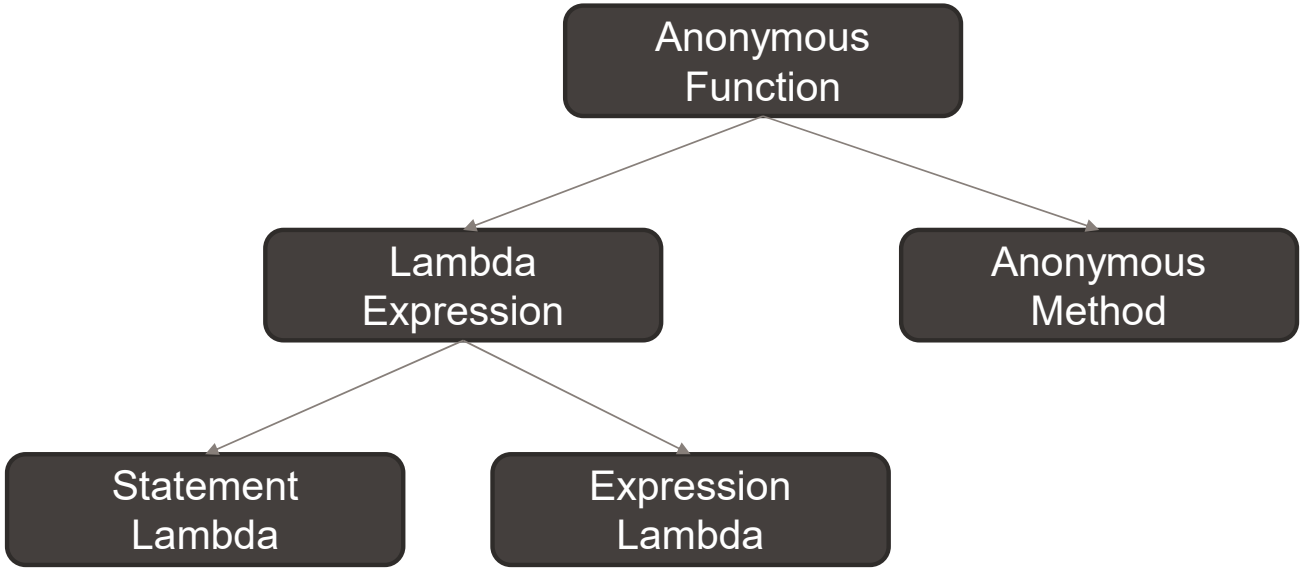
# Anonymous Function

- An anonymous function is an "inline" statement or expression that can be used wherever a *delegate type is expected*



# Anonymous Method

```
public delegate bool Mydelegate(int L, int M); //Declare Delegate

void bubbleSort(int[] arr, Mydelegate d) //Declare Method
{ ...}
```

```
bubbleSort( arr,
            delegate ( int A, int B )
                    {
                    return A > B ;
                    }
);
```

Anonymous Method

# Anonymous Method

- anonymous method could access outer method variables

```
void method1()
        {
            string str ="Hello";
            method2(delegate (string s)
              {
                Console.writline(str +" " +s);
              }
            );
        }
void mehod2(Action<string> action)
        {
            action("world");
        }
```

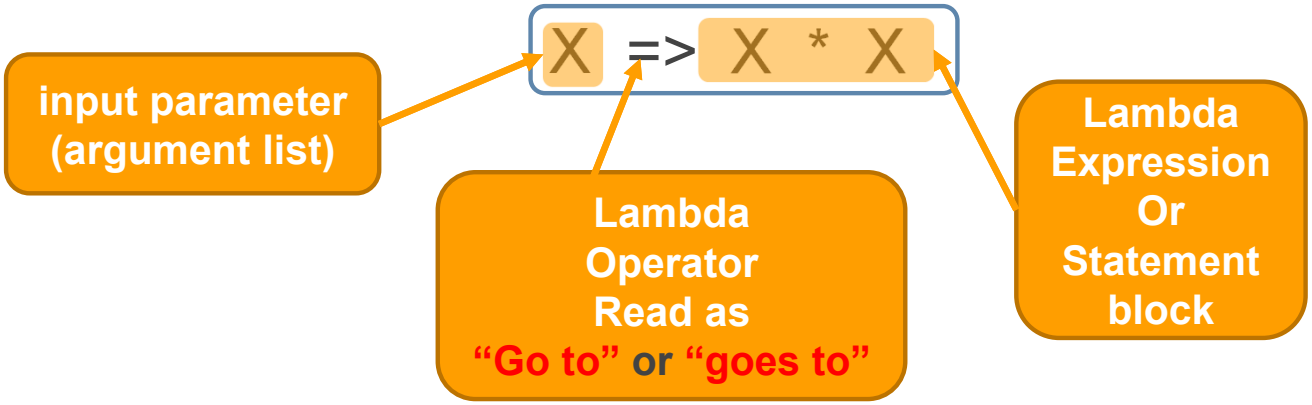
## Anonymous Method

- It cannot contain jump statement like *goto* , *break* or *continue*.
- It cannot access *ref* or *out* parameter of an outer method.
- It cannot have or access unsafe code.
- It cannot be used on the left side of the is operator.

