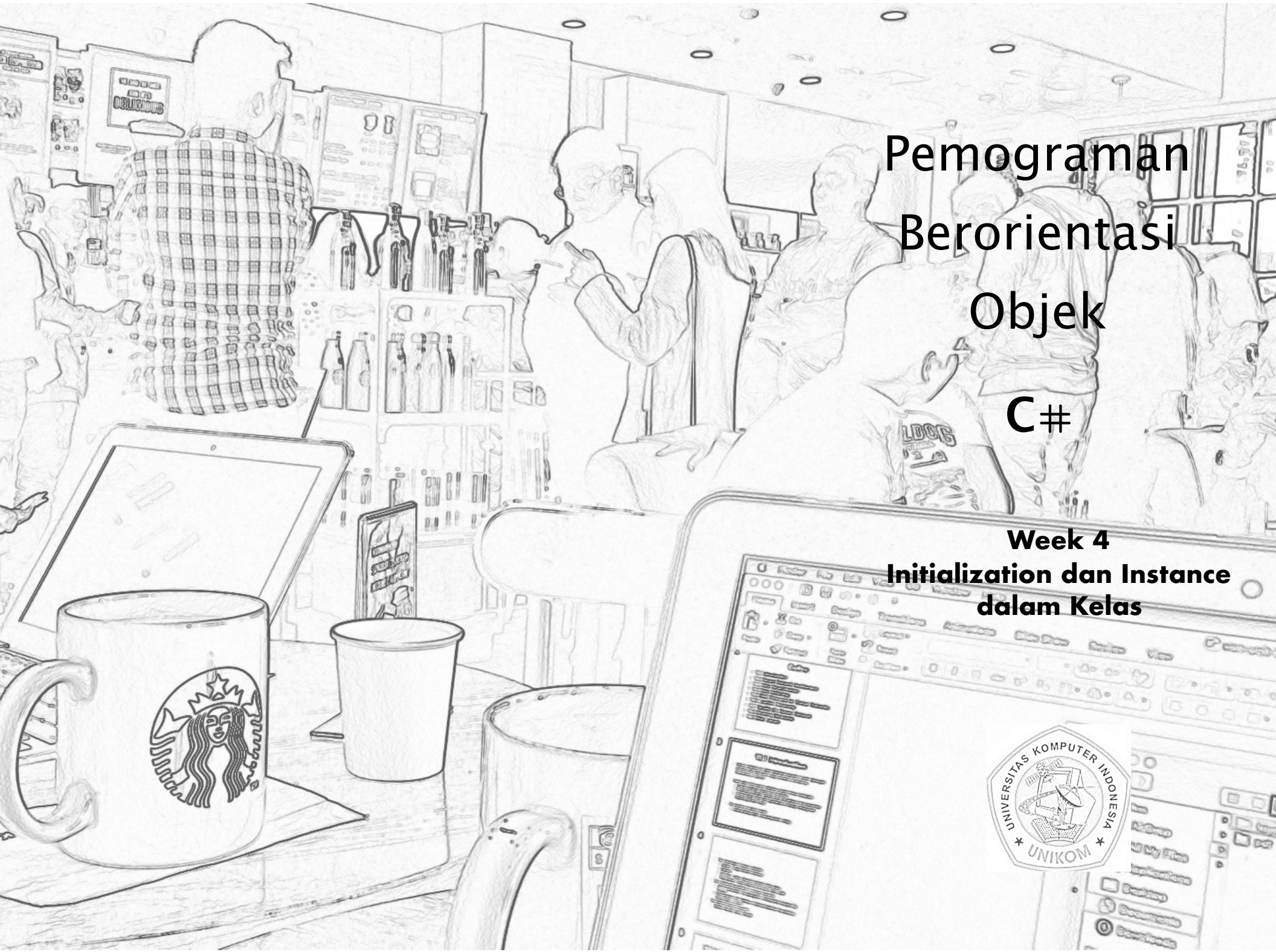


Pemograman Berorientasi Objek

C#

Week 4
Initialization dan Instance dalam Kelas

Constructor

- **Konstruktor** adalah method yang pertama kali dikerjakan ketika suatu kelas (class) diciptakan
- Bertugas untuk melakukan proses inisialisasi
- **Nama Konstruktor = nama class**
- Konstructor harus dideklarasi sebagai **public**
- Dapat memiliki parameter, tetapi tidak mengembalikan nilai (biarpun void)
- Tiap class harus mempunyai minimal 1 (satu) konstruktor, bila tidak dideklarasikan oleh user, maka kompiler akan secara otomatis membuat default konstruktor

Macam-macam Konstructor

1. ***Default Constructor***

   Konstructor yang digunakan untuk inisialisasi dan didefinisikan tanpa argumen (parameter)

2. ***Constructor dengan parameter***

   Konstructor yang digunakan untuk inisialisasi obyek (object) dan didefinisikan dengan argumen (parameter)

3. ***Copy Constructor***

   Konstruktor khusus yang digunakan untuk meng-copy isi dari suatu object ke dalam obyek baru yang sedang diciptakan

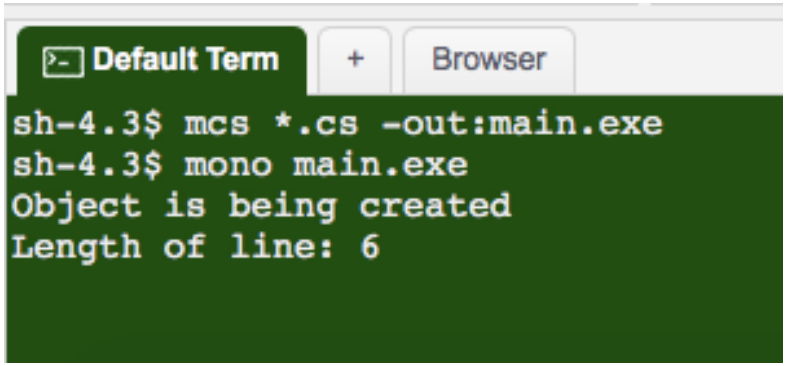
Default Constructor

```
using System;

namespace LineApplication
{
    class Line
    {
        private double length; //length of a line
        public Line()          //Default constructor
