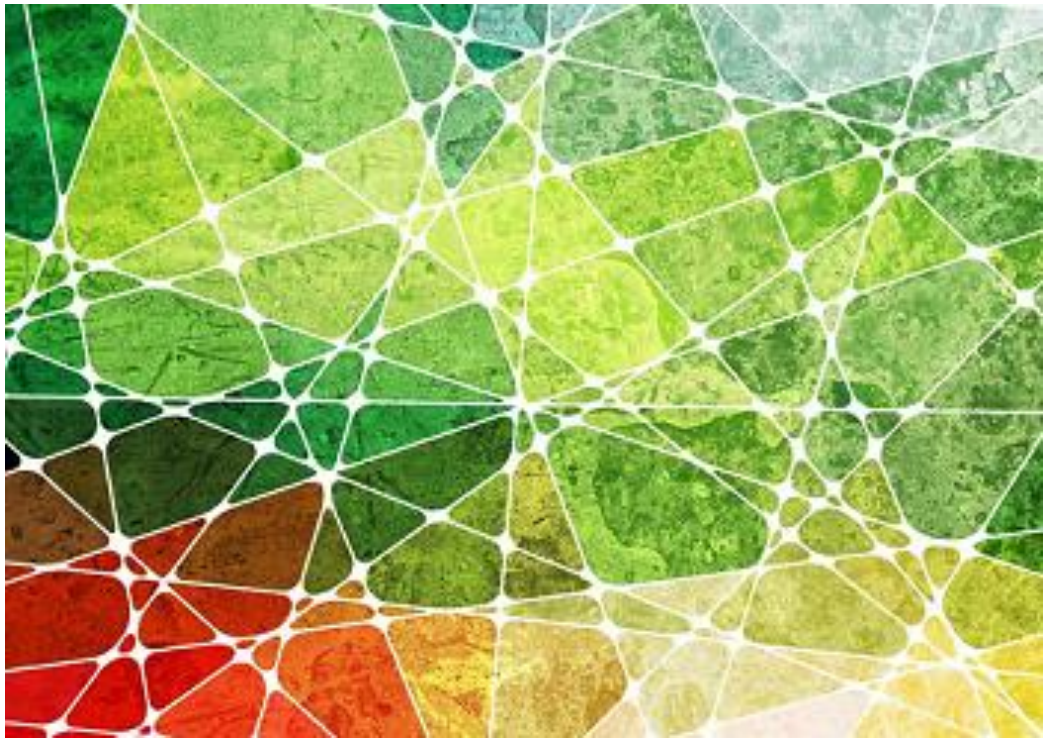
C#
VI
OOP

Abstract & Polymorphism

Plan

1. Abstractions
2. Polymorphism
3. Static classes



Abstraction definitions

АБСТРАКЦИЯ (от лат. abstractio - отвлечение) (абстрактное) – мысленное отвлечение от ряда свойств предметов и отношений между ними; понятие, образуемое в результате отвлечения.

АБСТРАКЦИЯ – форма познания, основанная на мысленном выделении существенных свойств и связей предмета и отвлечении от других, частных его свойств и связей;

АБСТРАКЦИЯ – мысленное обособление, отвлечение от тех или иных свойств, сторон или связей предметов и явлений для выделения существенных их признаков.

Abstract problem

```
using static System.Console;

class Animal { }


class Rabbit : Animal
{
    void Hunt()
    {
        // охотится на морковку
        WriteLine("Carrot O_O");
    }
}

class Cat : Animal
{
    void Hunt()
    {
        // охотится на мышку
        WriteLine("Mouse O_O");
    }
}
```

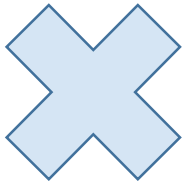
- Пусть существует базовый тип (*Parent*), от которого унаследовано несколько потомков (*Childs*)
- Все потомки имеют метод с одним и тем же назначением, но различными реализациями

  метод для каждого типа выполняет одно и тоже действие, но по-разному

Проблема: не можем вызвать метод через переменную базового класса

```
Animal rabbit = new Rabbit();
rabbit.Hunt();
```

"Animal" не содержит определения для "Hunt",



Abstract syntax

```
abstract class Animal
{
    abstract public void Hunt();
}

class Cat : Animal
{
    public override void Hunt()
    {
        // охотится на мышку
        Console.WriteLine("Mouse O_O");
    }
}

class Rabbit : Animal
{
    public override void Hunt()
    {
        // охотится на морковку
        Console.WriteLine("Carrot O_O");
    }
}
```