# Lambda Expressions

- Code could be represented as Expression that compiled to delegate
- Delegate Types
  - Action delegate →no return
  - Func delegate → return
- Lambda expression anatomy

```
Func<int, int> square = x => x * x;
Console.WriteLine(square(5)); // 25
```



# Lambda Expression

## Statement Lambda

```
bubbleSort(arr,
        (A,B) =>
        {
          return A > B;
        }
);
```

- One parameter

```
A=>
```

- Parameterles

```
()=>
```

# Lambda Expression

- Expression Lambda

```
bubbleSort( arr, (a, b) => a < b );
```

## Expression-bodied members C# 6 -7

- Could be used whenever the logic of the member Consist of single-line statements
- This member could be
  - Method
  - Property

```
member => expression;
```

# Expression-bodied members C# 6 -7

- Method

```
class Person
{
    private string fname;
    private string lname;
    public Person(string firstName, string lastName)
    {
        fname = firstName;
        lname = lastName;
    }
    public override string ToString() => $"First Name:{fname} \nLastName:{lname}";
    public void Display() => Console.WriteLine(ToString());
}
```

# Expression-bodied members C# 6 -7

- Property
  - Read only Property (get only)

```
PropertyType PropertyName => expression;
```

```
□řụčľîç□çľắșș□Ľộçắƫîộŋ
□□□□
□□□□□□řsîŵắƫêđ□ștsîŋĝ□ľộçắƫîộŋŅắṇê□
□□□ □ □
 □ □□□řụčľîç□ștsîŋĝ□Ņắṇê□□□□ľộçắƫîộŋŅắṇê□
□□□□□
```

# Expression-bodied members C# 6 -7

- Property
  - set, get

```
řụčľîç□çľắșș□Ľộçắʧîộŋ
□
□□□řsîŵắʧê□ștsîŋĝ□ľộçắʧîộŋṆắṇê□
□□ □□
 □ řụčľîç□ștsîŋĝ□Ṇắṇê
□□□□□□□□
□□□□□□□□□□□□ĝêʧ□□□□ľộçắʧîộŋṆắṇê□
□□□□□□□□□□□□ṣêʧ□□□□ľộçắʧîộŋṆắṇê□□□ŵắľụê□
□□□□□□□□
□
```

# Implicit Typed Local Variable var

- *var* keyword
  - Used for declare a variable its data type would be detected at the run-time by the compiler
    - Variable initialization must be on the declaration

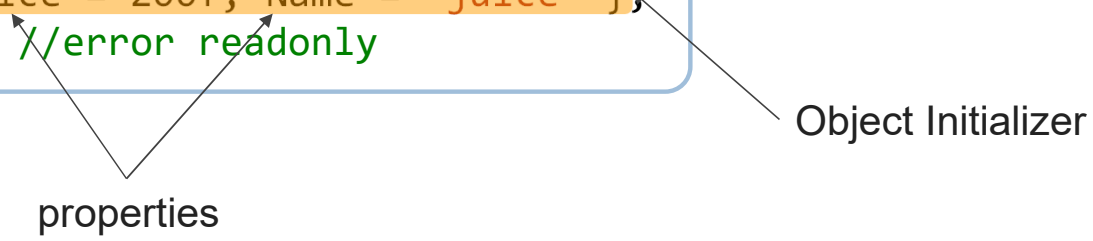
```
employee em = new employee { ID = 10, Name = "Ahmed", Salary = 1000f };
em.ID = 2;
```

```
var v = new employee { ID = 10, Name = "Ahmed", Salary = 1000f };
v.ID = 2;
```

# Anonymous Type

- Anonymous type is a simple class that is created by the compiler on the fly to store a set of values
- Declare an object of anonymous data type done by using *new* keyword followed by object initializer
- Var keyword must be used to reference anonymous Type since it doesn't have a type name

```
var v2 = new { Price = 200f, Name = "juice" };
v2.name = "milk"; //error readonly
```



# Anonymous Type

- Anonymous type overrides  methods
  - ToString()
  - Equals()
- Mainly used in LINQ

