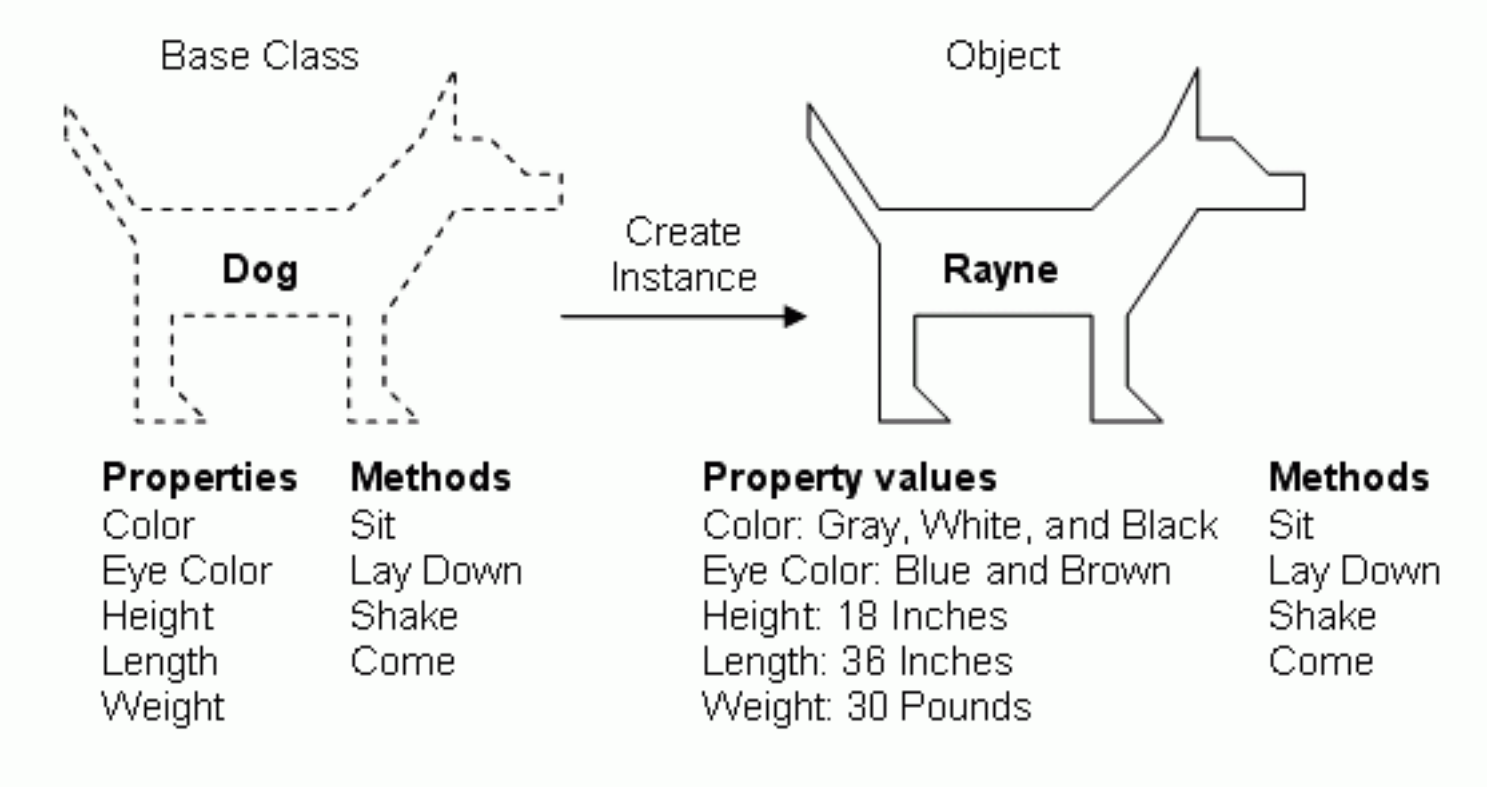
OOP principi: klasa/objekt

Klasa je opisnik objekta
Objekt je pojavnost (instanca) klase

Klase omogućuju opisivanje realnog sustava kreirajući podatkovni model za koji radimo aplikaciju u čovjeku razumljivom načinu.

