THM **Uebung 03**

Komponenten & Frameworks

Prof. Dr. Henrich

Ziele: Operatorüberladung, Interface, Klonen, Indexer

Aufgabe 1
Modellieren Sie komplexe Zahlen (Klasse compl) so, dass folgende Main-Funktion die gewünschte Ausgabe erzeugt.

```
 static void Main(string[] args)
 {
  double d = 8;
  compl c1 = new compl(3);      // 3 + 0j
  compl c2 = new compl(13,14);  // 13 + 14j
  Console.WriteLine(c1+c2);       //    // Ausgabe: Realteil: 16   Imaginärteil: 14
  Console.WriteLine(c2+d) ;       //    // Ausgabe: Realteil: 21   Imaginärteil: 14
  Console.ReadLine();
 }
// Hinweis: Komplexe Zahlen bestehen aus 2 Teilen, einem Realteil und einem Imaginärteil (=Zahl*j)
//              Bsp.: 13 + 14j
//              Addition erfolgt getrennt nach Real- bzw. Imaginärteil:  (3 + 5j) + (7 + 2j) = (10 + 7j)
```

Aufgabe 2
Gegeben ist eine Klasse Strecke in der x-y-Ebene, sowie die Main-Funktion.
Vervollständigen Sie das Programm.
a) Verwenden Sie class Punkt
b) Verwenden Sie struct Punkt

```
class Strecke
 {
    public Punkt anfang;
    public Punkt ende;
    public Strecke (Punkt a, Punkt e){anfang = a; ende = e;}
    public double laenge()
    {
        // TODO  Berechnet die Laenge der Strecke
    }
}

class T
{
    [STAThread]
    static void Main(string[] args)
    {
     Strecke g = new Strecke(new Punkt(2,3),new Punkt(-1,-1));
     ICloneable gklone = (ICloneable) g.Clone();    // Kopie erstellen
     Console.WriteLine(g.laenge());     // Ausgabe 5
```

```
      g.anfang.X=3000;
      g.anfang.Y=3000;
      Console.WriteLine(((Strecke)gklone).laenge()); // Ausgabe 5
      Console.ReadLine();
     }
 }
```

Aufgabe 3

Erstellen Sie eine Klasse Matrix, zu folgender Main-Fkt.:

```
Matrix m1 = new Matrix(2,3);
m1.belegen();
m1[1,2] = 7;
Console.WriteLine(m1[0,0]);
m1.show();
```

Die Methode belegen() initialisiert die Matrixelemente aufsteigend beginnend mit 100 und show() gibt die Matrix zeilenweise aus.

Bsp.: 2x3 Matrix

100 101 102

103 104 105