Наследяване

Йерархии от класове

BY NC SA

Учителски екип

Обучение за ИТ кариера

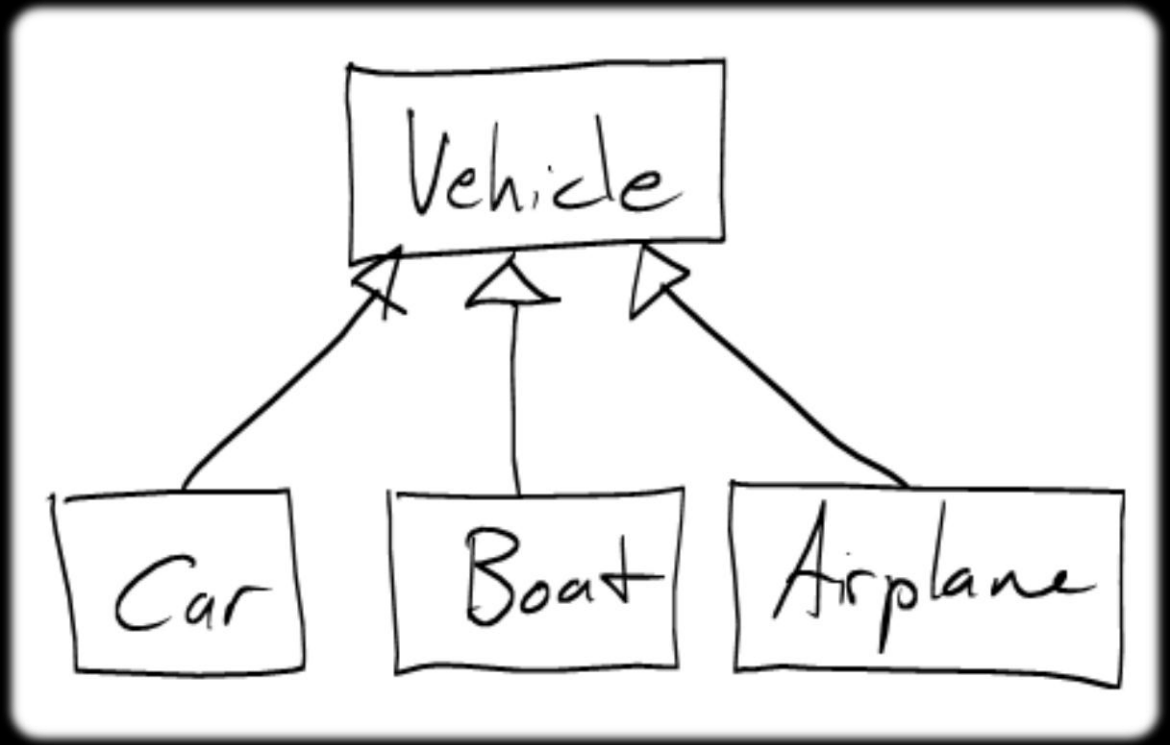
https://it-kariera.mon.bg/e-learning/

Съдържание

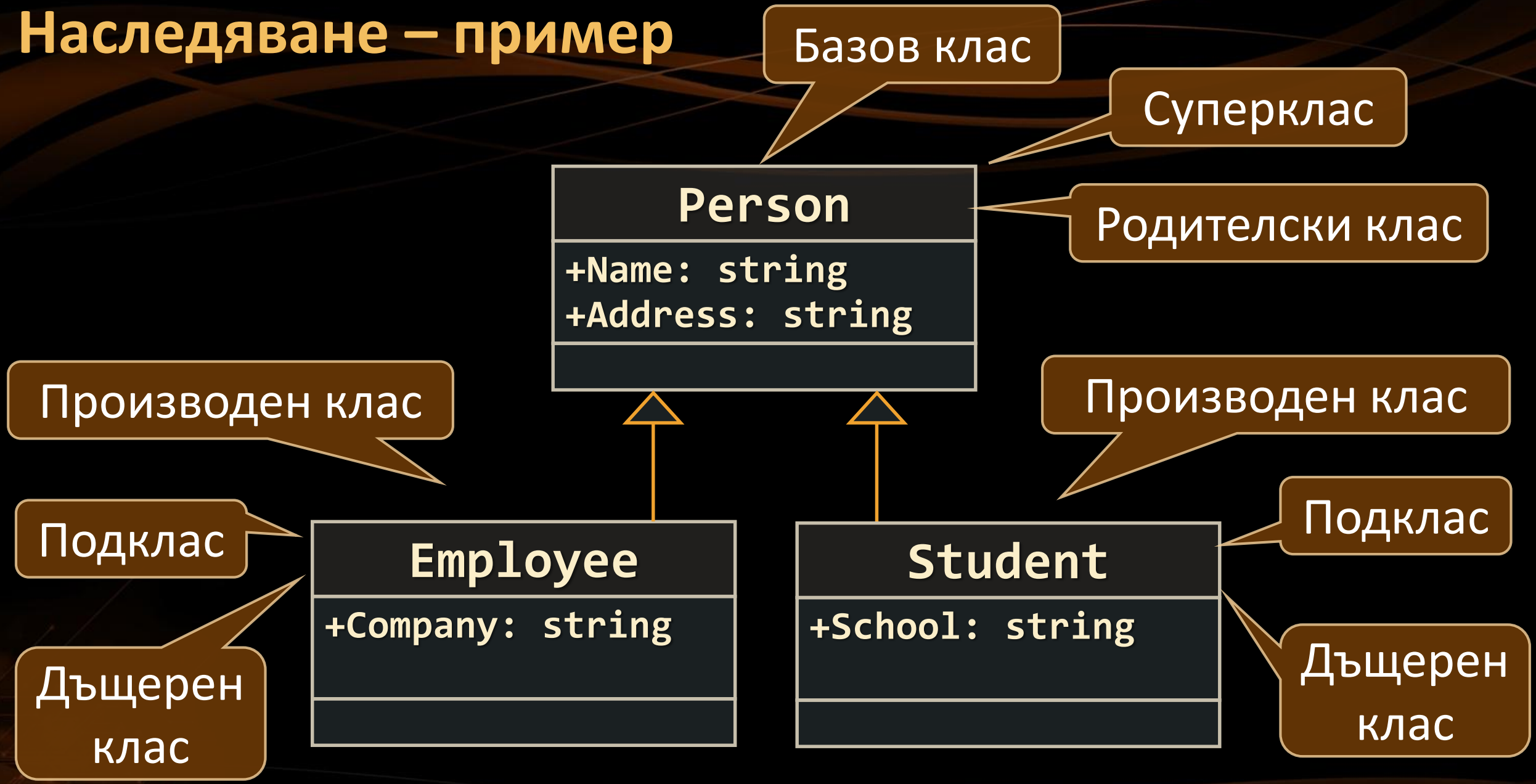
1. Наследяване
2. Йерархии от класове
3. Наследяване в C#
4. Достъп до членове на базовия клас
