

Assignment (Day-3)

Rahul Kumar

Boxing

```
1 //Boxing convert an object type into value type.  
2 //Boxing is an implicit conversion process.  
3 using System;  
4  
5 namespace day3  
6 {  
7     0 references  
8     class AssignDay3  
9     {  
10        0 references  
11        public static void Main()  
12        {  
13            int val1 = 500;  
14  
15            //Boxing  
16            object obj1 = val1; //Boxing convert an object type into value type.  
17  
18            //change the value of val  
19            val1 = 2500;  
20  
21            Console.WriteLine(val1);  
22            Console.WriteLine(obj1);  
23        }  
24    }  
25 }  
26
```

Time Elapsed 00:00:01.23

C:\day3>dotnet run
2500
500

C:\day3>

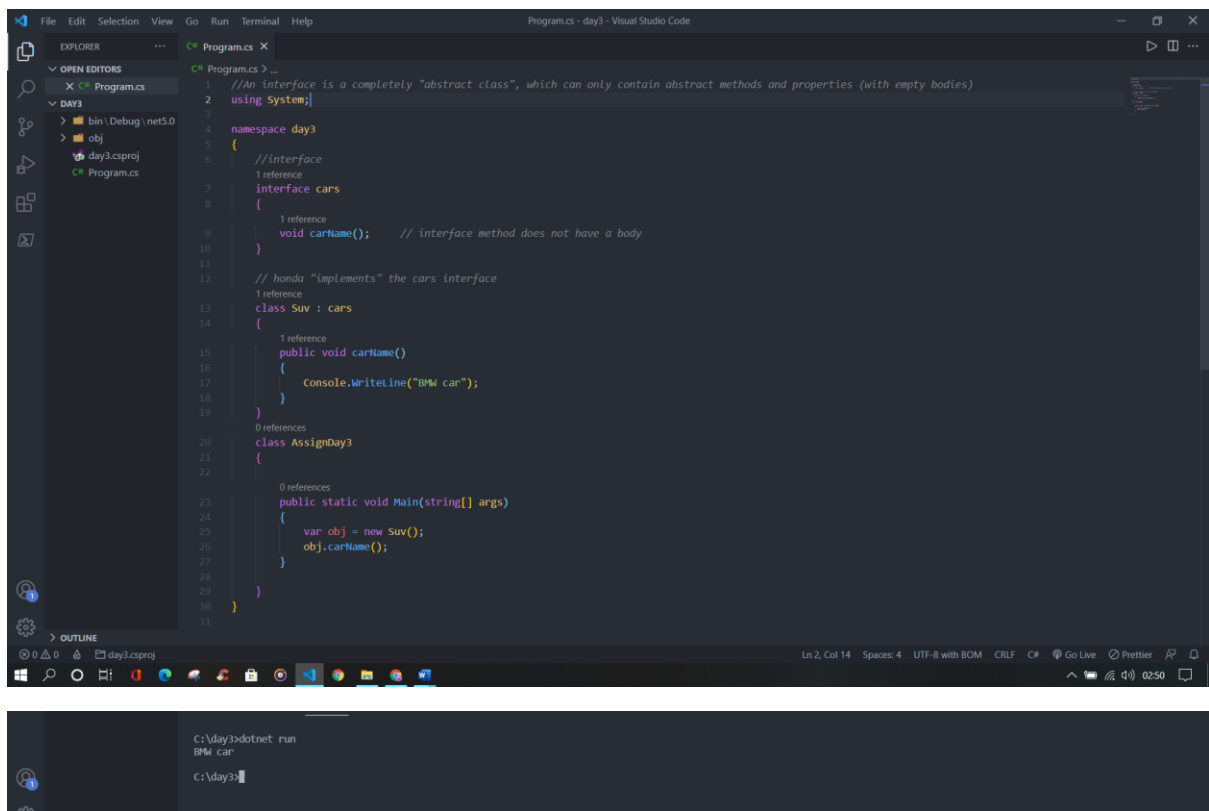
Unboxing

```
1 //unboxing convert an object type into value type.  
2 //unboxing is the explicit conversion process.  
3 using System;  
4  
5 namespace day3  
6 {  
7     0 references  
8     class AssignDay3  
9     {  
10        0 references  
11        public static void Main()  
12        {  
13            int val1 = 500;  
14  
15            //Boxing  
16            object obj1 = val1;  
17  
18            //Unboxing  
19            int x1 = (int)obj1;  
20  
21            Console.WriteLine(obj1);  
22            Console.WriteLine(x1);  
23        }  
24    }  
25 }  
26
```

