
 public override void Show()  
 {  
 Console.*WriteLine*($"The polyline is created with the chain points:");  
 Console.*WriteLine*($"({StartPoint.X},{StartPoint.Y})");  
 foreach (var point in Chain)  
 {  
 Console.*WriteLine*($"({point.X},{point.Y})");  
 }  
 }  
   
 public override string ToString()  
 {  
 string str;  
 str = base.ToString();  
 foreach (var point in Chain)  
 {  
 str = str + $" -> ({point.X},{point.Y})";  
 }  
 return str;  
 }  
**}**

Class Program

internal class Program  
**{** public static void *Main*(string[] *args*)  
 {  
 Line line = new Line(new Point(1,1), new Point(2,2));  
 line.Show();  
 Console.*WriteLine*(line.ToString());  
  
 Circle circle = new Circle(new Point(1, 1), 10);  
 circle.Show();  
 Console.*WriteLine*(circle);  
  
 Rectangle rectangle = new Rectangle(new Point(1,1), new Point(2,2), new Point(3,3));  
 rectangle.Show();  
 Console.*WriteLine*(rectangle);  
  
 Point[] chain =  
 {  
 new Point(2,2),  
 new Point(3,3),  
 new Point(4,4),  
 new Point(5,5),  
 };  
 PolyLine polyLine = new PolyLine(new Point(1,1), chain);  
 polyLine.Show();  
 Console.*WriteLine*(polyLine);  
  
 Console.*ReadLine*();  
 }  
**}**

Exercise 2:

Class Animal

abstract class Animal  
**{** public string Type { get; set; }  
  
 protected Animal(string *type*)  
 {  
 Type = *type*;  
 }  
  
 public abstract string GetSound();  
 public abstract string GetInformation();  
**}**

Class Dog

class Dog : Animal  
**{** public string Breed { get; set; }  
  
 public Dog(string *type*, string *breed*) : base(*type*)  
 {  
 Breed = *breed*;  
 }  
  
 public override string GetSound()  
 {  
 return "Gau Gau";  
 }  
  
 public override string GetInformation()  
 {  
 return $"Type: {Type}\n" +  
 $"Animal: Dog\n" +  
 $"Breed: {Breed}";  
 }  
**}**

Class Cat

class Cat : Animal  
**{** public Cat(string *type*) : base(*type*)  
 {  
 }  
  
 public void Climb(string *place*)  
 {  
 Console.*WriteLine*($"Cat climbs {*place*}");  
 }  
  
 public override string GetSound()  
 {  
 return "Meo meo";  
 }  
  
 public override string GetInformation()  
 {  
 return $"Type: {Type}\n" +  
 $"Animal: Cat";  
 }  
**}**

Class Duck

class Duck : Animal  
**{** public Duck(string *type*) : base(*type*)  
 {  
 }  
  
 public void Swim(string *place*)  
 {  
 Console.*WriteLine*($"Cat swims {*place*}");  
 }  
  
 public override string GetSound()  
 {  
 return "Wack wack";  
 }  
  
 public override string GetInformation()  
 {  
 return $"Type: {Type}\n" +  
 $"Animal: Duck";  
 }  
**}**

Class Program

internal class Program  
**{** public static void *Main*(string[] *args*)  
 {  
 Dog dog = new Dog("Mammal", "Pub");  
 Console.*WriteLine*(dog.GetSound());  
 Console.*WriteLine*(dog.GetInformation());  
  
 Cat cat = new Cat("Mammal");  
 Console.*WriteLine*(cat.GetSound());  
 Console.*WriteLine*(cat.GetInformation());  
 cat.Climb("Tree");  
  
 Duck duck = new Duck("Bird");  
 Console.*WriteLine*(duck.GetSound());  
 Console.*WriteLine*(duck.GetInformation());  
 duck.Swim("Pool");  
  
 Console.*ReadLine*();  
 }  
**}**

File: lab01\_9033\_OOP tieng Viet.pdf

Exercise 1:

Class Student

public class Student  
**{** public string Sid { get; set; }  
 public string Name { get; set; }  
 public string Department { get; set; }  
 public double Gpa { get; set; }  
  
 public Student()  
 {  
 }  
  
 public Student(string *sid*, string *name*, string *department*, double *gpa*)  
 {  
 Sid = *sid*;  
 Name = *name*;  
 Department = *department*;  
 Gpa = *gpa*;  
 }  
  
 public Student(Student *student*)  
 {  
 Sid = *student*.Sid;  
 Name = *student*.Name;  
 Department = *student*.Department;  
 Gpa = *student*.Gpa;  
 }  
  
 public override string ToString()  
 {  
 return $"╔══════════════ Student ══════════════╗\n" +  
 $"║ ID: {Sid,31} ║\n" +  
 $"║ Name: {Name,29} ║\n" +  
 $"║ Department: {Department,23} ║\n" +  
 $"║ GPA: {Gpa,30} ║\n" +  
 $"╚═════════════════════════════════════╝";  
 }  
**}**

Class Program

internal class Program  
**{** public static void *Main*(string[] *args*)  
 {  
 Student[] list;  
  
 Console.*Write*("Enter number of students: ");  
 int numOfStudents = int.*Parse*(Console.*ReadLine*() ?? string.*Empty*);  
  
 list = new Student[numOfStudents];  
 for (int i=0; i<list.Length ; i++)  
 {  
 list[i] = new Student();  
 Console.*WriteLine*($"\n══════════════ STUDENT ═══════════════");  
 Console.*Write*("Enter student's ID: "); list[i].Sid = Console.*ReadLine*();  
 Console.*Write*("Enter student's name: "); list[i].Name = Console.*ReadLine*();  
 Console.*Write*("Enter student's department: "); list[i].Department = Console.*ReadLine*();  
 Console.*Write*("Enter student's GPA: "); list[i].Gpa = Double.*Parse*(Console.*ReadLine*() ?? string.*Empty*);  
 }  
  
 Console.*WriteLine*("\n════════════ LIST STUDENT ═════════════");  
 foreach (var student in list) Console.*WriteLine*(student);  
 Console.*ReadLine*();  
 }  
**}**

Exercise 2:

Class Student

public class Student  
**{** private string sid;  
 private string name;  
 private string department;  
 private double gpa;  
  
 public Student()  
 {  
 }  
  
 public Student(string *sid*, string *name*, string *department*, double *gpa*)  
 {  
 this.sid = *sid*;  
 this.name = *name*;  
 this.department = *department*;  
 this.gpa = *gpa*;  
 }  
  
 public Student(Student *student*)  
 {  
 sid = *student*.GetSid();  
 name = *student*.GetName();  
 department = *student*.GetDepartment();  
 gpa = *student*.GetGpa();  
 }  
  
 public string GetSid()  
 {  
 return sid;  
 }  
  
 public void SetSid(string *sid*)  
 {  
 this.sid = *sid*;  
 }  
   
 public string GetName()  
 {  
 return name;  
 }  
  
 public void SetName(string *name*)  
 {  
 this.name = *name*;  
 }  
   
 public string GetDepartment()  
 {  
 return department;  
 }  
  
 public void SetDepartment(string *department*)  
 {  
 this.department = *department*;  
 }  
   
 public double GetGpa()  
 {  
 return gpa;  
 }  
  
 public void SetGpa(double *gpa*)  
 {  
 this.gpa = *gpa*;  
 }  
  
 public override string ToString()  
 {  
 return $"╔═════════════════════════════════════╗\n" +  
 $"║ ID: {sid,31} ║\n" +  
 $"║ Name: {name,29} ║\n" +  
 $"║ Department: {department,23} ║\n" +  
 $"║ GPA: {gpa,30} ║\n" +  
 $"╚═════════════════════════════════════╝";  
 }  
**}**

Class Program

internal class Program  
**{** public static Student *CreateAStudent*()  
 {  
 Student student = new Student();  
   
 Console.*WriteLine*($"\n══════════════ STUDENT ═══════════════");  
 Console.*Write*("Enter student's ID: "); student.SetSid(Console.*ReadLine*());  
 Console.*Write*("Enter student's name: "); student.SetName(Console.*ReadLine*());  
 Console.*Write*("Enter student's department: "); student.SetDepartment(Console.*ReadLine*());  
 Console.*Write*("Enter student's GPA: "); student.SetGpa(double.*Parse*(Console.*ReadLine*() ?? string.*Empty*));  
  
 return student;  
 }  
  
 public static Student[] *CreateStudentList*(int *numOfStudents*)  
 {  
 Student[] tempList = new Student[*numOfStudents*];  
 for (var i=0; i<tempList.Length ; i++)  
 {  
 tempList[i] = new Student(*CreateAStudent*());  
 }  
  
 return tempList;  
 }  
   
 public static void *PrintStudentList*(Student[] *list*)  
 {  
 Console.*WriteLine*("\n════════════ LIST STUDENT ═════════════");  
 foreach (var student in *list*) Console.*WriteLine*(student);  
 Console.*ReadLine*();  
 }  
  
 public static void *Main*(string[] *args*)  
 {  
 Console.*Write*("Enter number of students: ");  
 int numOfStudents = int.*Parse*(Console.*ReadLine*() ?? string.*Empty*);  
   