5. Повторно използване на класове

Наследяване

- Начин за разширяване на класовете и повторно използване на кода



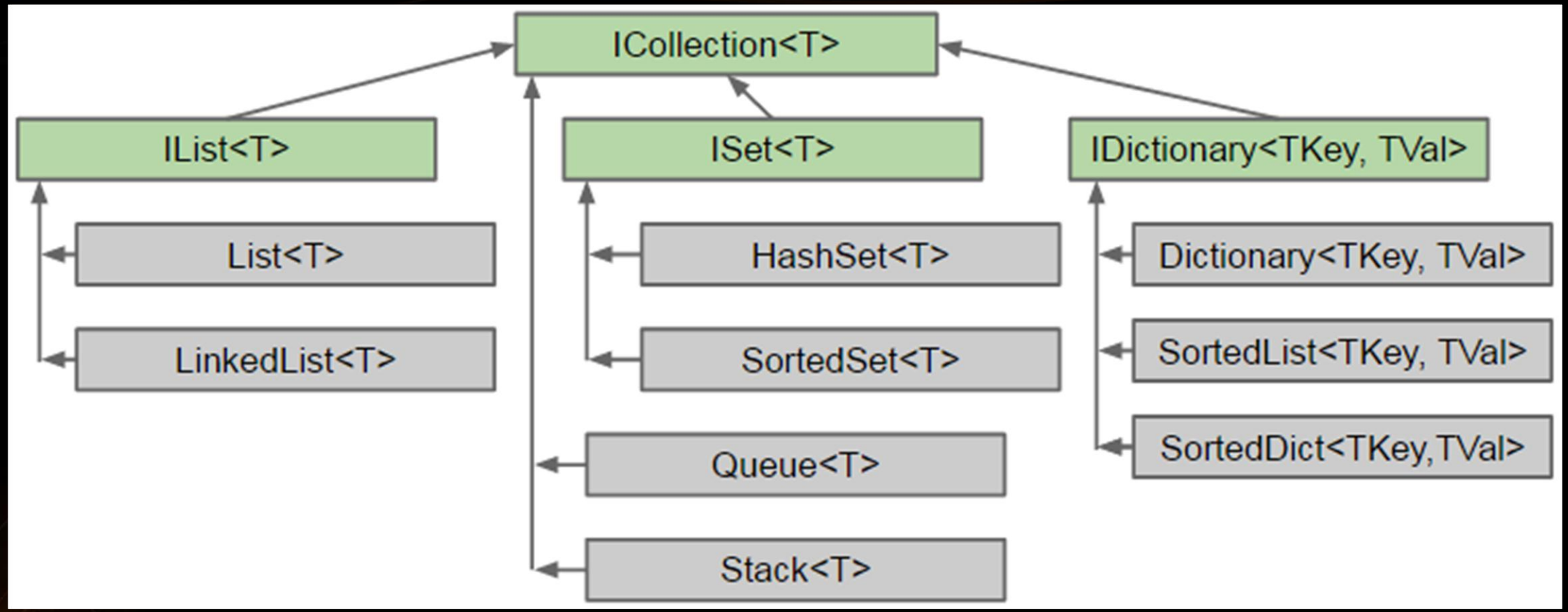
Наследяване – пример



Йерархия от класове

- Наследяването води до йерархии от класове и/или интерфейси в приложението:

Йерархия от класове – C# Collection

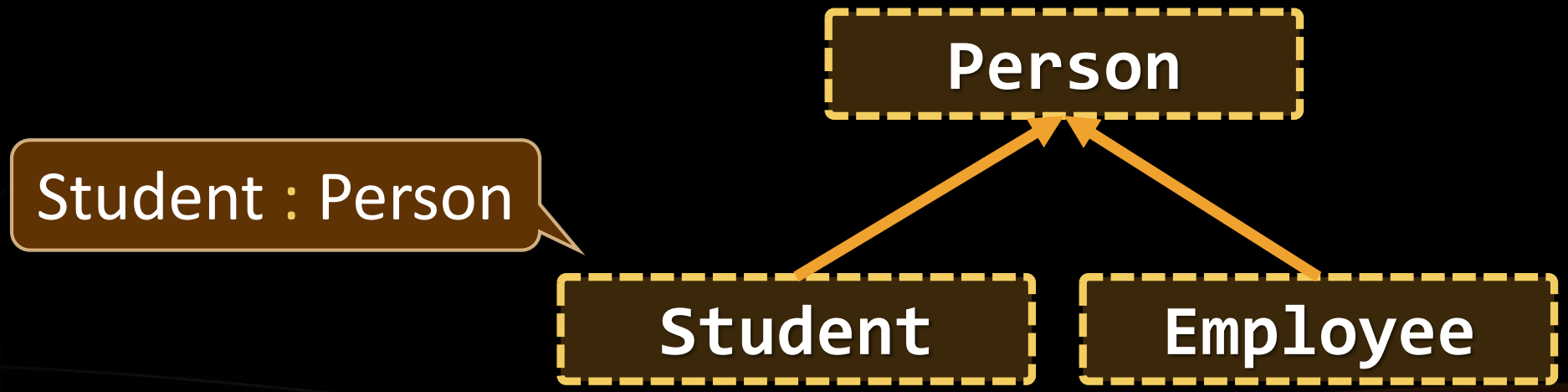


Наследяване в C#

- В C# наследяването се отбелязва чрез : оператора

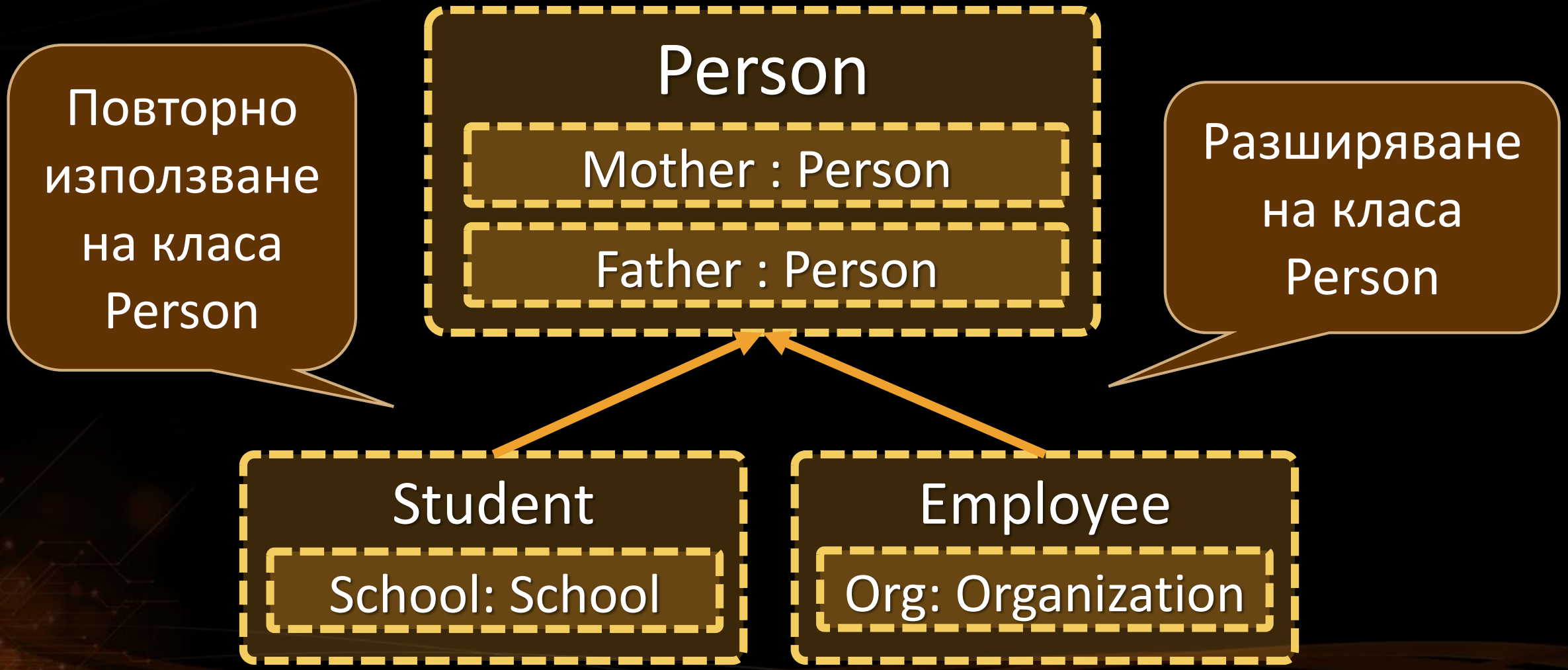
```
class Person { … }

class Student : Person { … }
class Employee : Person { … }
```