Klase mogu imati
- svojstva
- metode



Izvor:
http://programmingbond.blogspot.com/2014/06/object-and-class.html

OOP principi: klasa/objekt

Klasa je referencirajući tip podatka

Deklariranje klase Klasa

```
1   using System;
2   namespace Predavanje6
3   {
4       public class Klasa
5       {
6           private int svojstvo;
7
8           public int Svojstvo
9           {
10              get
11              {
12                  return svojstvo;
13              }
14              set
15              {
16                  this.svojstvo = value;
17              }
18          }
19
20          public int Broj { get; set; }
21
22
23      }
24  }
25
```

Pojavnost klase (instanca) – objekt s

```
1   using System;
2
3   namespace Predavanje6
4   {
5       class Program
6       {
7           static void Main(string[] args)
8           {
9               Klasa s = new Klasa();
10
11              s.Broj = 1;
12              s.Svojstvo = 2;
13
14              Console.WriteLine("{0}, {1}", s.Broj, s.Svojstvo);
15
16          }
17      }
18  }
19
```

```
Terminal – Predavanje6

1, 2
```

Kraće sintakse

```
Klasa s = new Klasa
{
    Broj = 1,
    Svojstvo = 2
};
```

```
Klasa s = new()
{
    Broj = 1,
    Svojstvo = 2
};
```

Izvor:
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/classes

OOP principi: klasa/objekt

Klasa ima kao svojstvo drugu klasu (ugrađenu ili vlastitu)

```
using System;
namespace Predavanje6.KlasaKoristenje
{
    public class Mjesto
    {
        public string Naziv { get; set; }
        public int BrojStanovnika { get; set; }
        public Zupanija zupanija { get; set; }
    }
}
```

```
using System;
namespace Predavanje6.KlasaKoristenje
{
    public class Zupanija
    {
        public string Naziv { get; set; }
        public Zupan Zupan { get; set; }
    }
}
```

```
using System;
namespace Predavanje6.KlasaKoristenje
{
    public class Zupan
    {
        public string Ime { get; set; }
        public string Prezime { get; set; }
    }
}
```

```
Zupan zupan = new Zupan();
zupan.Ime = "Mario";
zupan.Prezime = "Perit";

Zupanija zupanija = new Zupanija();
zupanija.Zupan = zupan;
zupanija.Naziv = "Moja županija";

Mjesto mjesto = new Mjesto();
mjesto.zupanija = zupanija;
mjesto.Naziv = "Donji Miholjac";
mjesto.BrojStanovnika = 12000;

Console.WriteLine(mjesto.zupanija.Zupan.Ime);
```

```
Terminal – Predavanje6

Mario
```

struct

Struktura je vrijednosni tip podatka

Deklariranje strukture Struktura Klasa

```
1     using System;
2     namespace Predavanje6
3     {
4         public struct Struktura
5         {
6             public int Svojstvo { get; set; }
7
8             public int Broj { get; set; }
9
10        }
11    }
```

Korištenje strukture

```
var st = new Struktura();
st.Svojstvo = 1;
st.Broj = 2;

Console.WriteLine("{0}, {1}", st.Broj, st.Svojstvo);
```

```
Terminal – Predavanje6

2, 1
```

U nastavku predavanja raditi ćemo isključivo s klasama.