 Student[] list = *CreateStudentList*(numOfStudents);  
 *PrintStudentList*(list);  
 }  
**}**

Exercise 3:

Class People

public class People  
**{** public string Sid { get; set; }  
 public string Name { get; set; }  
  
 public People(string *sid*, string *name*)  
 {  
 Sid = *sid*;  
 Name = *name*;  
 }  
**}**

Class Student

public class Student : People  
**{** public string Department { get; set; }  
 public double Gpa { get; set; }  
  
 public Student(string *sid*, string *name*, string *department*, double *gpa*) : base(*sid*, *name*)  
 {  
 Department = *department*;  
 Gpa = *gpa*;  
 }  
   
 public override string ToString()  
 {  
 return $"╔═════════════════════════════════════╗\n" +  
 $"║ ID: {Sid,31} ║\n" +  
 $"║ Name: {Name,29} ║\n" +  
 $"║ Department: {Department,23} ║\n" +  
 $"║ GPA: {Gpa,30} ║\n" +  
 $"╚═════════════════════════════════════╝";  
 }  
**}**

Class Program

internal class Program  
**{** public static void *Main*(string[] *args*)  
 {  
 List<Student> list = new List<Student>();  
  
 Console.*Write*("Enter number of students: ");  
 int numOfStudents = int.*Parse*(Console.*ReadLine*() ?? string.*Empty*);  
  
 for (int i=0; i<numOfStudents ; i++)  
 {  
 Console.*WriteLine*($"\n══════════════ STUDENT ═══════════════");  
 Console.*Write*("Enter student's ID: "); string sid = Console.*ReadLine*();  
 Console.*Write*("Enter student's name: "); string name = Console.*ReadLine*();  
 Console.*Write*("Enter student's department: "); string department = Console.*ReadLine*();  
 Console.*Write*("Enter student's GPA: "); double gpa = Double.*Parse*(Console.*ReadLine*() ?? string.*Empty*);  
   
 list.Add(new Student(sid, name, department, gpa));  
 }  
  
 Console.*WriteLine*("\n════════════ LIST STUDENT ═════════════");  
 foreach (var student in list) Console.*WriteLine*(student);  
 Console.*ReadLine*();  
 }  
**}**

Exercise 4:

Class Employee

public abstract class Employee  
**{** public string Name { get; set; }  
 public int BirthYear { get; set; }  
 public string Certificate { get; set; }  
  
 protected Employee(string *name*, int *birthYear*, string *certificate*)  
 {  
 Name = *name*;  
 BirthYear = *birthYear*;  
 Certificate = *certificate*;  
 }  
  
 public abstract double GetSalary();  
  
 public override string ToString()  
 {  
 return $"║ {Name,-18}" +  
 $"║ {BirthYear,11}" +  
 $"║ {Certificate,-12}";  
 }  
**}**

Class Manager

public class Manager : Employee  
**{** public string Role { get; set; }  
 public int NumOfWorkdays { get; set; }  
 public double SalaryCoefficient { get; set; }  
  
 public Manager(  
 string *name*,   
 int *birthYear*,   
 string *certificate*,   
 string *role*,   
 int *numOfWorkdays*,   
 double *salaryCoefficient*) : base(*name*, *birthYear*, *certificate*)  
 {  
 Role = *role*;  
 NumOfWorkdays = *numOfWorkdays*;  
 SalaryCoefficient = *salaryCoefficient*;  
 }  
  
 public override double GetSalary()  
 {  
 return NumOfWorkdays \* SalaryCoefficient;  
 }  
  
 public override string ToString()  
 {  
 return base.ToString() +  
 $"║ {Role,-5}" +  
 $"║ {NumOfWorkdays,19}" +  
 $"║ {SalaryCoefficient,19}";  
 }  
**}**

