**HOME WORK 8/9/2023  
LÊ HỮU HOÀN – 20IT684**

Bài tập 1: Tạo lớp Student có các dữ liệu và phương thức sau:

* SID (mã số sinh viên)
* Tên sinh viên
* Khoa
* Điểm TB
* Thêm các property cho các dữ liệu thành viên trên
* Viết các phương thức hiển thị thông tin của sinh viên.

Tạo lớp Tester, trong lớp này chỉ chứa duy nhất hàm main(). Hàm cho phép người dung nhập vào số n là số sinh viên, sau đó lần lượt tạo các đối tượng sinh viên và add vào danh sách sinh viên theo những thông tin do user nhập vào (dùng vòng lặp for). Cuối cùng xuất ra danh sách chi tiết thông tin sinh viên.

Yêu cầu:

* Sinh viên xây dựng chương trình theo nội dung mô tả bên trên.
* Compile & Build chương trình.
* Run chương trình ở hai chế độ debug và không debug.
* Chạy từng bước chương trình trong chế độ debug: dùng breakpoint hoặc chạy từng dòng lệnh. Kiểm tra những giá trị của các biến trong chương trình ở cửa sổ Watch.

CONSOLE

A computer screen with a black screen

Description automatically generated

CODE:

|  |
| --- |
| using System;  namespace OOP  {  internal class Program  {  static void Main(string[] args)  {  Student[] student;  int n;  Console.WriteLine("Nhap so luong sv");  n = int.Parse(Console.ReadLine());  student = new Student[n]; //tao mang n phan tu  Console.WriteLine("\n ====NHAP DS SINH VIEN====");  for (int i = 0; i < n; i++) //Lap n lan nhap thong tin sv  {  student[i] = new Student();  Console.Write("Nhap MaSV {0}:", i + 1);  student[i].SID = int.Parse(Console.ReadLine());  Console.Write("Ho ten SV:");  student[i].TenSV = Console.ReadLine();  Console.Write("Nhap khoa:");  student[i].Khoa = Console.ReadLine();  Console.Write("Nhap Diem TB:");  student[i].DiemTB = float.Parse(Console.ReadLine());  }  //Xuat DS Sinh vien  Console.WriteLine("\n ====XUAT DS SINH VIEN====");  foreach (Student sv in student)  sv.PrintInformation();  Console.ReadLine();  }  }  } |

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace OOP  {  internal class Student  {    public int SID { get; set; }  public String TenSV { get; set; }  public String Khoa { get; set; }  public float DiemTB { get; set; }  public Student()  {  }  public Student(Student student)  {  SID = student.SID;  TenSV = student.TenSV;  Khoa = student.Khoa;  DiemTB = student.DiemTB;  }  public Student(int id, String name, String khoa, float diem)  {  SID = id;  TenSV = name;  Khoa = khoa;  DiemTB = diem;  }  public void PrintInformation()  {  Console.WriteLine($"- MSSV:{SID} - Ten SV:{TenSV} - Khoa:{Khoa} - Diem TB:{DiemTB}");  }  }  } |

Bài tập 2:

* Trong lớp Student trên không dùng Property cho các dữ liệu mà thay vào dùng phương thức getter và setter.
* Không viết lệnh nhập xuất danh sách sinh viên trực tiếp trong hàm Main() mà hãy viết hàm nhập 1 sinh viên Nhap1SV(), hàm nhập danh sách sinh viên NhapDS() và hàm xuất danh sách sinh viên XuatDS(), gọi hàm Show() đã viết. Sau đó trong hàm Main() gọi các hàm NhapDS(), XuatDS().
* Viết lại chương trình theo yêu cầu trên.

Console

A computer screen shot of a black square with white text

Description automatically generated

Code:

|  |
| --- |
| using System;  namespace OOP\_02  {  internal class Program  {  static public Student[] NhapDS()  {  Console.WriteLine("Nhap so luong sv");  int n = int.Parse(Console.ReadLine());  Student[] students = new Student[n];  Console.WriteLine("Hello World!");  Console.WriteLine("\n ====NHAP DS SINH VIEN====");  for (int i = 0; i < n; i++)  {  students[i] = new Student();  students[i].NhapSV();  }  return students;  }  static public void XuatDS(Student[] students)  {  Console.WriteLine("\n ====XUAT DS SINH VIEN====");  foreach (Student sv in students)  sv.PrintInformation();  Console.ReadLine();  }  static void Main(string[] args)  {  Student[] student = NhapDS();  XuatDS(student);  }  }  } |