Izvori:
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/struct
https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/choosing-between-class-and-struct

Ključna riječ var

Implicitno/eksplicitno deklariranje varijable

```
int i = 0; //eksplicitno
var j = 0; //implicitno

Klasa klasa = new(); //eksplicitno
var k = new Klasa(); //implicitno


klasa.Broj = 2;
k.Svojstvo = 1;

var x = i + j + klasa.Broj + k.Svojstvo;

Console.WriteLine("{0}", x);
```

var podrazumijeva implicitno dodjeljivanje:
desna strana će definirati kojeg je tipa

```
var t = 0;

// jako puno koda tako da
// dođete do dijela kada niste sigurni kojeg je varijabla tipa

Console.WriteLine("{0}", t.GetType());
```

```
Terminal – Predavanje6

System.Int32
```

```
var o = new Klasa();
Console.WriteLine("{0}", o.GetType());
```

```
Terminal – Predavanje6

Predavanje6.Klasa
```

Izvor:
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/var

Rad s klasama/objektima

String klasa

* dodatno istraživanje, izlazi iz opsega nastave

Rad s klasama/objektima

Jednakost objekata

```
var s1 = "Osijek";
var s2 = "Osijek";

Console.WriteLine(s1 == s2);
Console.WriteLine(s1.Equals(s2));
```

```
Terminal – Predavanje6

True
True
```

```
var m1 = new Mjesto
{
    Naziv = "Osijek"
};
var m2 = new Mjesto
{
    Naziv = "Osijek"
};

Console.WriteLine(m1 == m2);
Console.WriteLine(m1.Equals(m2));
```

```
Terminal – Predavanje6

False
False
```

```
Console.WriteLine(m1.GetHashCode());
Console.WriteLine(m2.GetHashCode());
```

```
Terminal – Predavanje6

58225482
54267293
```

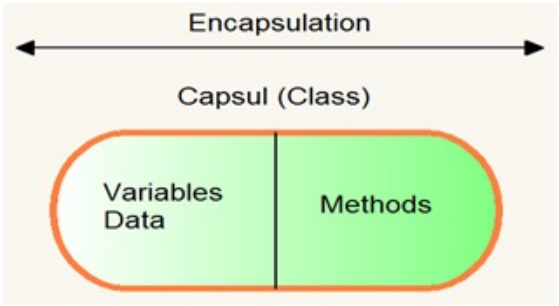
Izvor:
https://docs.microsoft.com/en-us/dotnet/api/system.object.equals?view=net-5.0

OOP principi: učahurivanje

Princip kojim klasa **skriva svoja svojstva** (čini ih privatnima) te **omogućuje pristup svojstvima putem javnih metoda**

```
1   using System;
2   namespace Predavanje6
3   {
4       public class Klasa
5       {
6           private int svojstvo;
7
8           public int Svojstvo
9           {
10              get
11              {
12                  return svojstvo;
13              }
14              set
15              {
16                  this.svojstvo = value;
17              }
18          }
19
20          public int Broj { get; set; }
21
22
23      }
24  }
25
```

eng. Encapsulation zato što se klasa promatra kao kapsula.



Zašto učahurivanje

- Fleksibilnost: Fleksibilnije je i lakše mijenjati učahureni kôd s novim zahtjevima.
- Ponovna upotreba: Učahureni kôd može se ponovno upotrijebiti u cijeloj aplikaciji ili u više aplikacija.
- Održavanje: Dijelovi aplikacije su učahureni u zasebne jedinice, lako je promijeniti ili ažurirati dio aplikacije bez utjecaja na druge dijelove, što smanjuje vrijeme održavanja.

Izvori:

https://www.geeksforgeeks.org/c-sharp-encapsulation/

https://github.com/tjakopec/OOP_JAVA_PHP_PYTHON_SWIFT

Metode

5. Konstruktor – specifična metoda, poziva se u trenutku instanciranja klase

Deklaracija metode unutar klase

```
1       using System;
2       namespace Predavanje6
3       {
4           public class Metode
5           {
6
7               public Metode()
8               {
9                   Console.WriteLine("Ispis iz konstruktora");
10              }
11
12              public Metode(int i)
13              {
14                  Console.WriteLine("Ispis iz konstruktora, primio sam {0}",i);
15              }
16
17          }
18      }
```

Deklaracija instance klase

```
var m = new Metode();
new Metode(7);
```

Rezultat izvođenja

```
Terminal – Predavanje6

Ispis iz konstruktora
Ispis iz konstruktora, primio sam 7
```

Naziv mora biti identičan nazivu klase u kojoj se nalazi