Наследяване – дъщерен клас

- Класът получава всички членове от родителския си клас



Използване на наследени членове

- Наследените членове се използват както обикновено:

```
class Person { public void Sleep() { … } }
class Student : Person { … }
class Employee : Person { … }
```

```
Student student = new Student();
student.Sleep();
Employee employee = new Employee();
employee.Sleep();
```

Преизползване на конструктори

- Конструкторите не се наследяват
- Конструкторите може да се ползват от дъщерните класове

```
class Student : Person {
  private School school;
  public Student(String name, School school)
    :base(name)
  {
    this.school = school;
  }
}
```

Наследяване и модификатори за достъп

- Подкласовете могат да достъпят всички public и protected членове, както и могат да достъпят internal членовете, ако са в същия проект
- Private полетата не се наследяват в подкласовете

```
class Person {
  private string id;
  string name;
  protected string address;
  public void Sleep();
}
```

Може да се достъпи чрез другите методи на класа Person

„Засенчване“ на променливи

- Подкласовете могат да скрият променливи от суперкласа

```
class Person { protected int weight; }
```

```
class Patient : Person
{
  protected float weight;
  public void Method()
  {
    double weight = 0.5d;
  }
}
```

Скрива `int weight`

Скрива и двете

Достъп до членове на базовия клас

- Чрез ключовата дума **base**

```
class Person { … }
class Employee : Person
{
  void Fire(string reasons) {
    Console.Writeline
    ($"{base.name} got fired because {reasons}");
  }
}
```

„Засенчване“ на променливи – решение

- Използвайте **base** и **this**, за да уточните достъпа

```
class Patient : Person
{
  protected float weight;
  public void Method()
  {
    double weight = 0.5d;
    this.weight = 0.6f;
    base.weight = 1;
  }
}
```

