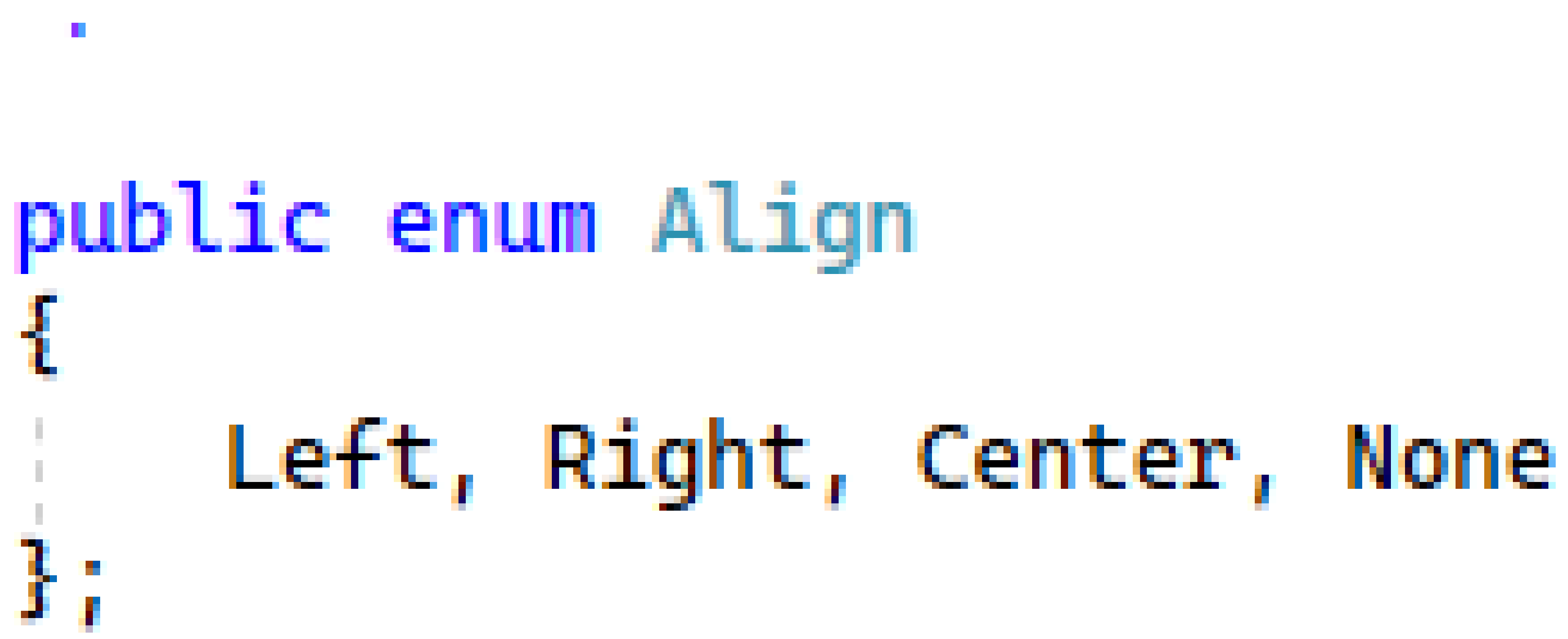
MIDTERM

Review

Code Structure

- Class Declaration
- Property
- Field
- Constructor
- Method
- Enum Declaration
- Public / Private
- Virtual / Override

