BÀI 6. THỰC HÀNH

Sinh viên được yêu cầu: Tạo **`Project Console`** và đặt tên project theo quy tắc: **`HoTen_masv_proj62`** (ví dụ **`NguyenLeHa_201987435_proj62`**) và thực hiện các công việc dưới sau đây:

1) Tạo giao diện `IVehicle`, sau đó tạo có 2 phương thức:

- `void Input();`
- `void Output();`

2) Tạo lớp `Vehicle` thực thi interface `IVehicle`, và tạo có 3 thuộc tính :

- `string id; (mã định danh)`
- `string maker; (hãng sản xuất)`
- `string model; (model, tên xe)`
- `int year; (năm sản xuất)`
- `double price; (giá tiền)`

- Thiết lập và lấy ra các giá trị cho từng thuộc tính.

- Viết phương thức khởi tạo không tham số, một tham số và năm tham số.

- Viết phương thức `virtual Input()` để nhập dữ liệu cho năm thuộc tính.

- Viết phương thức `virtual Output()` để hiển thị dữ liệu cho năm thuộc tính.

- Viết đè phương thức `Equals()`, nếu id bằng nhau thì hai đối tượng là bằng nhau.

- Viết đè phương thức `ToString()`, trả về xâu có năm thuộc tính trên.

3) Tạo lớp `Car` thừa kế từ lớp Vehicles và có thêm 1 thuộc tính color (màu sắc) kiểu string.

Viết phương thức khởi tạo không tham số và sáu tham số, gọi phương thức khởi tạo ở lớp cơ sở.

Viết đè phương thức `Input()`, trong đó có gọi phương thức Input() ở lớp cơ sở.

Viết đè phương thức `Output()`, trong đó có gọi phương thức Output () ở lớp cơ sở.

4) Tạo lớp `Truck` thừa kế từ lớp Vehicles và có thêm 1 thuộc tính truckload (tải trọng) kiểu int.

Viết phương thức khởi tạo không tham số và sáu tham số, gọi phương thức khởi tạo ở lớp cơ sở.

Viết đè phương thức `Input()`, trong đó có gọi phương thức Input() ở lớp cơ sở.

Viết đè phương thức `Output()`, trong đó có gọi phương thức Output () ở lớp cơ sở.

4) **Viết mã trong phương thức `Main` ở lớp Program để kiểm tra sự hoạt động của các lớp đã viết:**

1. Nhập dữ liệu
2. Hiển thị dữ liệu
3. Tìm kiếm theo id
4. Tìm kiếm theo maker
5. Sắp xếp theo theo price
6. Sắp xếp theo year
7. Kết thúc

- Khi người dùng chọn 1, nhập thông tin vào cho mảng 3 đối tượng car và mảng 3 đối tượng truck.
- Khi người dùng chọn 2, hiển thị toàn bộ thông tin vừa nhập là 3 car và 3 truck.
- Khi người dùng chọn 3, người dùng sẽ phải nhập vào 1 id, sau đó chương trình in ra một đối tượng có id đó nếu tìm thấy.
- Khi người dùng chọn 4, người dùng sẽ phải nhập vào 1 maker, sau đó chương trình in ra các đối tượng có maker đó nếu tìm thấy.
- Khi người dùng chọn 5, hiển thị danh sách các xe trước khi sắp xếp và sau khi sắp xếp theo giá.
- Khi người dùng chọn 6, hiển thị danh sách các xe trước khi sắp xếp và sau khi sắp xếp theo năm sản xuất.
- Khi người dùng chọn 7, chương trình kết thúc.