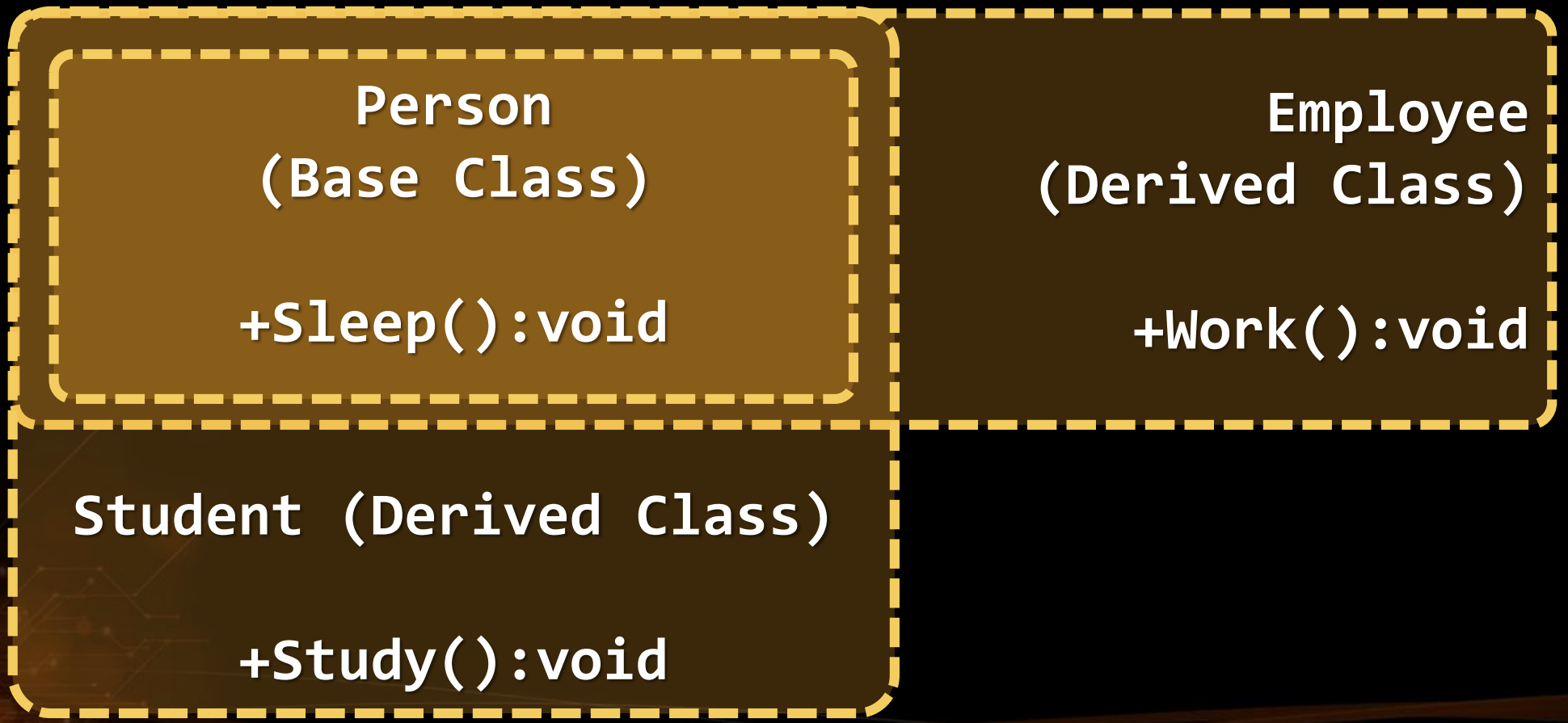
Локална променлива

Член на инстанцията

Член на базовия клас

Наследяването е разширяване

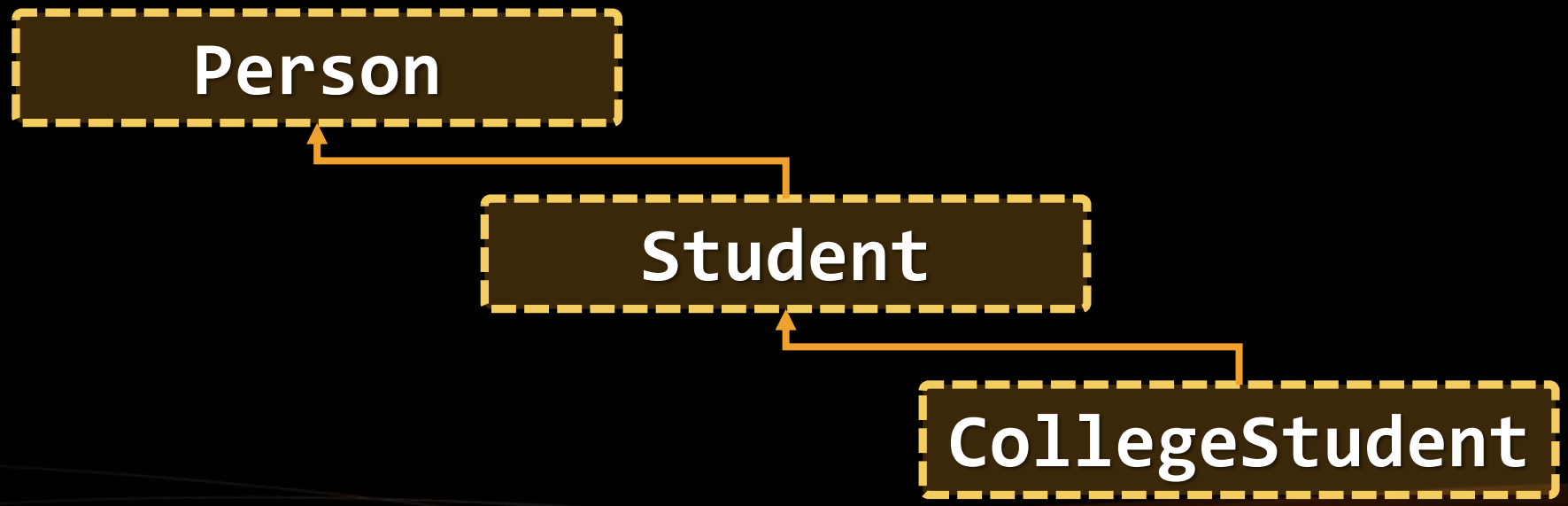
- Инстанцията на дъщерен клас съдържа инстанция на неговия базов клас



