

# Operator Overload

Peter Levinsky IT Roskilde

25.10.2024

# Overload of operators

- +, -, \*, / ... Operator
- Indexer (i.e. [] )
- <= and >= Operator
- == and != Operator
- Convert types ~> Cast

# Overload: +, -, \*, /, ++ and -- Operator

- Example:

```
public static Time operator +(Time tA, Time tB)
{
    ... // this add two time objects and return a new
}
```

Call:

```
Time timeC = timeA + timeB;
```

The general definition

```
public static TReturn operator operatorGoesHere(TA opA, TB opB)
{
    ...
}
```

# Overload: +, -, \*, /, ++ and -- Operator

- Example variation:

```
public static Time operator *(int val, Time t)
{
    int totalMin = (t.Hours * 60 + t.Minutes) * val;
    return new Time(totalMin / 60, totalMin % 60);
}
```

Call:

```
Time timeMult3A = 3 * timeA; // OK
```

but

```
Time timeMultA3 = timeA * 3; // Error
```

```
public static Time operator *(Time t, int val) // reverse parameter order
{
    return val * t;
}
```

# Overload - Indexer (i.e. [] )

```
public class NPC
{
    private Dictionary<NPCStateTypes, int> State;

    public NPC()
    {
        State = new Dictionary<NPCStateTypes, int>();
        SetDefaultValues();
    }

    public void SetStateValue(NPCStateTypes stateType, int
value)
    {
        State[stateType] = value;
    }

    public int GetStateValue(NPCStateTypes stateType)
    {
        return State[stateType];
    }

    private void SetDefaultValues() { ... }
}
```

```
public enum NPCStateTypes
{
    hungry, rested, aggressive, fear, gullible
}
```





**Wish to use [] Instead of  
GetStateValue,  
SetStateValue**

# Overload - Indexer (i.e. [] )

- The implementation in the class:

meaning this class

define the brackets + what's inside

```
public int this[NPCStateTypes stateType]  
{  
    get { return State[stateType]; }  
    set { State[stateType] = value; }  
}
```

# Overload == and != Operator

If overload == also overload != 😊

```
public static bool operator ==(Time tA, Time tB)
{
    return (tA.Length == tB.Length); // tA.Equals(tB)
}
```

```
public static bool operator !=(Time tA, Time tB)
{
    return !(tA == tB);
}
```

BUT this implementations do not work – risk for `NullReferenceException`! DO THIS!

1. First override `Equals` appropriately (i.e. also checking for null)
2. As a consequence of overriding `Equals`, we must also override the method `GetHashCode` from `Object`.
3. Now override `==` and `!=` by using `Equals`

# Overload: <= and >= Operator ( < and > )

If overload >= also overload <= ☺

```
public static bool operator >=(Time tA, Time tB)
{ // ToDo check for null
    return (tA.Length >= tB.Length);
}
```

```
public static bool operator <=(Time tA, Time tB)
{
    return (tA.Length <= tB.Length);
}
```



# Overload: Casting operator

## Implicit:

```
public static implicit operator byte(Digit d) => d.digit;
```

## Use:

```
var d = new Digit(7);  
byte number = d;
```

## Explicit:

```
public static explicit operator Digit(byte b) => new Digit(b);
```

## Use:

```
Digit digit = (Digit)number;
```

# That's it

- Training: OOP.4.1, OOP.4.2

- **And the Mandatory Assignment**