Class Scientist

public class Scientist : Manager  
**{** public int NumOfPublishedArticles { get; set; }  
  
 public Scientist(  
 string *name*,   
 int *birthYear*,   
 string *certificate*,   
 string *role*,   
 int *numOfWorkdays*,   
 double *salaryCoefficient*,   
 int *numOfPublishedArticles*) : base(*name*, *birthYear*, *certificate*, *role*, *numOfWorkdays*, *salaryCoefficient*)  
 {  
 NumOfPublishedArticles = *numOfPublishedArticles*;  
 }  
  
 public override string ToString()  
 {  
 return base.ToString() +  
 $"║ {NumOfPublishedArticles,29}";  
 }  
**}**

Class LabStaff

public class LabStaff : Employee  
**{** public double Salary { get; set; }  
  
 public LabStaff(  
 string *name*,   
 int *birthYear*,   
 string *certificate*,   
 double *salary*) : base(*name*, *birthYear*, *certificate*)  
 {  
 Salary = *salary*;  
 }  
  
 public override double GetSalary()  
 {  
 return Salary;  
 }  
**}**

Class ScienceInstituteManagement

public class ScienceInstituteManagement  
**{** public List<Employee> List { get; set; }  
  
 public ScienceInstituteManagement()  
 {  
 List = new List<Employee>();  
 }  
  
 *//╚╝╬═╩╠╣╦╔╗║ BORDER* public int MainMenu()  
 {  
 Console.*Clear*();  
 Console.*WriteLine*("╔════════════════════════════════════════╗");  
 Console.*WriteLine*("║ SCIENCE INSTITUTE MANAGEMENT ║");  
 Console.*WriteLine*("╠════════════════════════════════════════╣");  
 Console.*WriteLine*("║ 1. Insert employees ║");  
 Console.*WriteLine*("║ 2. Print employees ║");  
 Console.*WriteLine*("║ 3. Print salary ║");  
 Console.*WriteLine*("║ 4. Exit ║");  
 Console.*WriteLine*("╚════════════════════════════════════════╝");  
   
 Console.*Write*("Please choose one option: "); int choice;  
 int.*TryParse*(Console.*ReadLine*(), out choice);  
 return choice;  
 }  
   
 public int EmployeeTypeMenu()  
 {  
 Console.*Clear*();  
 Console.*WriteLine*("╔════════════════════════════════════════╗");  
 Console.*WriteLine*("║ EMPLOYEE ║");  
 Console.*WriteLine*("╠════════════════════════════════════════╣");  
 Console.*WriteLine*("║ 1. Scientist ║");  
 Console.*WriteLine*("║ 2. Manager ║");  
 Console.*WriteLine*("║ 3. Lab Staff ║");  
 Console.*WriteLine*("║ 4. Go back ║");  
 Console.*WriteLine*("╚════════════════════════════════════════╝");  
   
 Console.*Write*("Please choose one option: "); int choice;  
 int.*TryParse*(Console.*ReadLine*(), out choice);  
 return choice;  
 }  
  
 public void InsertScientist()  
 {  
 Console.*Clear*();  
 Console.*Write*("Enter number of scientist(s): ");  
 int numOfScientists = int.*Parse*(Console.*ReadLine*() ?? string.*Empty*);  
  
 for (int i = 0; i < numOfScientists; i++)  
 {  
 Console.*Clear*();  
 Console.*WriteLine*($"═══════════════ Scientist {i+1}/{numOfScientists} ══════════════");  
 Console.*Write*("Insert name: "); string name = Console.*ReadLine*();  
 Console.*Write*("Insert birth year: "); int birthYear = int.*Parse*(Console.*ReadLine*() ?? String.*Empty*);  
 Console.*Write*("Insert certificate: "); string certificate = Console.*ReadLine*();  
 Console.*Write*("Insert role: "); string role = Console.*ReadLine*();  
 Console.*Write*("Insert number of workdays: "); int numOfWorkdays = int.*Parse*(Console.*ReadLine*() ?? String.*Empty*);  
 Console.*Write*("Insert salary coefficient: "); double salaryCoefficient = double.*Parse*(Console.*ReadLine*() ?? String.*Empty*);  
 Console.*Write*("Insert number of published articles: "); int numOfPublishedArticles = int.*Parse*(Console.*ReadLine*() ?? String.*Empty*);  
  