Наследяване

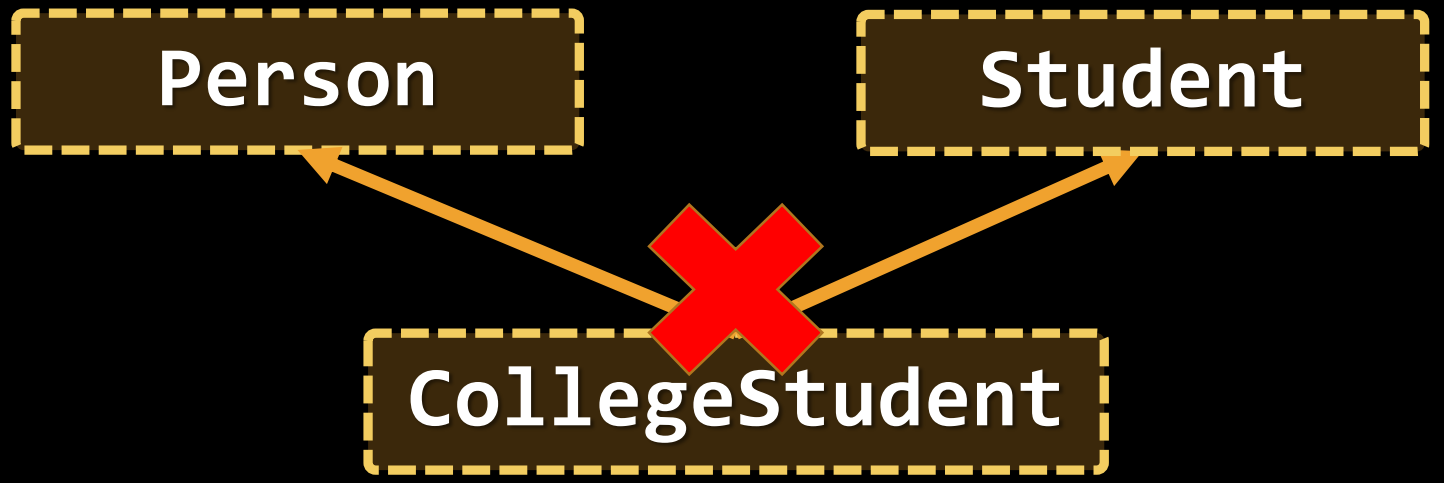
- Наследяването има преходна връзка

```
class Person { … }
class Student : Person { … }
class CollegeStudent : Student { … }
```