Абстрагируемся от частностей (реализации), добавляем абстракцию в базовый класс

Реализуем абстрактный метод в классах потомках

для каждого типа своя реализация абстрактного метода

```
Animal animal;

animal = new Rabbit();
animal.Hunt();      // Carrot O_O

animal = new Cat();
animal.Hunt();      // Mouse O_O
```

Abstract keyword

```
abstract class Animal
{
    abstract public void Hunt();

    public void Eat(int weight)
    {

    }
}
```

```
class Rabbit : Animal  { }
```

class ForPresentation.Rabbit

"Rabbit" не реализует наследуемый абстрактный член "Animal.Hunt()".

Показать возможные решения (Alt+ВВОДилиCtrl+ю)

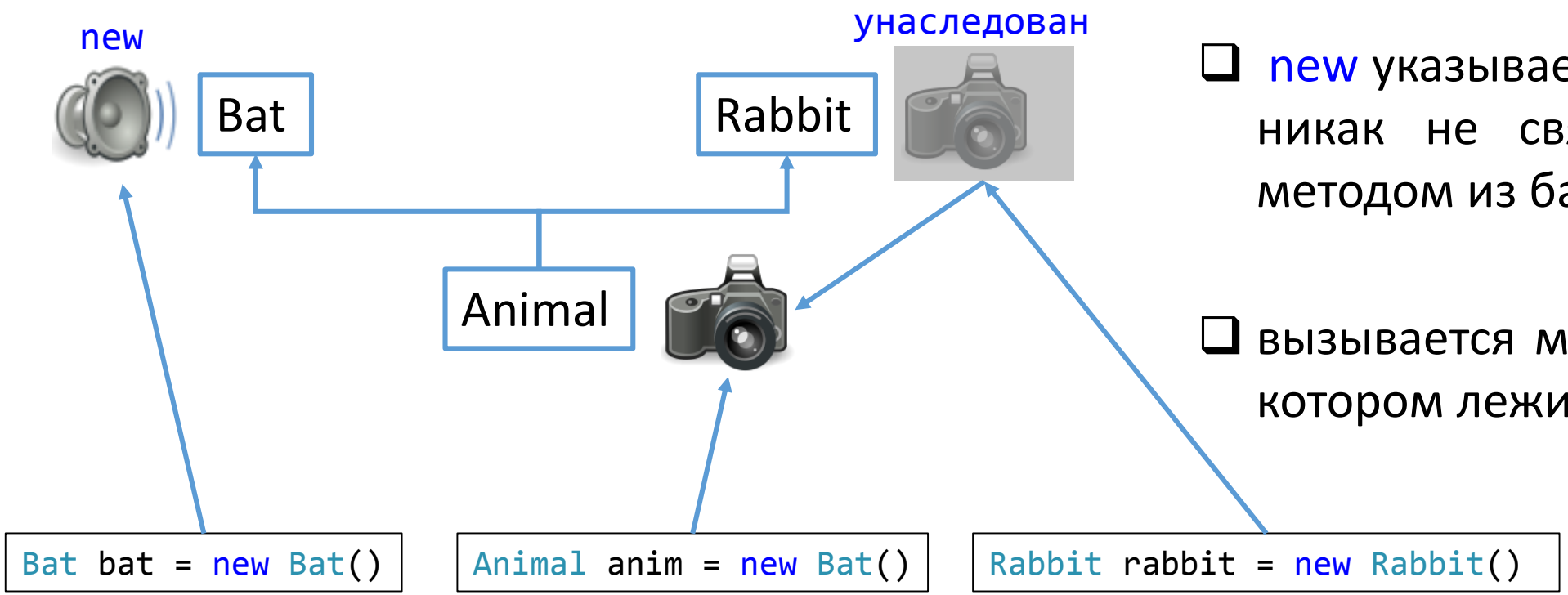
- Метод который объявляется с ключевым словом `abstract` не имеет реализации
- Класс у которого есть хотя бы один абстрактный метод, обязан быть `abstract`. Объект абстрактного класса создать нельзя!
- Классы потомки должны реализовывать все абстрактные методы (`override`)
- Поля нельзя сделать абстрактными
- Свойства могут быть абстрактными, т.к. они методы