C:\day3>dotnet run
500
500

C:\day3>

Interfaces



Abstract Class

```
using System;

namespace day3
{
    // declare class 'Area Class' as abstract
    abstract class areaClass
    {
        abstract public int Area(); // declare method area as abstract
    }

    class Square : areaClass
    {
        int side = 0;

        // constructor
        public Square(int n)
        {
            side = n;
        }

        // the abstract method, area is overridden here
        public override int Area()
        {

```

```

        return side*side;
    }

}

class AssignDay3
{

    public static void Main()
    {
        var sq= new Square(10);
        Console.WriteLine(sq.Area());
    }

}
}

```

The terminal window shows the following output:

```

build succeeded.
0 Warning(s)
0 Error(s)

Time Elapsed 00:00:01.20
C:\day3>dotnet run
100
C:\day3>

```

Value Type

The screenshot shows a C# program in Visual Studio Code. The code defines a class `AssignDay3` with two methods: `changeValue(int x)` and `Main()`. The `Main` method creates an integer `i` with the value 50, prints it, calls `changeValue(i)`, and prints it again. The `changeValue` method takes an integer `x` and sets it to 725, then prints it. The terminal output shows the sequence of values: 50, 725, and 50, demonstrating that the original value of `i` is not changed by the method call because `int` is a value type.

```

1 //when you pass a value-type variable from one method to another, the system creates a separate copy of a variable in another method. If value got c
2 using System;
3
4 namespace day3
5 {
6     class AssignDay3
7     {
8         1 reference
9         public static void changeValue(int x)
10        {
11            x= 725;
12            Console.WriteLine(x);
13        }
14
15     0 references
16     public static void Main()
17     {
18         int i = 50;
19         Console.WriteLine(i);
20         changeValue(i);
21         Console.WriteLine(i);
22     }
23 }
24
25
26
27

```

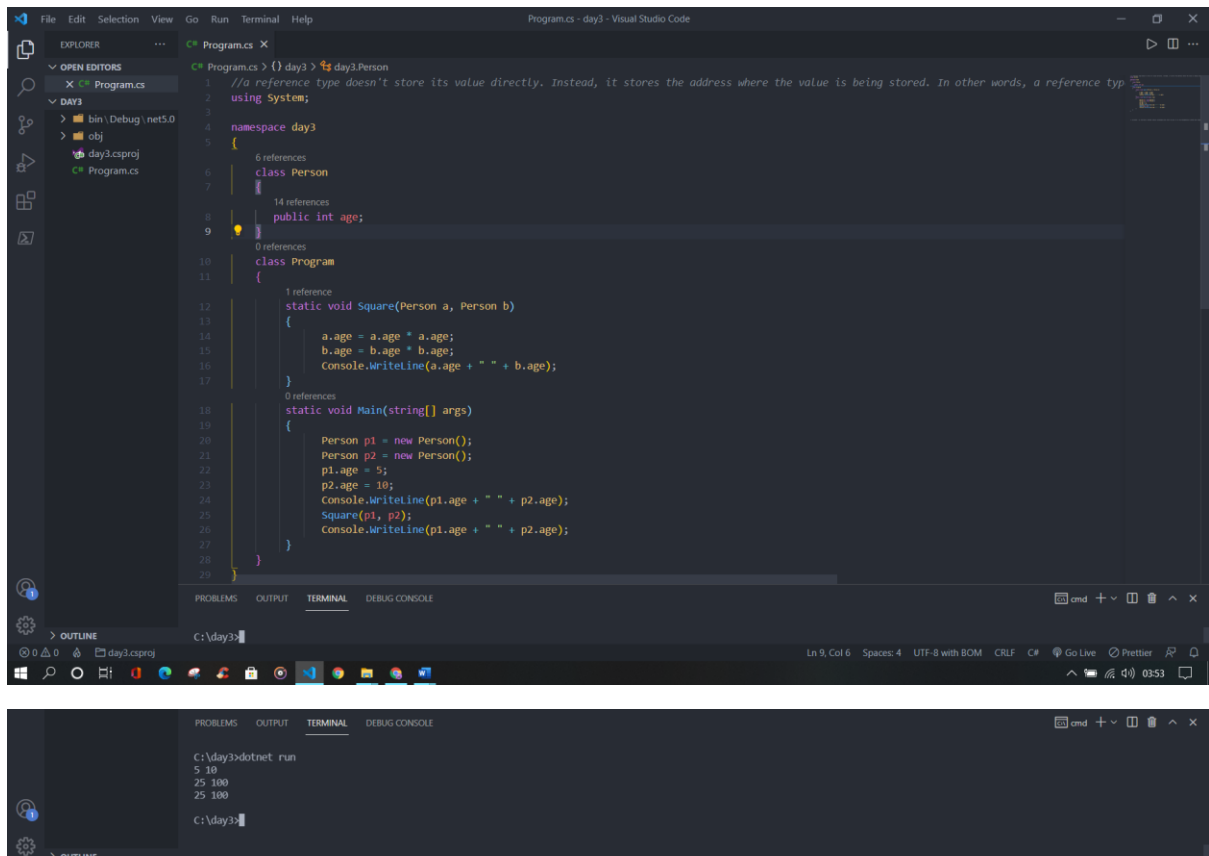
Terminal Output:

```

C:\day3>dotnet run
50
725
50
C:\day3>