 List.Add(new Scientist(name, birthYear, certificate, role, numOfWorkdays, salaryCoefficient, numOfPublishedArticles));  
 Console.*Write*($"Insert scientist {i+1}/{numOfScientists} successfully!");  
 Console.*ReadLine*();  
 }  
 }  
  
 public void InsertManager()  
 {  
 Console.*Clear*();  
 Console.*Write*("Enter number of Manager(s): ");  
 int numOfManagers = int.*Parse*(Console.*ReadLine*() ?? string.*Empty*);  
  
 for (int i = 0; i < numOfManagers; i++)  
 {  
 Console.*Clear*();  
 Console.*WriteLine*($"════════════════ Manager {i+1}/{numOfManagers} ═══════════════");  
 Console.*Write*("Insert name: "); string name = Console.*ReadLine*();  
 Console.*Write*("Insert birth year: "); int birthYear = int.*Parse*(Console.*ReadLine*() ?? String.*Empty*);  
 Console.*Write*("Insert certificate: "); string certificate = Console.*ReadLine*();  
 Console.*Write*("Insert role: "); string role = Console.*ReadLine*();  
 Console.*Write*("Insert number of workdays: "); int numOfWorkdays = int.*Parse*(Console.*ReadLine*() ?? String.*Empty*);  
 Console.*Write*("Insert salary coefficient: "); double salaryCoefficient = double.*Parse*(Console.*ReadLine*() ?? String.*Empty*);  
  
 List.Add(new Manager(name, birthYear, certificate, role, numOfWorkdays, salaryCoefficient));  
 Console.*Write*($"Insert manager {i+1}/{numOfManagers} successfully!");  
 Console.*ReadLine*();  
 }  
 }  
  
 public void InsertLabStaff()  
 {  
 Console.*Clear*();  
 Console.*Write*("Enter number of lab staff(s): ");  
 int numOfLabStaffs = int.*Parse*(Console.*ReadLine*() ?? string.*Empty*);  
  
 for (int i = 0; i < numOfLabStaffs; i++)  
 {  
 Console.*Clear*();  
 Console.*WriteLine*($"═══════════════ Lab Staff {i + 1}/{numOfLabStaffs} ══════════════");  
 Console.*Write*("Insert name: ");  
 string name = Console.*ReadLine*();  
 Console.*Write*("Insert birth year: ");  
 int birthYear = int.*Parse*(Console.*ReadLine*() ?? String.*Empty*);  
 Console.*Write*("Insert certificate: ");  
 string certificate = Console.*ReadLine*();  
 Console.*Write*("Insert salary: ");  
 double salary = double.*Parse*(Console.*ReadLine*() ?? String.*Empty*);  
  
 List.Add(new LabStaff(name, birthYear, certificate, salary));  
 Console.*Write*($"Insert lab staff {i + 1}/{numOfLabStaffs} successfully!");  
 Console.*ReadLine*();  
 }  
 }  
  
 *//╚╝╬═╩╠╣╦╔╗║ BORDER* public void InsertEmployee()  
 {  
 switch (EmployeeTypeMenu())  
 {  
 case 1: InsertScientist(); break;  
 case 2: InsertManager(); break;  
 case 3: InsertLabStaff(); break;  
 }  
 }  
  
 public void PrintScientist()  
 {  
 Console.*Clear*();  
 Console.*WriteLine*("╔═══════════════════╦════════════╦═════════════╦══════╦════════════════════╦════════════════════╦══════════════════════════════╦════════╗");  
 Console.*WriteLine*("║ Name ║ Birth year ║ Certificate ║ Role ║ Number of workdays ║ Salary coefficient ║ Number of published articles ║ Salary ║");  
 Console.*WriteLine*("╠═══════════════════╬════════════╬═════════════╬══════╬════════════════════╬════════════════════╬══════════════════════════════╬════════╣");  
   
 foreach (var employee in List)  
 {  
 if (employee.GetType() == typeof(Scientist))  
 {  
 Console.*WriteLine*(employee + $"║ {employee.GetSalary(),7}║");  
 }  
 }  
 Console.*WriteLine*("╚═══════════════════╩════════════╩═════════════╩══════╩════════════════════╩════════════════════╩══════════════════════════════╩════════╝");  
 Console.*ReadLine*();  
 }  
   