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace OOP\_02  {  internal class Student  {  private int SID;  private string TenSV;  private string Khoa;  private float DiemTB;  public Student()  {  }  public void setID(int id)  {  SID = id;  }  public int getID()  {  return SID;  }  public void setName(string name)  {  TenSV = name;  }  public string getName()  {  return TenSV;  }  public void setDepartment(string department)  {  Khoa = department;  }  public string getDepartMent()  {  return Khoa;  }  public void setPoint(float point)  {  DiemTB = point;  }  public float getPoint()  {  return DiemTB;  }  public Student(Student student)  {  SID = student.SID;  TenSV = student.TenSV;  Khoa = student.Khoa;  DiemTB = student.DiemTB;  }  public Student(int id, String name, String khoa, float diem)  {  SID = id;  TenSV = name;  Khoa = khoa;  DiemTB = diem;  }  public void PrintInformation()  {  Console.WriteLine($"- MSSV:{SID} - Ten SV:{TenSV} - Khoa:{Khoa} - Diem TB:{DiemTB}");  }  public void NhapSV()  {  Console.Write("Nhap MaSV :");  SID = int.Parse(Console.ReadLine());  Console.Write("Ho ten SV:");  TenSV = Console.ReadLine();  Console.Write("Nhap khoa:");  Khoa = Console.ReadLine();  Console.Write("Nhap Diem TB:");  DiemTB = float.Parse(Console.ReadLine());  }  }  } |

Bài tập 3:

* Viết lại chương trình trên theo cách tạo thêm một lớp là People làm lớp cơ sở cho lớp Student. Sinh viên chọn những field thích hợp để đưa lên lớp cơ sở People...
* Viết lại phần hàm Main trong lớp Tester theo yêu cầu sau:
* Sử dụng lớp collection là List để chứa danh sách sinh viên thay thế cho mảng sinh viên trong bài tập trên
* Sử dụng lớp collection là ArrayList để chứa danh sách sinh viên thay vì mảng.
* Tìm hiểu chức năng hỗ trợ Refactor trong VS.NET, sử dụng chức năng Encapsulate Fields để tạo các property trong khi viết chương trình C#.

Hướng dẫn: Sử dụng chức năng Refactor: Encapsulate Fields bằng cách khi tạo các field

của lớp: click phải chuột vào tên của field cần tạo property, menu ngữ cảnh xuất hiện, chọn

chức năng Refactor và chọn mục Encapsulate Fields. Một dialog Encapsulate Field xuất

hiện cho phép mô tả property. Bước cuối cùng chọn OK, VS.NET sẽ phát sinh code tạo

property.

Console

A computer screen with a black screen

Description automatically generated

Code:

|  |
| --- |
| class Program  {  static public List<Student> NhapDS()  {  Console.WriteLine("Nhap so luong sv");  int n = int.Parse(Console.ReadLine());  List<Student> students = new List<Student>();  Console.WriteLine("Hello World!");  Console.WriteLine("\n ====NHAP DS SINH VIEN====");  for (int i = 0; i < n; i++)  {  Student std = new Student();  std.NhapSV();  students.Add(std);  }  return students;  }  static public void XuatDS(List<Student> students)  {  Console.WriteLine("\n ====XUAT DS SINH VIEN====");  foreach (Student sv in students)  sv.PrintInformation();  Console.ReadLine();  }  static public ArrayList ANhapDS()  {  Console.WriteLine("Nhap so luong sv");  int n = int.Parse(Console.ReadLine());  ArrayList students = new ArrayList();  Console.WriteLine("Hello World!");  Console.WriteLine("\n ====NHAP DS SINH VIEN====");  for (int i = 0; i < n; i++)  {  Student std = new Student();  std.NhapSV();  students.Add(std);  }  return students;  }  static public void AXuatDS(ArrayList students)  {  Console.WriteLine("\n ====XUAT DS SINH VIEN====");  foreach (Student sv in students)  sv.PrintInformation();  Console.ReadLine();  }  static void Main(string[] args)  {  List<Student> student = NhapDS();  XuatDS(student);  ArrayList std2 = ANhapDS();  AXuatDS(std2);  }  } |

|  |
| --- |
| class People  {  public string Name { get; set; }  public int Age { get; set; }  public People()  {    }  public People(string name, int age)  {  Name = name;  Age = age;  }  } |