```

Reference Type

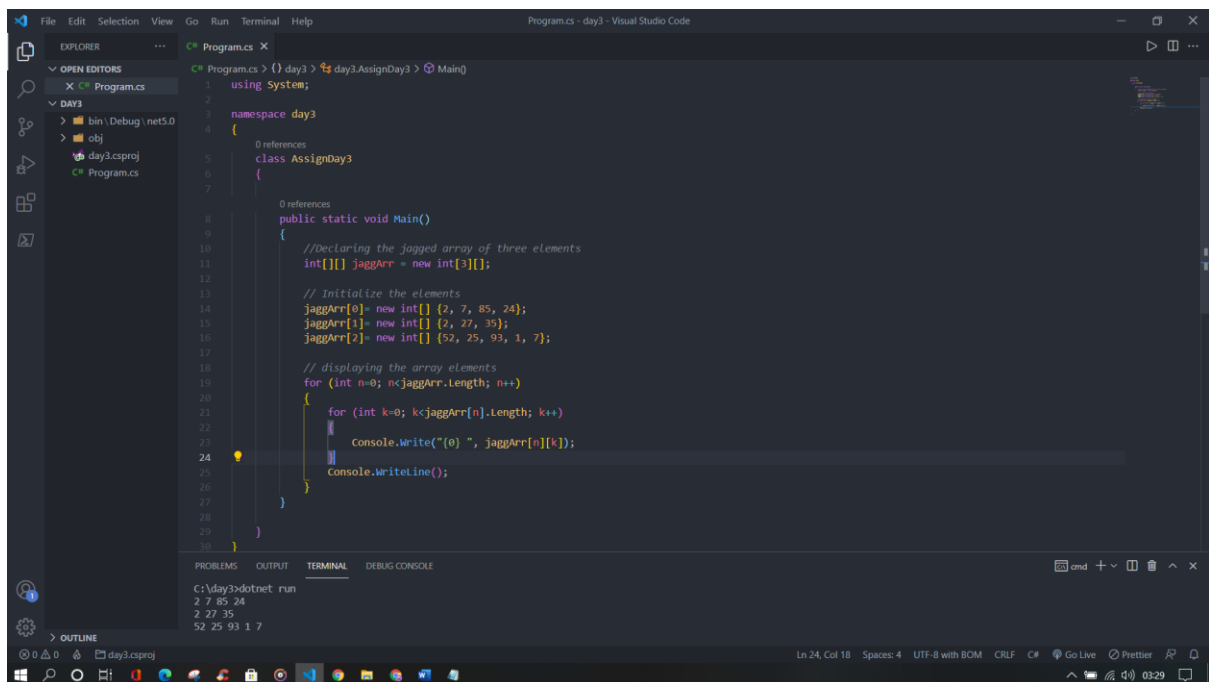


```
1 //A reference type doesn't store its value directly. Instead, it stores the address where the value is being stored. In other words, a reference type
2 using System;
3
4 namespace day3
5 {
6     class Person
7     {
8         public int age;
9     }
10
11     class Program
12     {
13         static void Square(Person a, Person b)
14         {
15             a.age = a.age * a.age;
16             b.age = b.age * b.age;
17             Console.WriteLine(a.age + " " + b.age);
18         }
19
20         static void Main(string[] args)
21         {
22             Person p1 = new Person();
23             Person p2 = new Person();
24             p1.age = 5;
25             p2.age = 10;
26             Console.WriteLine(p1.age + " " + p2.age);
27             Square(p1, p2);
28             Console.WriteLine(p1.age + " " + p2.age);
29         }
30     }
31 }
```