 public void PrintManager()  
 {  
 Console.*Clear*();  
 Console.*WriteLine*("╔═══════════════════╦════════════╦═════════════╦══════╦════════════════════╦════════════════════╦════════╗");  
 Console.*WriteLine*("║ Name ║ Birth year ║ Certificate ║ Role ║ Number of workdays ║ Salary coefficient ║ Salary ║");  
 Console.*WriteLine*("╠═══════════════════╬════════════╬═════════════╬══════╬════════════════════╬════════════════════╬════════╣");  
   
 foreach (var employee in List)  
 {  
 if (employee.GetType() == typeof(Manager))  
 {  
 Console.*WriteLine*(employee + $"║ {employee.GetSalary(),7}║");  
 }  
 }  
 Console.*WriteLine*("╚═══════════════════╩════════════╩═════════════╩══════╩════════════════════╩════════════════════╩════════╝");  
 Console.*ReadLine*();  
 }  
   
 public void PrintLabStaff()  
 {  
 Console.*Clear*();  
 Console.*WriteLine*("╔═══════════════════╦════════════╦═════════════╦════════╗");  
 Console.*WriteLine*("║ Name ║ Birth year ║ Certificate ║ Salary ║");  
 Console.*WriteLine*("╠═══════════════════╬════════════╬═════════════╬════════╣");  
   
 foreach (var employee in List)  
 {  
 if (employee.GetType() == typeof(LabStaff))  
 {  
 Console.*WriteLine*(employee + $"║ {employee.GetSalary(),7}║");  
 }  
 }  
 Console.*WriteLine*("╚═══════════════════╩════════════╩═════════════╩════════╝");  
 Console.*ReadLine*();  
 }  
   
 public void PrintEmployee()  
 {  
 switch (EmployeeTypeMenu())  
 {  
 case 1: PrintScientist(); break;  
 case 2: PrintManager(); break;  
 case 3: PrintLabStaff(); break;  
 }  
 }  
  
 public double GetScientistSalary()  
 {  
 double sumSalary = 0;  
 foreach (var employee in List)  
 {  
 if (employee.GetType() == typeof(Scientist))  
 {  
 sumSalary += employee.GetSalary();  
 }  
 }  
  
 return sumSalary;  
 }  
  
 public double GetManagerSalary()  
 {  
 double sumSalary = 0;  
 foreach (var employee in List)  
 {  
 if (employee.GetType() == typeof(Manager))  
 {  
 sumSalary += employee.GetSalary();  
 }  
 }  
   
 return sumSalary;  
 }  
  
 public double GetLabStaffSalary()  
 {  
 double sumSalary = 0;  
 foreach (var employee in List)  
 {  
 if (employee.GetType() == typeof(LabStaff))  
 {  
 sumSalary += employee.GetSalary();  
 }  
 }  
   
 return sumSalary;  
 }  
   
 public void PrintSalary()  
 {  
 Console.*Clear*();  
 Console.*WriteLine*("╔═══════════════════╦════════════╗");  
 Console.*WriteLine*("║ Type ║ Salary ║");  
 Console.*WriteLine*("╠═══════════════════╬════════════╣");  
 Console.*WriteLine*($"║ Scientist ║ {GetScientistSalary(),11}║");  
 Console.*WriteLine*($"║ Manager ║ {GetManagerSalary(),11}║");  
 Console.*WriteLine*($"║ Lab staff ║ {GetLabStaffSalary(),11}║");  
 Console.*WriteLine*("╚═══════════════════╩════════════╝");  
 Console.*ReadLine*();  
 }  
**}**

Class Program

internal class Program  
**{** public static void *Main*(string[] *args*)  
 {  
 ScienceInstituteManagement m = new ScienceInstituteManagement();  
 bool isChecked = false;  
  
 while (!isChecked)  
 {  
 switch (m.MainMenu())  
 {  
 case 1: m.InsertEmployee(); break;  
 case 2:  
 {  
 if (!m.List.Any())  
 {  
 Console.*Clear*();  
 Console.*Write*("The list is empty!");  
 Console.*ReadLine*();  
 }  
 else m.PrintEmployee();  
 break;  
 }  
 case 3:  
 {  
 if (!m.List.Any())  
 {  
 Console.*Clear*();  
 Console.*Write*("The list is empty!");  
 Console.*ReadLine*();  
 }  
 else m.PrintSalary();  
 break;  
 }  
 case 4: isChecked = true; break;  
 default:  
 {  
 Console.*Clear*();  
 Console.*Write*("Insert '4' to quit!");  
 Console.*ReadLine*();  
 break;  
 }  
 }  
 }  
 }  
}