Hướng dẫn làm bài:

Đoạn mã của các giao diện và lớp như sau:

Giao diện IVehicle chứa 2 phương thức để nhập và xuất dữ liệu

```
interface IVehicle
    {
        void Input();
        void Output();
    }
```

Lớp Vehicle chứa 5 thuộc tính

```
class Vehicle : IVehicle, IComparable
    {
        public string id { get; set; }
        public string maker { get; set; }
        public string model { get; set; }
        public double price { get; set; }
        public int year { get; set; }
```

```
        public Vehicle()
        {

        }
        public Vehicle(string id)
        {
            this.id = id;
        }
        public Vehicle(string id, string maker, string model, double price, int
year)
        {
            this.id = id;
            this.maker = maker;
            this.model = model;
            this.price = price;
            this.year = year;
        }
        public virtual void Input()
        {
            Console.Write("Id: ");
            id = Console.ReadLine();
            Console.Write("Maker: ");
            maker = Console.ReadLine();
            Console.Write("Model: ");
            model = Console.ReadLine();
            Console.Write("Price: ");
            price = double.Parse(Console.ReadLine());
            Console.Write("Year: ");
            year = int.Parse(Console.ReadLine());
        }
        public virtual void Output()
        {
            Console.WriteLine("--------------------");
            Console.WriteLine("Id: " + id);
            Console.WriteLine("Maker: " + maker);
            Console.WriteLine("Model: " + model);
            Console.WriteLine("Price: " + price);
            Console.WriteLine("Year:" + year);
        }
        public override bool Equals(object obj)
        {
            Vehicle v = (Vehicle)obj;
            return (this.id.Equals(v.id));
        }
        public override string ToString()
        {
            var str = String.Format("{0,-8}{1,-10}{2,-25}{3,10}{4,10}",
                        id, maker, model, year, price);
            return str;
        }
        public int CompareTo(object obj)
        {
            Vehicle v = (Vehicle)obj;
            return (this.price.CompareTo(v.price));
        }
    }
```

```
    class CompareYear : IComparer<Vehicle>
    {
        public int Compare(Vehicle x, Vehicle y)
        {
            return (x.year - y.year);
        }
    }
```

Trong phương thức `ToString()`, lưu ý cách viết mã để định dạng dữ liệu in ra dạng cột cho các đối tượng Vehicle:

```
            var str = String.Format("{0,-8}{1,-10}{2,-25}{3,10}{4,10}",
                                    id, maker, model, year, price);
```

In ra 5 cột `id, maker, model, year, price` với số chỗ tương ứng dành ra để viết là 8, 10, 25, 10, 10. Dấu - ở phía trước là căn thẳng lề trái.

Lớp Car kế thừa từ lớp Vehicle.

```
class Car:Vehicle
    {
        public string color{ get; set; }
        public Car():base()
        {

        }
        public Car(string id, string maker, string model, double price,int
year,string color):base(id,maker,model,price,year)
        {
            this.color = color;
        }
        public override void Input()
        {
            base.Input();
            Console.Write("Color: ");
            color = Console.ReadLine();
        }
        public override void Output()
        {
            base.Output();
            Console.WriteLine("Color: "+color);
        }
```

Lớp Truck kế thừa từ lớp Vehicle

```
class Truck:Vehicle
    {
        public int truckload { get; set; }
        public Truck() : base()
        {

        }
        public Truck(string id, string maker, string model, double price, int
year, int truckload) : base(id,maker, model, price,year)
        {
            this.truckload = truckload;
```

```
        }
        public override void Input()
        {
            base.Input();
            Console.Write("Truckload: ");
            truckload = int.Parse(Console.ReadLine());
        }
        public override void Output()
        {
            base.Output();
            Console.WriteLine("Truckload: " + truckload);
        }
    }
```

Chúng ta thấy, do đặc điểm kế thừa, nên trong lớp `Car, Truck,` các đoạn mã chúng ta viết thêm rất ít. Đây chính là ưu điểm của việc sử dụng kế thừa.