Enum

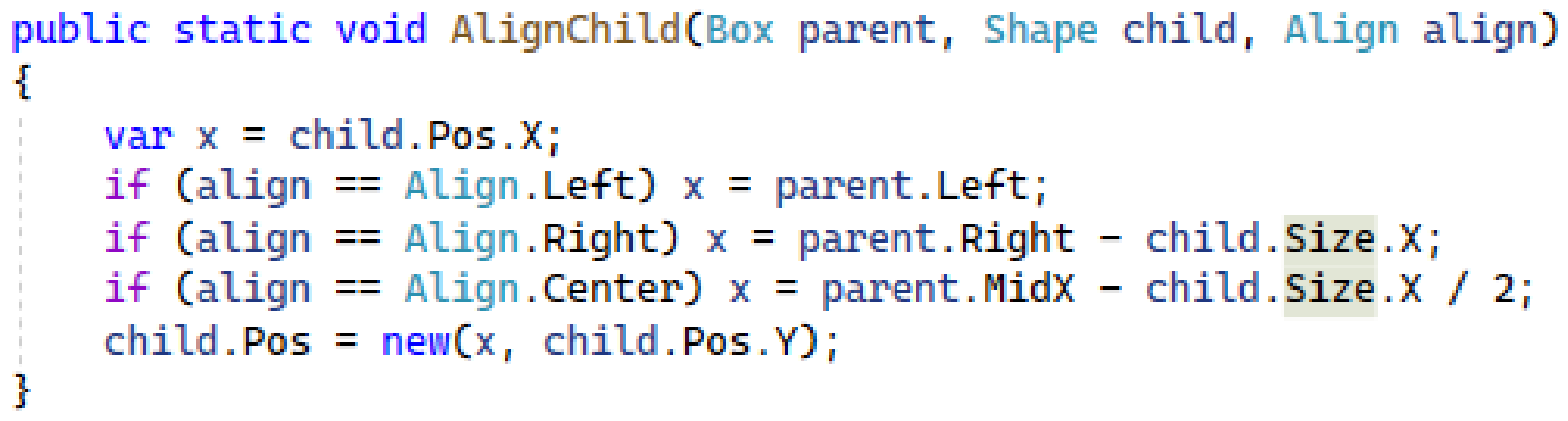


```
public enum Align
{
    Left, Right, Center, None
};
```

An Enum is a type

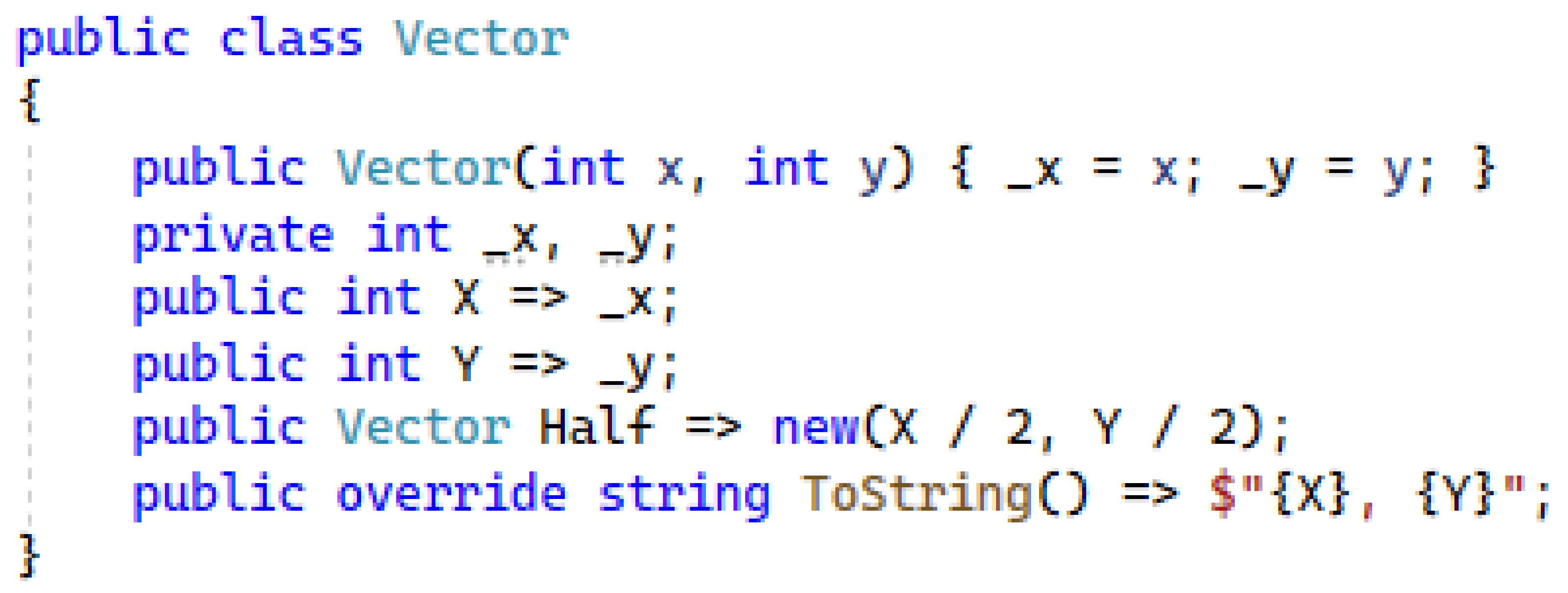
- Represents a small set of distinct values
- Each value is a constant
- Each value corresponds to a unique integer value and a unique name