        {
        Console.WriteLine("Object is being created");
    }

    public void setLength(double len)
    {
        length = len;
    }
    public double getLength()
    {
        return length;
    }

    static void Main(string[] args)
    {
    Line line=new Line();
    //set line lentgth
    line.setLength(6.0);
    Console.WriteLine("Length of line: {0}", line.getLength());
    Console.ReadKey();
    }
    }
}
```



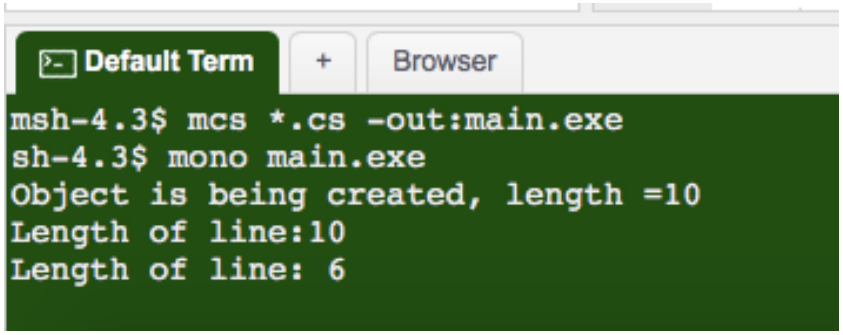
Parameterized Constructor

```
using System;

namespace LineApplication
{
    class Line
    {
        private double length; //length of a line
        public Line(double len) //Parameterized constructor
        {
        Console.WriteLine("Object is being created, length ={0}", len);
        length=len;
    }

    public void setLength(double len)
    {
        length = len;
    }
    public double getLength()
    {
        return length;
    }

   static void Main(string[] args)
       {
       Line line=new Line(10.0);
       Console.WriteLine("Length of line:{0}", line.getLength());
       //set line lentgth
       line.setLength(6.0);
       Console.WriteLine("Length of line: {0}", line.getLength());
       Console.ReadKey();
       }
    }
}
```