Lớp Program, tạo menu cho người dùng chọn và thực hiện các chức năng:

1. Nhập dữ liệu
2. Hiển thị dữ liệu
3. Tìm kiếm theo id
4. Tìm kiếm theo maker
5. Sắp xếp theo theo price
6. Sắp xếp theo year
7. Kết thúc

```
class Program
    {
        static void Main(string[] args)        //main dang o Product.
        {
            int opt = 0;
            Console.OutputEncoding = System.Text.Encoding.UTF8;
            List<Vehicle> li = new List<Vehicle>();
            do
            {
                Console.WriteLine("============================");
                Console.WriteLine("1. Nhập dữ liệu");
                Console.WriteLine("2. Hiển thị dữ liệu");
                Console.WriteLine("3. Tìm kiếm theo id");
                Console.WriteLine("4. Tìm kiếm theo maker");
                Console.WriteLine("5. Sắp xếp theo price");
                Console.WriteLine("6. Sắp xếp theo năm sản xuất");
                Console.WriteLine("7. Kết thúc");
                Console.Write("Choice: ");
                opt = int.Parse(Console.ReadLine());
                switch (opt)
                {
                    case 1:
                        li.Add(new Car("veh1", "Mercedes", "BMW Serie 5",
2000, 2017, "blue"));
                        li.Add(new Car("veh2", "Mercedes", "Audi A6", 2500,
2019, "white"));
                        li.Add(new Car("veh3", "Honda", "Honda Civic", 1200,
2018, "gray"));
```

```
                    li.Add(new Truck("veh4", "Hyundai", "New Mighty N250",
2200, 2016, 3));
                    li.Add(new Truck("veh5", "Hyundai", "New Mighty 110s",
1500, 2015, 7));
                    li.Add(new Truck("veh6", "Isuzu", "QKR77FE4", 3000,
2017, 2));
                    break;
                case 2:

                    foreach (var item in li)
                    {
                        item.Output();
                    }
                    break;
                case 3:
                    Console.Write("Nhập id cần tìm kiếm:");
                    string id = Console.ReadLine();
                    Vehicle v = new Vehicle(id);
                    int index = li.IndexOf(v);
                    if (index != -1)
                    {
                        Console.WriteLine("Tìm thấy id: " + li[index]);
                    }
                    else
                    {
                        Console.WriteLine("--> Không có id " + id + "
trong danh sách");

                    }
                    break;

                case 4:
                    Console.Write("Nhập maker cần tìm kiếm:");
                    string m = Console.ReadLine();

                    List<Vehicle> li2 = new List<Vehicle>();

                    foreach (var item in li)
                    {
                        if (item.maker.Equals(m))
                        {
                            li2.Add(item);
                        }

                    }
                    if (li2.Count > 0)
                    {
                        Console.WriteLine("Tìm thấy maker: ");
                        foreach (var item in li2)
                        {
                            Console.WriteLine(item);
                        }
                    }
                    else
                    {
                        Console.WriteLine("---> Không có xe nào của "
+maker);
                    }
```

```
                    break;
                case 5:
                    Console.WriteLine("\n=== Danh sach truoc khi sap xep
===");
                    var header = String.Format("{0,-8}{1,-10}{2,-
25}{3,10}{4,10}",
                        "ID", "MAKER", "MODEL", "YEAR", "PRICE");
                    Console.WriteLine(header);

                    foreach (var item in li)
                    {
                        Console.WriteLine(item);
                    }
                    li.Sort();
                    Console.WriteLine("\n=== Danh sach sau khi sap xep
theo price ===");
                    foreach (var item in li)
                    {
                        Console.WriteLine(item);
                    }
                    break;
                case 6:
                    Console.WriteLine("\n=== Danh sach truoc khi sap xep
===");
                    var header2 = String.Format("{0,-8}{1,-10}{2,-
25}{3,10}{4,10}",
                        "ID", "MARKER", "MODEL", "YEAR", "PRICE");
                    Console.WriteLine(header2);

                    foreach (var item in li)
                    {
                        Console.WriteLine(item);
                    }

                    li.Sort(new CompareYear());
                    Console.WriteLine("\n=== Danh sach sau khi sap xep
theo year ===");
                    foreach (var item in li)
                    {
                        Console.WriteLine(item);
                    }
                    break;
                }
            } while (opt != 7);
        }
    }
```