How to use an Enum



```
public static void AlignChild(Box parent, Shape child, Align align)
{
    var x = child.Pos.X;
    if (align == Align.Left) x = parent.Left;
    if (align == Align.Right) x = parent.Right - child.Size.X;
    if (align == Align.Center) x = parent.MidX - child.Size.X / 2;
    child.Pos = new(x, child.Pos.Y);
}
```

Simple Class Declaration



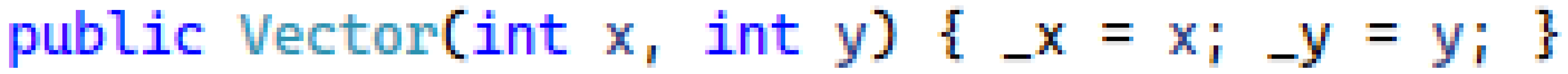
```
public class Vector
{
    public Vector(int x, int y) { _x = x; _y = y; }
    private int _x, _y;
    public int X => _x;
    public int Y => _y;
    public Vector Half => new(X / 2, Y / 2);
    public override string ToString() => $"{X}, {Y}";
}
```

Understanding Member Declarations

- Are there parentheses? If yes, then it is a method, otherwise, a field or property.
- Does it have an "=>"? If so, then it is a method or property.
- Does it have an "="? If so, then it is a field or property.
- Does it have a "get" or "set"? If so then it is a property.
- Is it a method with same name as class: then it is a constructor

Constructor: like a method

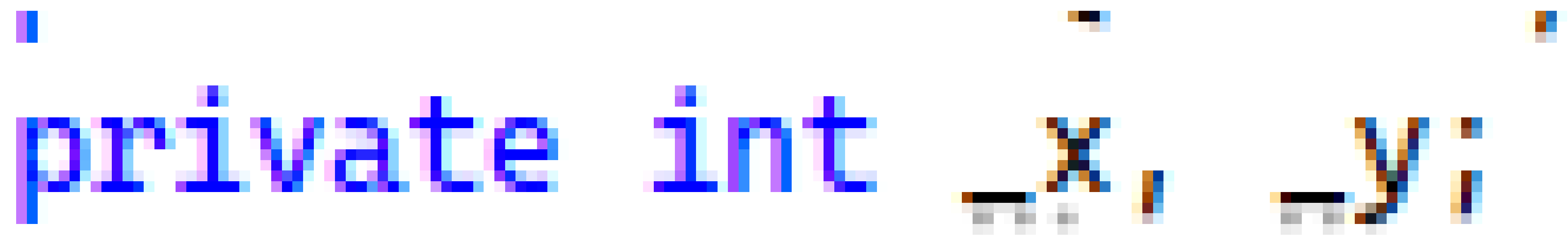
but no return type, and same name as class



```
public Vector(int x, int y) { _x = x; _y = y; }
```

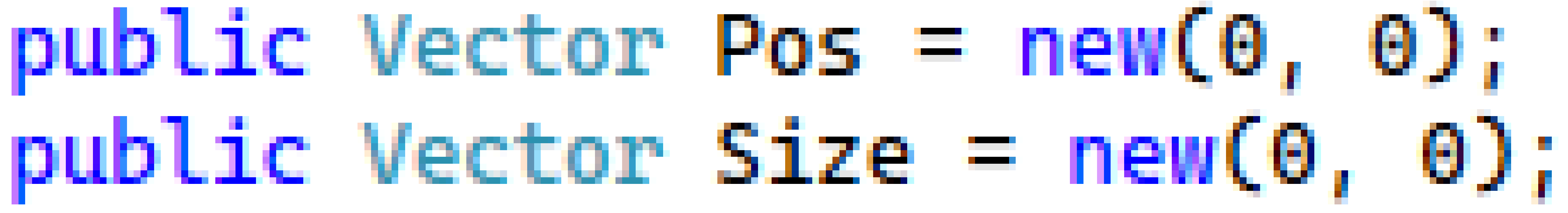
Private Fields

(should always start with _underscore)



```
private int _x, _y;
```

Public fields
(any field may be initialized in declaration)

