Programming 3

UCN – Computer Science - C#

Interfaces and ArrayList

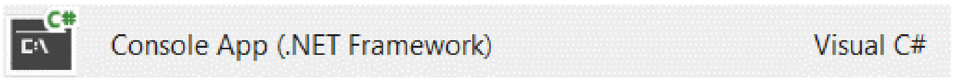
Topics:

- Interfaces
- Polymorphism
- ArrayList (non-generic list)
- Properties
- Enumeration types
- Type long

Content

First implementation

Create a console project. Project type:



Console App (.NET Framework) Visual C#

Create classes House And Boat in their own namespace called Investment or similar. Then insert the code below – modify *namespace* to your own *namespaces*.

Code for classes

House

```
using System;

namespace InterfacesAndArrayList.Investment {

    public class House {

        public string Address { get; set; }
        public int SquareMeters { get; set; }
        public int SquareMeterPrice { get; set; }
        public string Condition { get; set; }
        public string Description { get; set; }

        public House(string address, int squareMeters, int squareMeterPrice, string condition,
string description) {
            Address = address;
            SquareMeters = squareMeters;
            SquareMeterPrice = squareMeterPrice;
            Condition = condition;
            Description = description;
        }

        public int PriceCalculate() {
            int foundPrice = SquareMeters * SquareMeterPrice;
            return foundPrice;
        }

        public string InvestSummary() {
            string summ = "House: " + Address + Environment.NewLine + "Area = " + SquareMeters;
            return summ;
        }

    }
}
```

Boat

```
using System;

namespace InterfacesAndArrayList.Investment {

    public class Boat {

        public int Length{ get; set; }
        public int MeterPrice { get; set; }
        public string Condition { get; set; }
        public string Description { get; set; }
        public int RegYear { get; set; }

        public Boat(int length, int meterPrice, string condition, string description, int
regYear) {
            Length = length;
            MeterPrice = meterPrice;
            Condition = condition;
            Description = description;
            RegYear = regYear;
        }

        public int CalculatePrice() {
            int foundPrice = Length * MeterPrice;
            return foundPrice;
        }

        public string InvestSummary() {
            string summ = "Boat: " + Description + Environment.NewLine + "Length " + Length;
            return summ;
        }
    }
}
```

Main

Modify Main until you get a result as shown under Expected result.

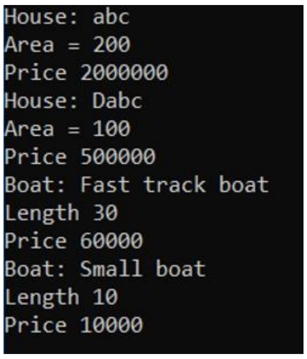
```
        static void Main(string[] args) {

            // Instantiate House and Boat objects
            ..

            // Add objects to an ArrayList
            ..

            int price = 0;
            string invSummary = null;
            // Loop through object and run either Boat or House methods
            // Use is opererator and Type Casting
            ..

            Console.ReadLine();
        }
```

Fill in the blanks – that is: *Replace .. with your own code.*

Compile and run!

Expected result

Output must be equivalent with:



```
House: abc
Area = 200
Price 2000000
House: Dabc
Area = 100
Price 500000
Boat: Fast track boat
Length 30
Price 60000
Boat: Small boat
Length 10
Price 10000
```

Improvement using interfaces

Problem

In the *Main* method you write very similar code for each investment item – and what if 5 new investment items were to be implemented.

Solution

You can improve code a lot by using an interface covering the calculation and summary methods. Therefore create an interface called *IInvestment* and define the methods *PriceCalculate* and *InvestSummary* inside the interface.

Next implement the interface in *House* and *Boat* classes.

Now having the interface implemented in *House* and *Boat* it´s time for impovements in the *Main* method. After the improvement the code inside the loop should have these properties:

* No type casting
* No use of is operator
* Approx. 4 lines of code incl. 2 for console write

Compile and run! Output should be as before.

More investment items

Try to implement more investment items besides houses and boats – they must implements *IInvestment*. Notice how easily you can handle the new items in *Main* method.

Enumeration types

Problem

In the *House* method the property Condition is a string. It means it´s possible to input all kinds of strange conditions and misspellings may occur as well.

Solution

Only allow strict conditions.

Therefore create an enum HouseCondition with these posssible conditions:

- Unknown
- Awful
- Bad
- Medium
- Good
- Perfect

Then use type HouseCondition instead of string - where it´s suitable.

Properties in work

Problem

You can input improper values for the numeric properties, e.g. negative boat length!

Solution

Enhance properties to check for improper values.

Ensure that boat length is greater than 0, otherwise set it to zero. Enhance property Length in Boat.
Do the equivalent with SquareMeters in House.
Possibly enhance more properties.

Run the program to test that improper values are dealt with.

Overflow

Problem

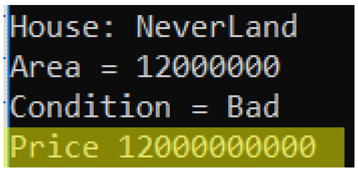
If you register a very large and expensive house you may get overflow using type *int*.

Try to create a House object for The Neverland Ranch (Set squareMeters to 12000000 and SquareMeterPrice to 1000) and watch the output!

Solution

Use type long when you calculate the price.

The output should look like this:



```
House: NeverLand
Area = 12000000
Condition = Bad
Price 12000000000
```

Alternative Solution

Another solution would be to test for overflow using the *checked* keyword. You may experiment with that