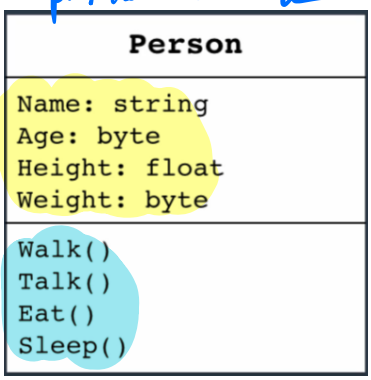
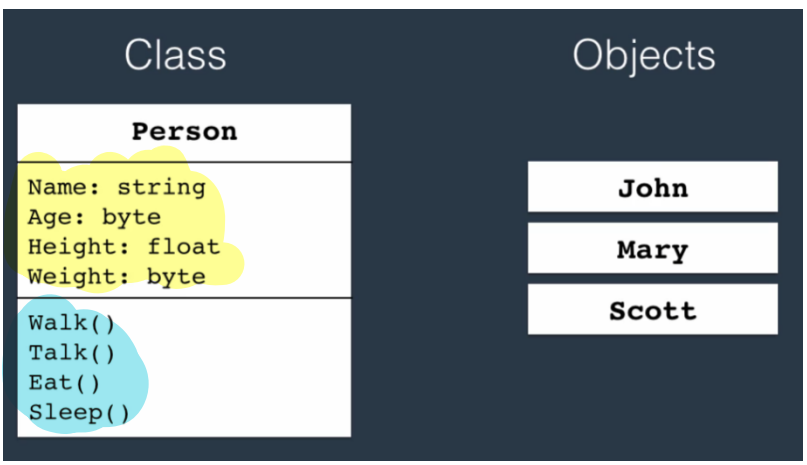
6. Třídy

- Anatomie třídy
  - data
    - vlastnosti
    - atributy
  - chování
    - metody
    - funkce
- Co je to objekt?
  - instance třídy

příklad v UML






```
public class Person {
    ....
}
```

- modifikátor přístupu
- název – PascalCase

- třída = předpis pro objekt (šablona) - člověk

- objekt = instance třídy (konkrétní věc) - pepa

Příklad třídy:

```
namespace Tridy
{
    Počet odkazů: 1
    public class Person
    {
        public readonly string FirstName; // proměnná readonly lze pouze nastavit v konstruktoru, pak ji nelze změnit
        public string LastName; // public proměnné je PascalCase
        private int _age; // privátní proměnně se píšou _camelCase
        private double _weight;

        Počet odkazů: 0
        public Person(string firstName, string lastName, double weight)
        {
            this.FirstName = firstName;
            this.LastName = lastName;
            this._age = 0;
            this._weight = weight;
        }

        Počet odkazů: 0
        public void IntroduceYourself()
        {
            Console.WriteLine($"Hello, I am {this.FirstName} {this.LastName} and I am {this._age} years old and weigh {this._weight}kg.");
        }
    }
}
```

- data
- konstruktor
- chování/metody

Vytvoření objektu z třídy Person

```
namespace Tridy
{
    Počet odkazů: 0
    internal class Program
    {
        Počet odkazů: 0
        static void Main(string[] args)
        {
            Person p = new Person("Jakub", "Doležal", 5);
            p.IntroduceYourself();
        }
    }
}
```

```
Konzola ladění sady Microsoft Visual Studio
Hello, I am Jakub Doležal and I am 0 years old and weigh 5kg.
```

Data

- mají různé modifikátory přístupu → podle toho jmenní konvence
  - ↳ public - PascalCase
  - ↳ private - _camelCase

```
public readonly string FirstName; // proměnná readonly lze pouze nastavit v konstruktoru, pak ji nelze změnit
public string LastName; // public proměnné je PascalCase
private int _age; // privátní proměnně se píšou _camelCase
private double _weight;
```

- readonly
  - ↳ jedná se o data, které se určí při vytvoření objektu a již je nelze měnit jsou jen ke čtení

Chování/metody

- tady se řídíme tím co známe → PascalCase

```
Počet odkazů: 1
public void IntroduceYourself()
{
    Console.WriteLine($"Hello, I am {this.FirstName} {this.LastName} and I am {this._age} years old and weigh {this._weight}kg.");
}

Počet odkazů: 0
private void Walk()
{
    // ... some code
}
```

Modifikátory přístupu

- zamezit přístupu ostatních tříd/objektů

- public
  - dostupné, viditelné
- private
  - nedostupné, viditelné pouze uvnitř třídy
- static
  - dostupné, viditelné, lze použít i bez vytvoření objektu

  např. `Console.WriteLine();`

Konstruktor

- inicializátor
- optional
- není void ani nic nevrací
- slouží k inicializaci dat

```
public readonly string FirstName; // proměnná readonly lze pouze nastavit v konstruktoru, pak ji nelze změnit
public string LastName; // public proměnné je PascalCase
private int _age; // privátní proměnně se píšou _camelCase
private double _weight;

Počet odkazů: 1
public Person(string firstName, string lastName, double weight)
{
    this.FirstName = firstName;
    this.LastName = lastName;
    this._age = 0;
    this._weight = weight;
}
```

konstruktor

- Přetížení konstruktoru

```
Počet odkazů: 0
public Person() { /*...*/ }
Počet odkazů: 0
public Person(string firstName, string lastName) { /*...*/ }
Počet odkazů: 1
public Person(string firstName, string lastName, double weight) { /*...*/ }
```

- Při přetížení lze využít jiný konstruktor
  - nepoužije se kód

```
Počet odkazů: 1
public Person(string firstName, string lastName) {
    this.FirstName = firstName;
    this.LastName = lastName;
    this._age = 0;
}
Počet odkazů: 1
public Person(string firstName, string lastName, double weight)
    : this(firstName, lastName)
{
    this._weight = weight;
}
```

← Zavolá se horní konstruktor

Enkapsulace

- každá třída dělá pouze jednu věc
- třídy spolu spolupracují
- třída A nemusí vědět o obsahu třídy B a naopak!
- kuchař nemusí vědět jak servírka roznáší jídlo a servírka nemusí vědět, jak kuchař vaří