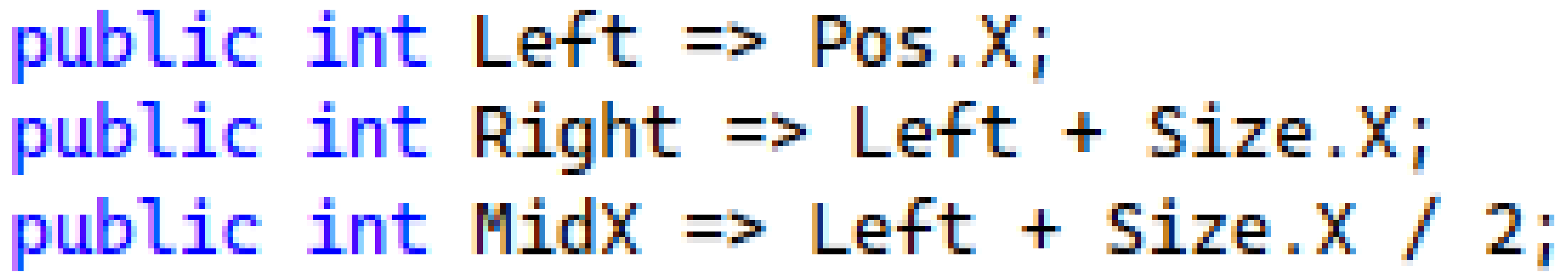


```
public Vector Pos = new(0, 0);
public Vector Size = new(0, 0);
```

These are Properties
(with auto-generated backing fields)

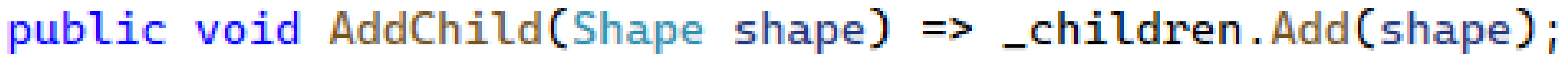
```
public Vector Pos { get; }
public int Radius { get; }
```

These are also Properties
(they are computed)



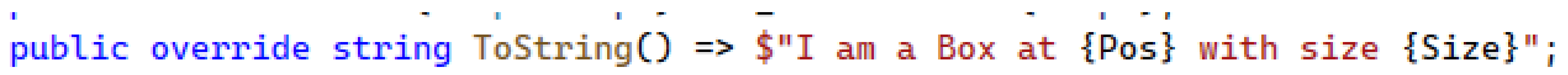
```
public int Left => Pos.X;
public int Right => Left + Size.X;
public int MidX => Left + Size.X / 2;
```

This a non-virtual method:



```
public void AddChild(Shape shape) => _children.Add(shape);
```

This is an override of a virtual function



```
public override string ToString() => $"I am a Box at {Pos} with size {Size}";
```

Mutability

- Fields can be changed after construction
- Includes inherited fields
- Includes private fields changed by functions
- Immutability means field values can never change after construction
- If there is a public field or property with public setter: automatically it is mutable
- If there are only private fields then depends on what functions do

Generic Types

- Types and Functions may have type arguments
- That means they are generic
- A List<T> is generic, an IEnumerable<T> is generic
- It requires a type argument
- Primitive types, other than array are not generic (int, string, float, etc.)

Miscellaneous Questions

- C# is a compiled language
- It is compiled into byte-code
- Assemblies (DLLs and EXEs) contain byte code
- The debugger executes compiled code
- No type of any value can ever change: you create new values.
- Everything has a type (including void and null)

Understanding Casts

- Explicit conversion/casts occur: when I
  - (T)x
  - This is a cast expression.
- Implicit conversion/casts occur: when I
  - pass a value to a function
  - assign it to a variable with different type
- https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/types/casting-and-type-conversions
- https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/type-testing-and-cast#cast-expression

Function Invocation

- Is the left hand of the "." an expression?
  - If so then it is an instance method
  - For example: 14.ToString()
  - For example: b.AddChild()
- Is the left hand of the "." a type?
  - If so then it is a static method
  - For example: Array.IndexOf
  - For example: Console.WriteLine
- Is it a lambda expression?
  - Then it is neither an instance or static method
  - For example: (x => x * 2)(12)