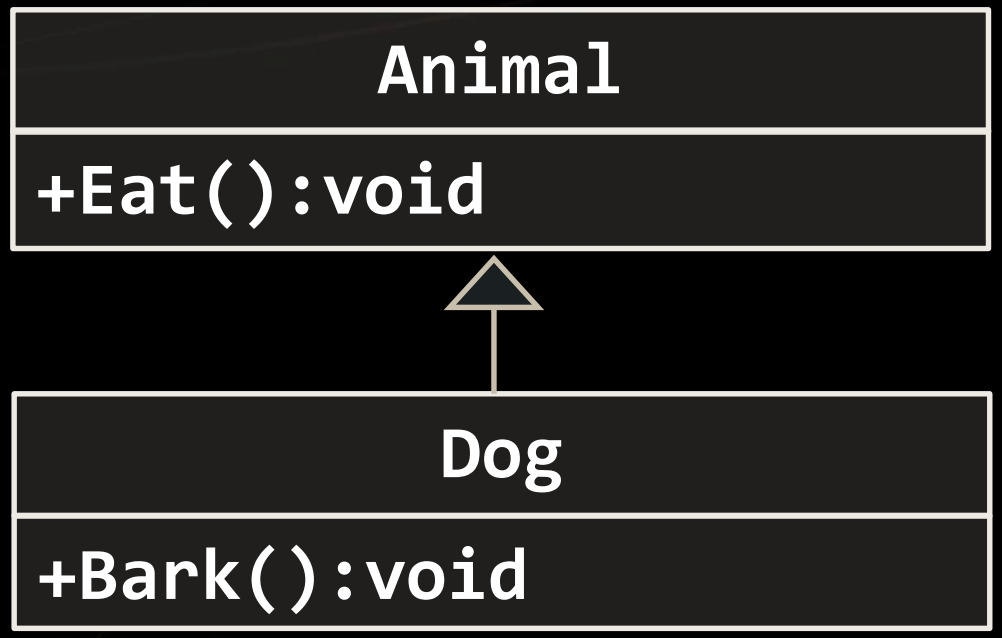
Множествено наследяване

- В C# не се поддържа множествено наследяване



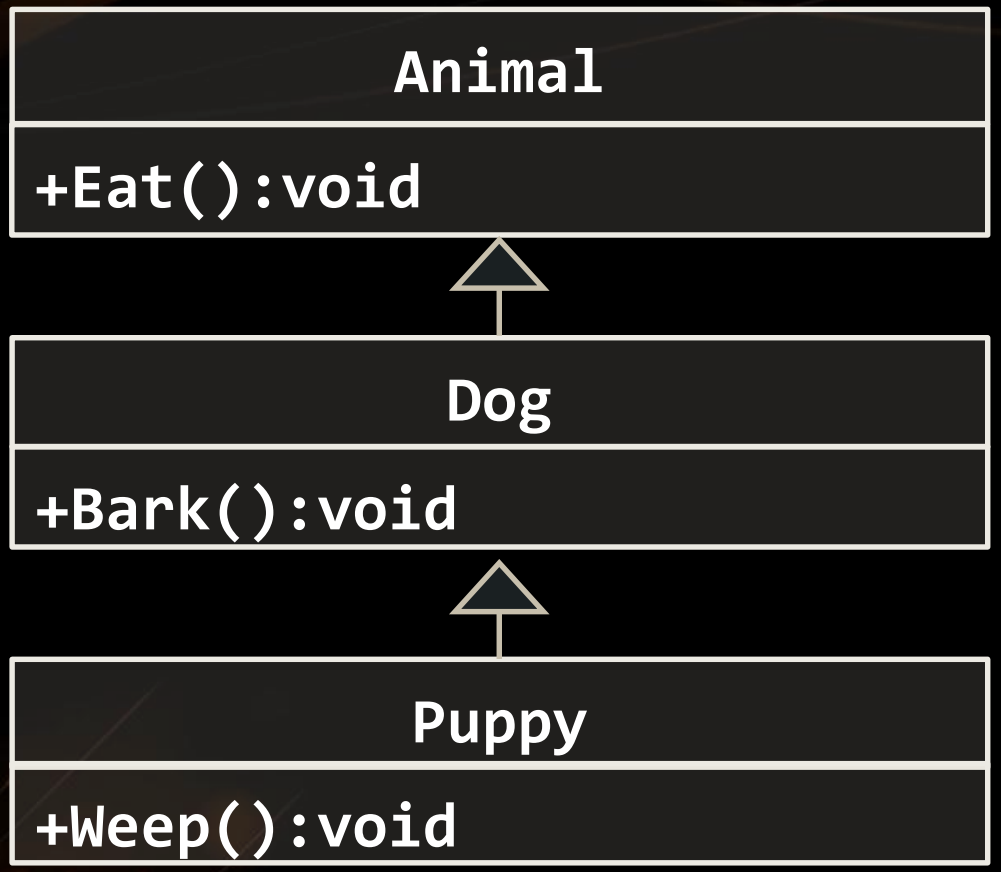
- Поддържа се само имплементиране на множество интерфейси

Задача: Наследяване



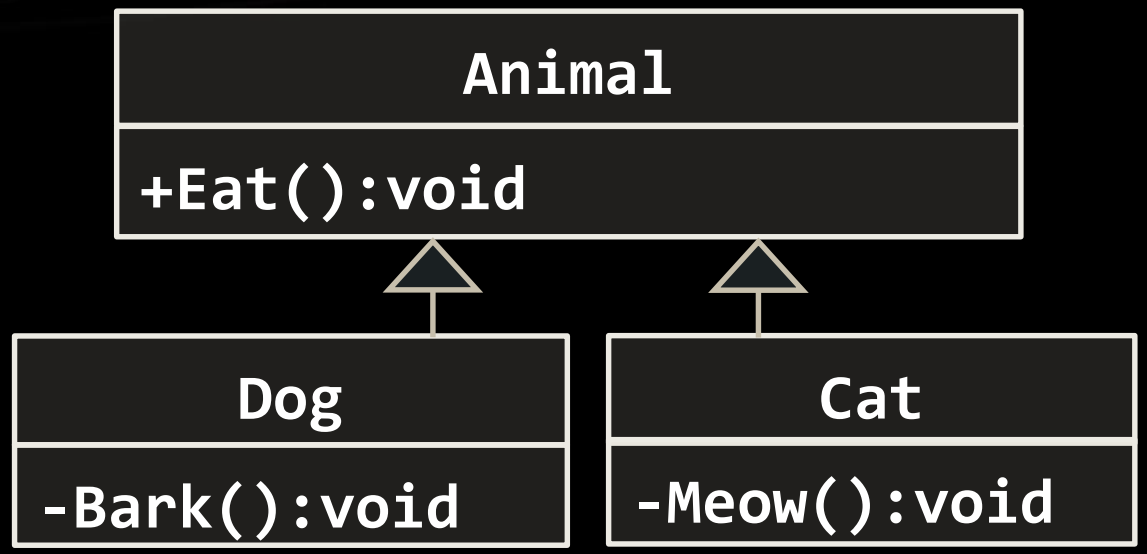

```
Dog dog = new Dog();
dog.Eat();
dog.Bark();
```

Задача: Наследяване на няколко нива




```
Puppy puppy = new Puppy();
puppy.Eat();
puppy.Bark();
puppy.Weep();
```

Задача: Йерархично наследяване




```
Dog dog = new Dog();
dog.Eat();
dog.Bark();

Cat cat = new Cat();
cat.Eat();
cat.Meow();
```

Видове повторно използване на класове

Разширяване, композиция, делегиране

Разширяване

- Дублирането на код е податливо на грешки
- Преизползване на код чрез разширение
- Това е отношение от типа E: свързаният списък E списък