|  |
| --- |
| class Student: People  {  public int SID { get; set; }  public String Khoa { get; set; }  public float DiemTB { get; set; }  public Student()  {  }  public Student(int id, String name,int age, String khoa, float diem):base(name,age)  {  SID = id;  Khoa = khoa;  DiemTB = diem;  }  public void PrintInformation()  {  Console.WriteLine($"- MSSV:{SID} - Ten:{Name} - Age:{Age} - Khoa:{Khoa} - Diem TB:{DiemTB}");  }  public void NhapSV()  {  Console.Write("Nhap MaSV :");  SID = int.Parse(Console.ReadLine());  Console.Write("Ho ten SV:");  Name = Console.ReadLine();  Console.Write("Nhap Age:");  Age = int.Parse(Console.ReadLine());  Console.Write("Nhap khoa:");  Khoa = Console.ReadLine();  Console.Write("Nhap Diem TB:");  DiemTB = float.Parse(Console.ReadLine());  }  }  } |

Bài tập 4:

Thiết kế chương trình quản lý các đối tượng sau trong một Viện khoa học: nhà khoa học, nhà

quản lý và nhân viên phòng thí nghiệm. Một nhà khoa học cũng có thể làm công tác quản lý. Các

thành phần dữ liệu của các đối tượng trên:

 Nhà khoa học: họ tên, năm sinh, bằng cấp, chức vụ, số bài báo đã công bố, số ngày công

trong tháng, bậc lương

 Nhà quản lý: họ tên, năm sinh, bằng cấp, chức vụ, số ngày công trong tháng, bậc lương

 Nhân viên phòng thí nghiệm: họ tên, năm sinh, bằng cấp, lương trong tháng.

Biết rằng nhân viên phòng thí nghiệm lãnh lương khoán, còn lương của nhà khoa học và nhà

quản lý bằng số ngày công trong tháng \* bậc lương. Nhập, xuất danh sách nhân viên và in tổng

lương đã chi trả cho từng loại đối tượng.

Console

A computer screen with a black screen

Description automatically generated

Code:

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  List<Employee> employees = new List<Employee>();  // Nhập danh sách nhân viên  Console.WriteLine("Nhập thông tin nhà khoa học:");  Scientist scientist = ReadScientistInformation();  employees.Add(scientist);  Console.WriteLine("Nhập thông tin nhà quản lý:");  Manager manager = ReadManagerInformation();  employees.Add(manager);  Console.WriteLine("Nhập thông tin nhân viên phòng thí nghiệm:");  LaboratoryStaff laboratoryStaff = ReadLaboratoryStaffInformation();  // Chuyển đổi thông tin nhân viên phòng thí nghiệm thành đối tượng Employee  Employee employee = ConvertToEmployee(laboratoryStaff);  employees.Add(employee);  // Tính tổng lương đã chi trả cho từng loại đối tượng  double totalScientistSalary = CalculateTotalScientistSalary(employees);  double totalManagerSalary = CalculateTotalManagerSalary(employees);  double totalLaboratoryStaffSalary = CalculateTotalLaboratoryStaffSalary(employees);  // In tổng lương đã chi trả cho từng loại đối tượng  Console.WriteLine("Tổng lương đã chi trả cho nhà khoa học: " + totalScientistSalary);  Console.WriteLine("Tổng lương đã chi trả cho nhà quản lý: " + totalManagerSalary);  Console.WriteLine("Tổng lương đã chi trả cho nhân viên phòng thí nghiệm: " + totalLaboratoryStaffSalary);  Console.ReadLine();  }  static Scientist ReadScientistInformation()  {  Console.Write("Họ tên: ");  string scientistName = Console.ReadLine();  Console.Write("Năm sinh: ");  int scientistBirthYear = Convert.ToInt32(Console.ReadLine());  Console.Write("Bằng cấp: ");  string scientistDegree = Console.ReadLine();  Console.Write("Chức vụ: ");  string scientistPosition = Console.ReadLine();  Console.Write("Số bài báo đã công bố: ");  int scientistPublishedPapers = Convert.ToInt32(Console.ReadLine());  Console.Write("Số ngày công trong tháng: ");  int scientistWorkDays = Convert.ToInt32(Console.ReadLine());  Console.Write("Bậc lương: ");  int scientistSalaryRank = Convert.ToInt32(Console.ReadLine());  Scientist scientist = new Scientist  {  FullName = scientistName,  BirthYear = scientistBirthYear,  Degree = scientistDegree,  Position = scientistPosition,  PublishedPapers = scientistPublishedPapers,  WorkDays = scientistWorkDays,  SalaryRank = scientistSalaryRank  };  return scientist;  }  static Manager ReadManagerInformation()  {  Console.Write("Họ tên: ");  string managerName = Console.ReadLine();  Console.ReadLine();  Console.Write("Năm sinh: ");  int managerBirthYear = Convert.ToInt32(Console.ReadLine());  Console.Write("Bằng cấp: ");  string managerDegree = Console.ReadLine();  Console.Write("Chức vụ: ");  string managerPosition = Console.ReadLine();  Console.Write("Số ngày công trong tháng: ");  int managerWorkDays = Convert.ToInt32(Console.ReadLine());  Console.Write("Bậc lương: ");  int managerSalaryRank = Convert.ToInt32(Console.ReadLine());  Manager manager = new Manager  {  FullName = managerName,  BirthYear = managerBirthYear,  Degree = managerDegree,  Position = managerPosition,  WorkDays = managerWorkDays,  SalaryRank = managerSalaryRank  };  return manager;  }  static LaboratoryStaff ReadLaboratoryStaffInformation()  {  Console.Write("Họ tên: ");  string laboratoryStaffName = Console.ReadLine();  Console.Write("Năm sinh: ");  int laboratoryStaffBirthYear = Convert.ToInt32(Console.ReadLine());  Console.Write("Bằng cấp: ");  string laboratoryStaffDegree = Console.ReadLine();  Console.Write("Lương trong tháng: ");  double laboratoryStaffMonthlySalary = Convert.ToDouble(Console.ReadLine());  LaboratoryStaff laboratoryStaff = new LaboratoryStaff  {  FullName = laboratoryStaffName,  BirthYear = laboratoryStaffBirthYear,  Degree = laboratoryStaffDegree,  MonthlySalary = laboratoryStaffMonthlySalary  };  return laboratoryStaff;  }  static Employee ConvertToEmployee(LaboratoryStaff laboratoryStaff)  {  Employee employee = new Employee  {  FullName = laboratoryStaff.FullName,  BirthYear = laboratoryStaff.BirthYear,  Degree = laboratoryStaff.Degree,  Position = "Nhân viên phòng thí nghiệm"  };  return employee;  }  static double CalculateTotalScientistSalary(List<Employee> employees)  {  double totalSalary = 0;  foreach (Employee employee in employees)  {  if (employee is Scientist scientist)  {  totalSalary += scientist.CalculateSalary();  }  }  return totalSalary;  }  static double CalculateTotalManagerSalary(List<Employee> employees)  {  double totalSalary = 0;  foreach (Employee employee in employees)  {  if (employee is Manager manager)  {  totalSalary += manager.CalculateSalary();  }  }  return totalSalary;  }  static double CalculateTotalLaboratoryStaffSalary(List<Employee> employees)  {  double totalSalary = 0;  foreach (Employee employee in employees)  {  if (!(employee is Scientist) && !(employee is Manager))  {  // Chuyển đổi đối tượng Employee thành đối tượng LaboratoryStaff  LaboratoryStaff laboratoryStaff = ConvertToLaboratoryStaff(employee);  totalSalary += laboratoryStaff.MonthlySalary;  }  }  return totalSalary;  }  static LaboratoryStaff ConvertToLaboratoryStaff(Employee employee)  {  LaboratoryStaff laboratoryStaff = new LaboratoryStaff  {  FullName = employee.FullName,  BirthYear = employee.BirthYear,  Degree = employee.Degree,  MonthlySalary = 0 // Giá trị lương không được sử dụng cho nhân viên phòng thí nghiệm  };  return laboratoryStaff;  }  } |

|  |
| --- |
| class Employee  {  public string FullName { get; set; }  public int BirthYear { get; set; }  public string Degree { get; set; }  public string Position { get; set; }  } |

|  |
| --- |
| internal class LaboratoryStaff  {  public string FullName { get; set; }  public int BirthYear { get; set; }  public string Degree { get; set; }  public double MonthlySalary { get; set; }  } |

|  |
| --- |
| class Manager : Employee  {  public int WorkDays { get; set; }  public int SalaryRank { get; set; }  public double CalculateSalary()  {  return WorkDays \* SalaryRank;  }  } |

|  |
| --- |
| class Scientist : Employee  {  public int PublishedPapers { get; set; }  public int WorkDays { get; set; }  public int SalaryRank { get; set; }  public double CalculateSalary()  {  return WorkDays \* SalaryRank;  }  } |