Terminal Output:

```
C:\day3>dotnet run
5 10
25 100
25 100
C:\day3>
```

Jagged Array

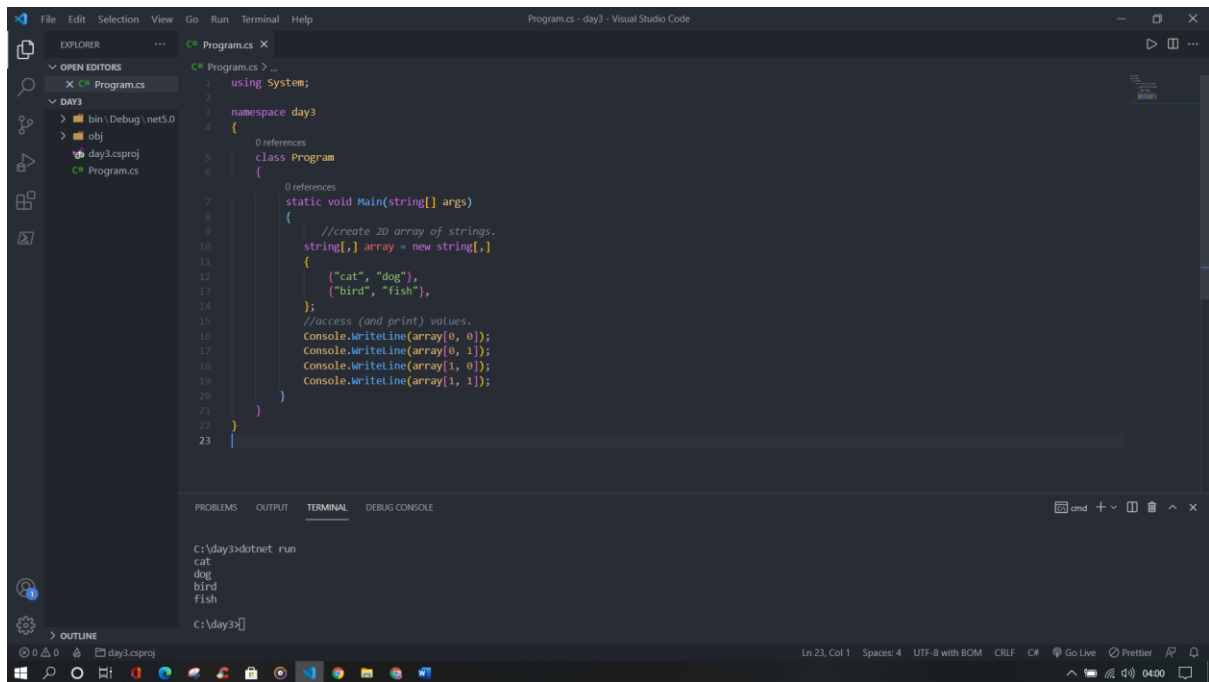


```
1 using System;
2
3 namespace day3
4 {
5     class AssignDay3
6     {
7
8         public static void Main()
9         {
10             //Declaring the jagged array of three elements
11             int[][] jaggedArr = new int[3][];
12
13             // Initialize the elements
14             jaggedArr[0] = new int[] { 2, 7, 85, 24 };
15             jaggedArr[1] = new int[] { 2, 27, 35 };
16             jaggedArr[2] = new int[] { 52, 25, 93, 1, 7 };
17
18             // displaying the array elements
19             for (int n=0; n<jaggedArr.Length; n++)
20             {
21                 for (int k=0; k<jaggedArr[n].Length; k++)
22                 {
23                     Console.Write("{0} ", jaggedArr[n][k]);
24                 }
25                 Console.WriteLine();
26             }
27         }
28     }
29 }
```

Terminal Output:

```
C:\day3>dotnet run
2 7 85 24
2 27 35
52 25 93 1 7
C:\day3>
```

2-D Array



```
File Edit Selection View Go Run Terminal Help
Program.cs - day3 - Visual Studio Code

EXPLORER
  OPEN EDITORS
    Program.cs
  DAY3
    bin \ Debug \ net5.0
    obj
    day3.csproj
    Program.cs

Program.cs
1  using System;
2
3  namespace day3
4  {
5      0 references
6      class Program
7      {
8          0 references
9          static void Main(string[] args)
10         {
11             //create 2D array of strings.
12             string[,] array = new string[,]{
13                 {"cat", "dog"},
14                 {"bird", "fish"},
15             };
16             //access (and print) values.
17             console.WriteLine(array[0, 0]);
18             console.WriteLine(array[0, 1]);
19             console.WriteLine(array[1, 0]);
20             console.WriteLine(array[1, 1]);
21         }
22     }
23

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
C:\day3>dotnet run
cat
dog
bird
fish
C:\day3>

OUTLINE
day3.csproj
Ln 23, Col 1 Spaces: 4 UTF-8 with BOM CRLF C# Go Live Prettier 04:00
```