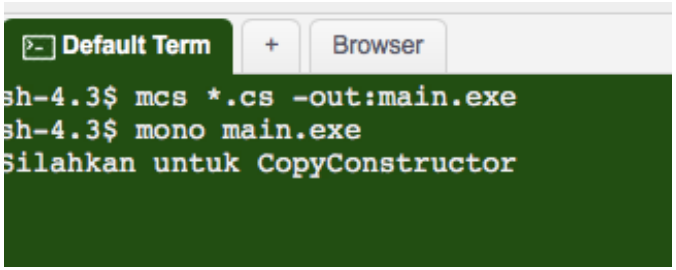


Copy Constructor

```
using System;

namespace CopyConstructor
{
   class Sample
   {
      public string param1, param2;
      public Sample(string x, string y)
      {
         param1 = x;
         param2 = y;
      }
      public Sample(Sample obj)     // Copy Constructor
      {
         param1 = obj.param1;
         param2 = obj.param2;
      }
}

class Program
   {
      static void Main(string[] args)
      {
         Sample obj = new Sample("Silahkan", "CopyConstructor");  // Create instance to class Sample
         Sample obj1=new Sample(obj); // Here obj details will copied to obj1
         Console.WriteLine(obj1.param1 +" untuk " + obj1.param2);
         Console.ReadLine();
      }
   }
}
```



Destructor

- Destruktor adalah method yang terakhir kali dikerjakan sebelum siklus hidup object berakhir
- Bertugas melakukan proses deinisialisasi, clean up, dealokasi memory, dan lain-lain
- Seperti konstruktor nama destruktor sama dengan nama class dengan diberi tanda **~** di depan nama destruktor
- **Nama constructor = ~nama class**
- Didefinisikan tanpa return type dan tanpa argument (parameter).
- Destruktor dideklarasi sebagai **public**

Destructor

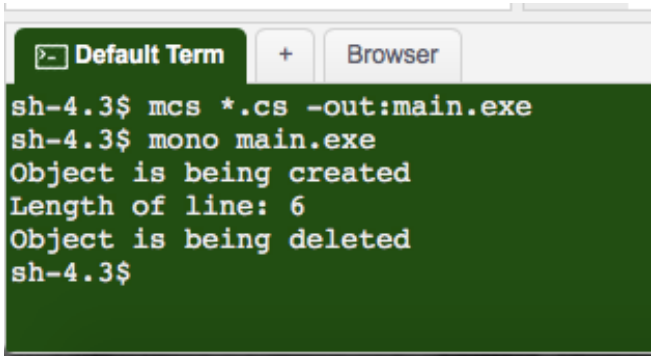
```
using System;

namespace LineApplication
{
    class Line
    {
        private double length; //length of a line
        public Line() //Constructor
        {
           Console.WriteLine("Object is being created");
        }

    ~Line() //destructor
       {
        Console.WriteLine("Object is being deleted");
       }

    public void setLength(double len)
       {
        length = len;
       }
    public double getLength()
       {
        return length;
       }

    static void Main(string[] args)
       {
          Line line=new Line();
          //set line lentgth
          line.setLength(6.0);
          Console.WriteLine("Length of line: {0}", line.getLength());
       }
    }
}
```



Pemanggilan Constructor dan Destructor

- Konstruktor & Destruktor dipanggil secara otomatis.
- Konstruktor pada object global dipanggil sebelum fungsi main dieksekusi dan destruktor dikerjakan setelah fungsi main selesai dieksekusi.
- Konstruktor pada object lokal dipanggil otomatis saat object dedefinisikan dan destructornya dieksekusi setelah proses object selesai.
- Pada object **static** lokal, konstruktor dipanggil sekali pada saat object didefinisikan dan destruktornya dipanggil setelah fungsi main selesai.