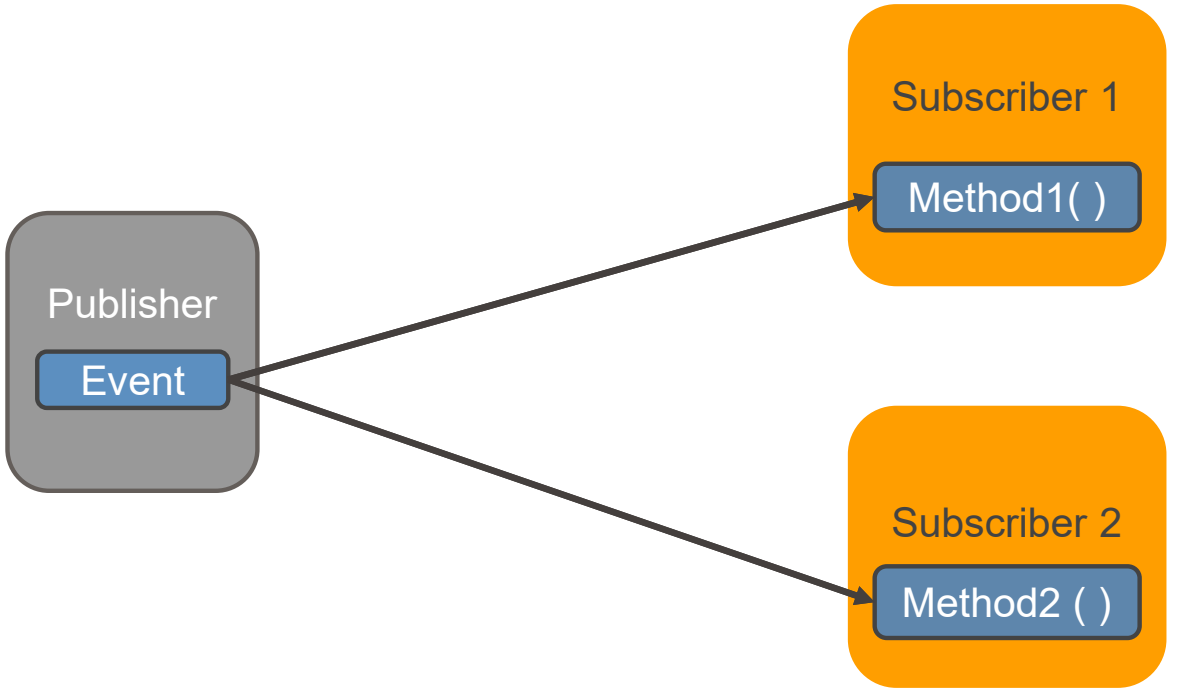
# Events

- Event is a ***special*** multicast Delegate variable
- Event Declaration

```
public delegate void mydelegate(int x); // declare type (delegate)
…
public event mydelegate d; // declare event
```

# Events

- Publisher Subscriber Design Pattern
  - Event
  - Event Handler

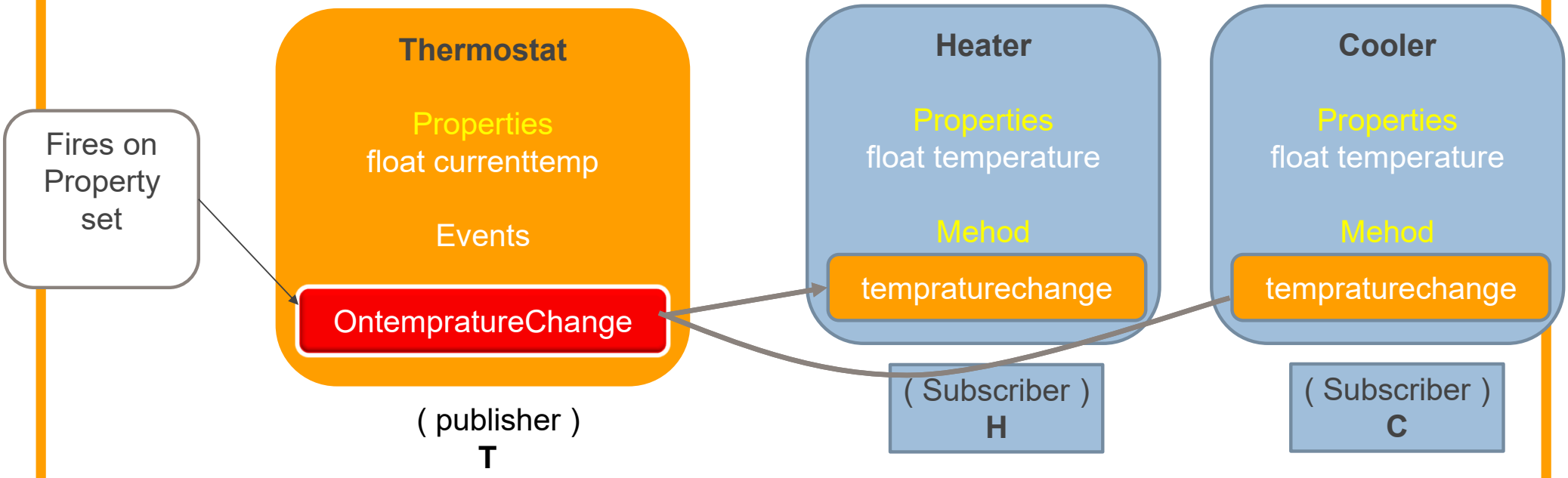


# Why event?

- Prevent errors or bugs results from
  - Cancelling other subscriber by (encapsulation subscriber )
    - Using Assignment  +=  instead  =
    - Only subscriber can cancel its subscription
  - Only Publisher can invoke (call) event

# Assignment

- Write a program for heater , cooler and thermostat




```
T.OntempratureChange += C.tempraturechange;
T.OntempratureChange += H.tempraturechange;
```