Перезапись методов в производном классе

```
class Animal
{
    public void See(object[] surroundings)
    {
        WriteLine("Form a VISIBLE image");
    }
}

class Rabbit : Animal
{
}


class Bat : Animal
{
    public new void See(object[] surroundings)
    {
        WriteLine("Form an ULTRASONIC image");
    }
}
```

```
Animal animal = new Rabbit();
// Form a VISIBLE image
animal.See(new[] { "house", "tree", "car" });



Bat bat = new Bat();
// Form a ULTRASONIC image
bat.See(new[] { "house", "tree", "car" });



animal = bat;
// Form a VISIBLE image
animal.See(new[] { "house", "tree", "car" });
```

new Keyword



- new указывает на то, что метод никак не связан с похожим методом из базового класса
- вызывается метод того типа, в котором лежит переменная

Виртуальный вызов метода

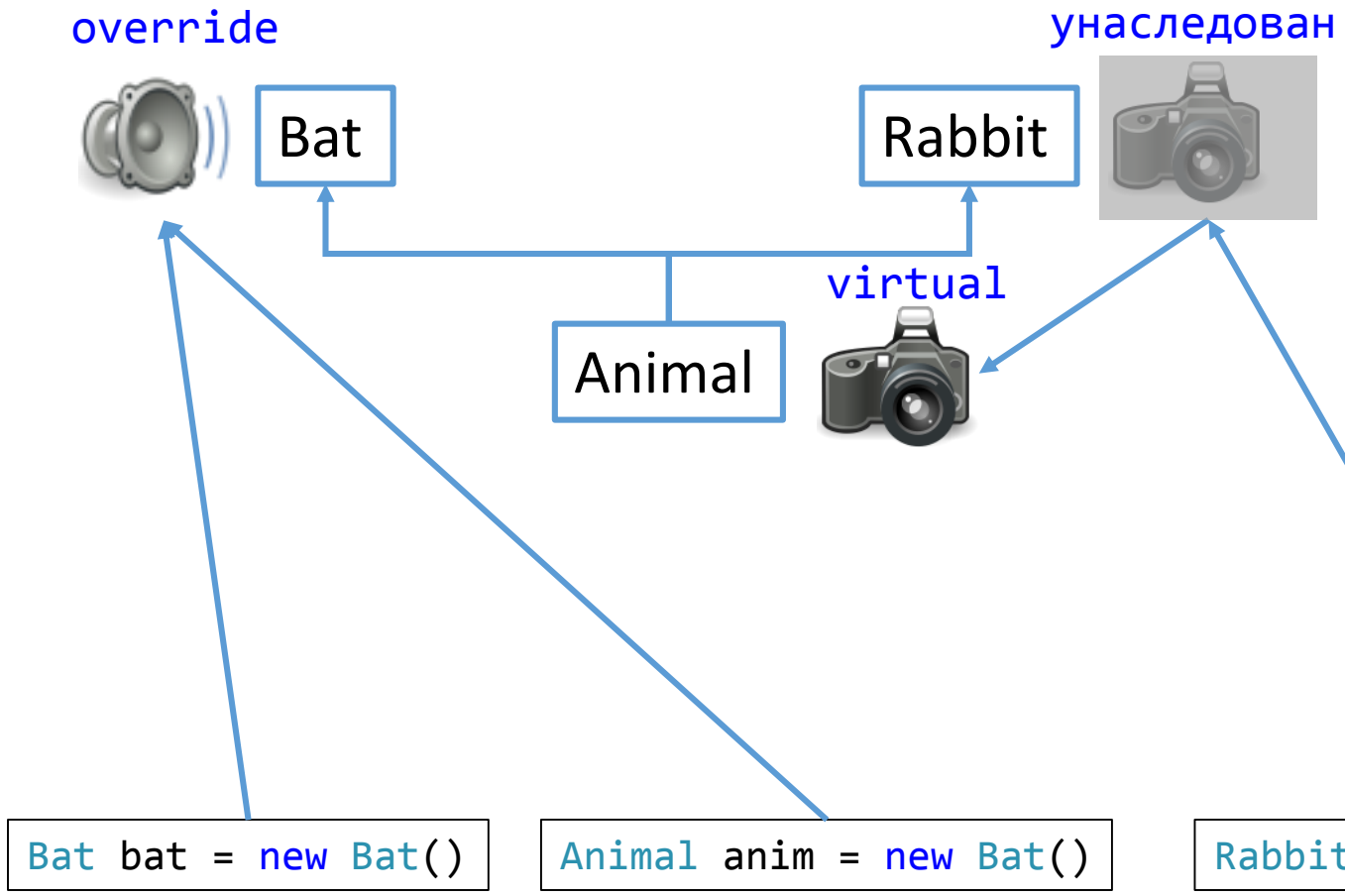
```
class Animal
{
   public virtual void See(object[] surroundings)
   {
      WriteLine("Form a VISIBLE image");
   }
}

class Rabbit : Animal
{
}

class Bat : Animal
{
   public override void See(object[] surroundings)
   {
      WriteLine("Form an ULTRASONIC image");
   }
}
```

```
Animal animal = new Rabbit();
// Form a VISIBLE image
animal.See(new[] { "house", "tree", "car" });
```

```
Bat bat = new Bat();
// Form a ULTRASONIC image
bat.See(new[] { "house", "tree", "car" });
```

```
animal = bat;
// Form a ULTRASONIC image
animal.See(new[] { "house", "tree", "car" });
```