Композиция

- Използване на класове в дефиницията на друг клас
- Това е отношение от типа ИМА:
  лаптопа ИМА клавиатура

```
class Laptop {
  Monitor monitor;
  Touchpad touchpad;
  Keyboard keyboard;
  …
}
```

Преизползване на класове

Делегиране

- Делегирането е отношение от типа ИЗВЪРШВА: мониторът ИЗВЪРШВА промяната на яркостта на лаптопа

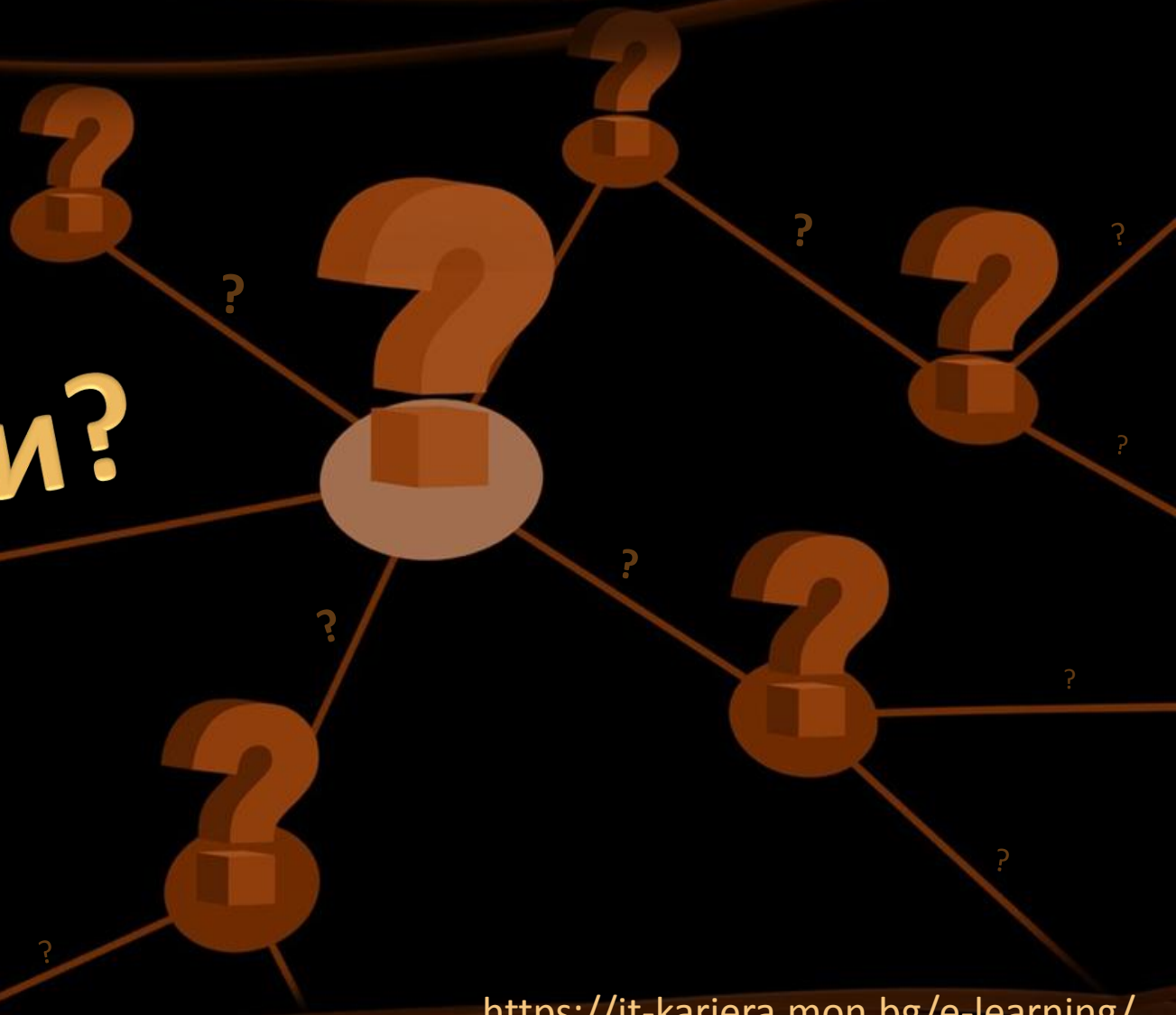
```
delegate void Setup();
class Laptop {
  private Monitor monitor;
  public Setup DecrBrightness;
  public Setup IncrBrightness;
  public Laptop() {
    this.monitor = new Monitor();
    this.DecrBrightness = monitor.Dim();
    this.IncrBrightness = monitor.Brighten();
  }
}
```

Какво научихме днес?

- Наследяването е инструмент за повторно използване на код
  - други начини са композиция и делегиране
- Производния клас наследява елементи от базовия клас
  - те могат да бъдат използвани все едно са негови
- Достъпът до елементите на базовия клас е възможен с base

Наследяване

Въпроси?



https://it-kariera.mon.bg/e-learning/

Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**