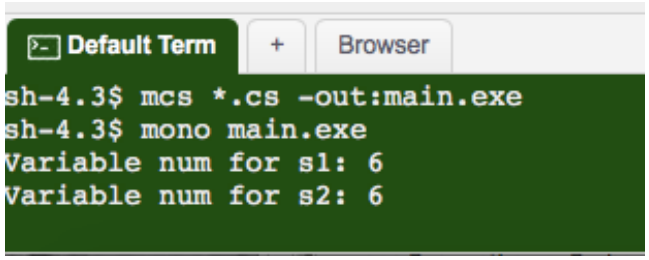
Static Members of a Class

- Kita dapat mendefinisikan anggota kelas sebagai statis (static) menggunakan keyword **static**
- Ketika kita mendeklarasikan anggota dari kelas statis, artinya berapapun banyaknya objek dari kelas (class) yang diciptakan, hanya ada satu salinan (copy) dari anggota statis (static member)
- Keyword **static** menyiratkan bahwa hanya ada satu contoh dari anggota yang ada untuk kelas, variable statis digunakan untuk mendefinisikan konstanta karena nilai-nilai mereka dapat diambil dengan menerapkan kelas tanpa menciptakan sebuah instance.
- Variabel statis dapat diinisialisasi diluar fungsi anggota atau definisi kelas.
- Static class member adalah member class yg diakses melalui class tidak melalui objek. Jadi seolah oleh member static bersifat global di semua objek.

Static Member

```
using System;

namespace StaticVarApplication
{
   class StaticVar
   {
      public static int num;
      public void count()
      {
         num++;
      }
      public int getNum()
      {
         return num;
      }
   }
   class StaticTester
   {
      static void Main (string[] args)
      {
         StaticVar s1=new StaticVar();
         StaticVar s2=new StaticVar();
         s1.count();
         s1.count();
         s1.count();
         s2.count();
         s2.count();
         s2.count();
         Console.WriteLine("Variable num for s1: {0}", s1.getNum());
         Console.WriteLine("Variable num for s2: {0}", s2.getNum());
         Console.ReadKey();
      }
   }
}
```

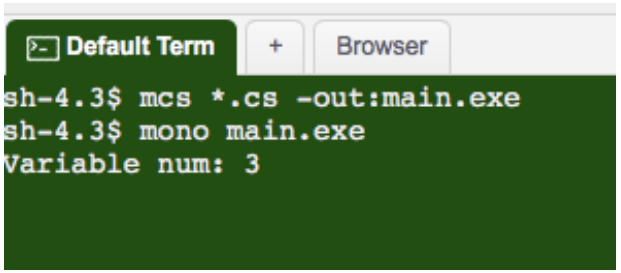


Member function as Static

- Mendeklarasikan fungsi anggota sebagai statis.
- Fungsi tersebut dapat mengakses variabel hanya statis.
- Fungsi statis ada bahkan sebelum objek dibuat.
  - Contoh berikut menunjukkan penggunaan fungsi statis:

Member function as Static

```
using System;
namespace StaticVarApplication
{
   class StaticVar
   {
      public static int num;
      public void count()
      {
         num++;
      }
      public static int getNum()
      {
         return num;
      }
   }
   class StaticTester
   {
      static void Main(string[] args)
      {
         StaticVar s = new StaticVar();
         s.count();
         s.count();
         s.count();
         Console.WriteLine("Variable num: {0}", StaticVar.getNum());
         Console.ReadKey();
      }
   }
}
```



Referensi

- C# Programming, Object Oriented Programming, Tutorialspoint
  - www.tutorialspoint.com