Operator overloading

Operator Overloading

- You can overload operators in order to use clear syntax for some operations with user-defined classes.
- Which operators can be overloaded?

operators	Description
+, -, !, ~, ++, --	These unary operators can be overloaded.
+, -, *, /, %	These binary operators can be overloaded.
==, !=, <, >, <=, >=	The comparison operators can be overloaded; always in pairs: == with !=, < with >, <= with >=. If you overload < and > as well as >= and <= you should implement IComparable and IComparable<T> interfaces.
&&, ||	The conditional logical operators cannot be overloaded directly. You have to overload & and I, and && and II are derived.
+=, -=, *=, /=, %=	The assignment operators cannot be overloaded. You have to overload +, -, *, / ,%. The others are derived.
=, ., ?:, ->, new, is, sizeof, typeof	These operators cannot be overloaded.

Operator Overloading

- To overload operator a function has to be defined, which:
  - Has „**operator**” **keyword** and the operator symbol in the name (e.g. „operator +”)
  - Is **public** and **static**
  - At least one argument has to be of the same type as the class in which the operator is defined.

Example

Function below implements the addition operator (+) for a user-defined class Box. It adds the attributes of two Box objects and returns the resultant Box object.

```
public static Box operator+ (Box b, Box c) {
   Box box = new Box();
   box.length = b.length + c.length;
   box.breadth = b.breadth + c.breadth;
   box.height = b.height + c.height;
   return box;
}
```

Function below implements the comparison operator (==) for a user-defined class Box.

```
public static bool operator == (Box lhs, Box rhs) {
      bool status = false;
      if (lhs.length == rhs.length && lhs.height == rhs.height && lhs.breadth == rhs.breadth) {
         status = true;
      }
      return status;
   }
```

https://www.tutorialspoint.com/csharp/csharp_operator_overloading.htm

Example: comparison operators

The following is the appropriate standard implementation of the comparison operators in C# for a type UDT:

```
public int  CompareTo(UDT x) { return CompareTo(this, x); }
public bool Equals(UDT x)    { return CompareTo(this, x) == 0; }
public static bool operator  < (UDT x, UDT y) { return CompareTo(x, y)  < 0; }
public static bool operator  > (UDT x, UDT y) { return CompareTo(x, y)  > 0; }
public static bool operator <= (UDT x, UDT y) { return CompareTo(x, y) <= 0; }
public static bool operator >= (UDT x, UDT y) { return CompareTo(x, y) >= 0; }
public static bool operator == (UDT x, UDT y) { return CompareTo(x, y) == 0; }
public static bool operator != (UDT x, UDT y) { return CompareTo(x, y) != 0; }
public override bool Equals(object obj)
{
    return (obj is UDT) && (CompareTo(this, (UDT)obj) == 0);
}
```

Just add the custom definition for private static int CompareTo(UDT x, UDT y)

https://stackoverflow.com/questions/20593755/why-do-i-have-to-overload-operators-when-implementing-compareto

Task

Class below represents a complex number. Write a code that allows to use following operators for this class: **addition** of complex numbers, addition of real value to the complex number; **comparison operators > and <.**

```
class Cmplx
   {
      double Re { set; get; }
      double Im { set; get; }

   }
```


WYDZIAŁ FIZYKI
POLITECHNIKA WARSZAWSKA
POLITECHNIKA WARSZAWSKA
WARSAW UNIVERSITY OF TECHNOLOGY

Attributes

Attributes

Attributes are **piece of information**.

This information can be attached to your method, class, namespace, assembly etc.

```
[Obsolete]
public void Add(Cmplx b, Cmplx c)
{
    this.Re = b.Re + c.Re;
    this.Im = b.Im + c.Im;

}
```

Attributes are part of your code this makes developers life easier as he can see the information right upfront in the code while he is calling the method or accessing the class and take actions accordingly.

THE END

dr inż. Małgorzata Janik

majanik@if.pw.edu.pl

Winter Semester 2017/2018