virtual / override Keywords



- override указывает, что метод перезаписывает базовый виртуальный метод
- вызывается метод того типа, который использовался при создании объекта, а не в котором лежит переменная

это и есть полиморфизм!

Polymorphism

- Полиморфизм — одна из трех основных парадигм ООП
- Полиморфизм в языках программирования и теории типов — способность функции обрабатывать данные разных типов
- Достигается виртуальными вызовами методов
- Виртуальные методы в базовом классе помечаются:
  - либо словом virtual (если существует одинаковая реализация для большинства производных классов)
  - либо abstract (если у каждого производного класса своя реализация данного метода).
- В производном классе виртуальные методы перезаписываются override

```
static void SeeAllAnimals(Animal[] anim)
{
   foreach (var item in anim)
   {
      item.See(new[] { "house", "tree"});
   }
}

static void Main(string[] args)
{
   var animals = new  Animal[] {
                      new Rabbit(),
                      new Bat(),
                      new Cat()   };

   SeeAllAnimals(animals);
}
```

```
Form a VISIBLE image
Form an ULTRASONIC image
Form a VISIBLE image
```

new для виртуальных методов

```
class Animal
{
   public virtual void See(object[] surroundings)
   {
      WriteLine("Form a VISIBLE image");
   }
}
class Bat : Animal
{
   public override void See(object[] surroundings)
   {
      WriteLine("Form an ULTRASONIC image");
   }
}
class BatMan : Bat
{
   public new void See(object[] surroundings)
   {
      // Следить за правопорядком
      WriteLine("Follow the rule of law");
   }
}
```

```
BatMan batMan = new BatMan();

batMan.See(null);
((Bat)batMan).See(null);
((Animal)batMan).See(null);
```

```
Follow the rule of law
Form an ULTRASONIC image
Form an ULTRASONIC image
```

- BatMan.See() – уже не виртуальный
- Ключевое слово new – прерывает цепочку виртуальности
- Можно начать новую цепочку виртуальных вызовов virtual new

Other KeyWords

Ключевое слово	Тип	Метод (Свойство / Событие)	Константа / Поле
abstract	Экземпляры такого типа нельзя создавать	Метод необходимо переопределить в производном типе	-
virtual	-	Метод можно переопределить в производном типе	-
override	-	Переопределение метода	-
sealed	Тип нельзя использовать в качестве базового при наследовании	Метод нельзя переопределить в производном типе (может применяться только к методу, переопределяющему виртуальный метод)	-
new	Применительно к вложенному типу, методу, константе или полю означает, что член никак не связан с похожим членом, который может существовать в базовом классе.		

Static classes & methods

- Объекты статических классов создавать нельзя
- Используются для логической группировки схожих функций. Например, классы `Math`, `Console`.
- Не путать со статическими методами – статические методы не могут обращаться к нестатическим полям
- Статические классы содержат только статические методы и только статические поля

```
public static class Calculator
{
    // params - возможность передачи
    // переменного количества аргументов
    public static int Sum(params int[] c) {
        int sum = 0;
        foreach (var item in c) {
            sum += item;
        }

        return sum;
    }

    public static double Devide(int a, int b) {
        return a / b;
    }
}

Calculator.Sum(params int[] c) {
```


Спасибо за внимание!