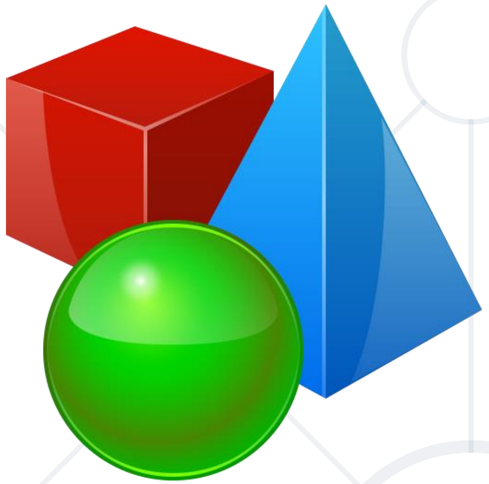
Objects and Classes

Using Objects and Classes

Defining Simple Classes



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#prgm-for-qa

Table of Contents

Objects and Classes

What is an Object? What is a Class?

Classes

- **Classes** provide the structure for **objects**
  - Act as **templates** for **objects** of the same type
- Classes define
  - **Properties** (data)
  - **Behaviors** (actions)
- One class may have many instances
- Sample class: **Dog**
  - Sample objects: **`sparky`**, **`rufus`**

Classes – Example

```
class Dog
{
        public string Name { get; set; }
        public string Breed { get; set; }
        public int Age { get; set; }
        public void Bark()
        {
                Console.WriteLine("Bark!");
        }
}
```

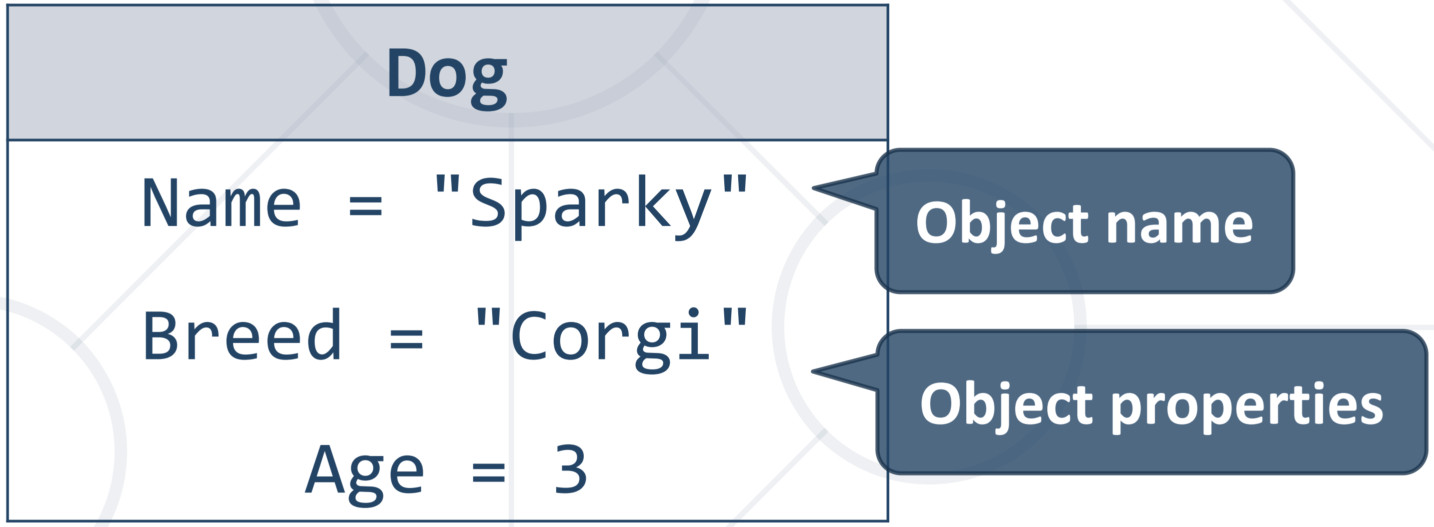
Name

Properties

Method

Objects

- An **object** holds a set of named values
  - Creating a **Dog** object



Example: Objects

Create a new object of type Dog

```
Dog puppy = new Dog ("Sparky", "Corgi", 3);
```

The new operator creates a new object

```
Dog puppy = new Dog
{Name = "Sparky", Breed = "Corgi", Age = 3 };
```

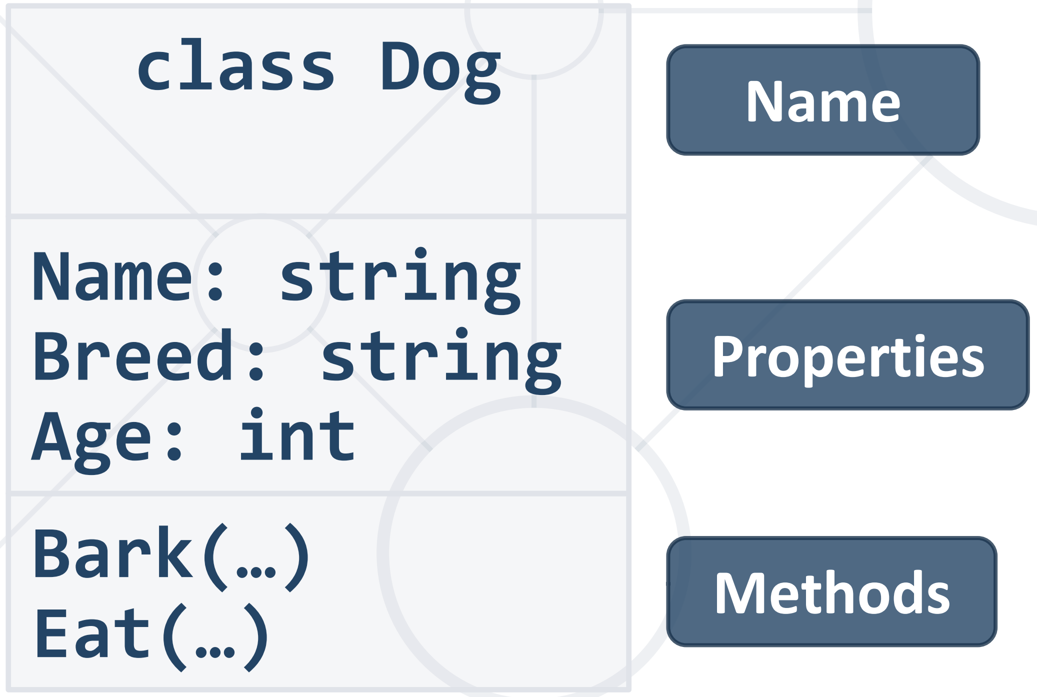
Objects – Instances of Classes

- Creating the object of a defined class is called **instantiation**
- The **instance** is the object itself, which is created runtime
- All instances have common **behaviour**

```
Dod sparky = new Dog("Sparky", "Corgi", 5);
Dog rufus = new Dog("Rufus", "Shepherd", 3);
Dog allie = new Dog("Allie", "Husky", 2);
```

Classes vs Objects

- Classes provide **structure** for creating objects

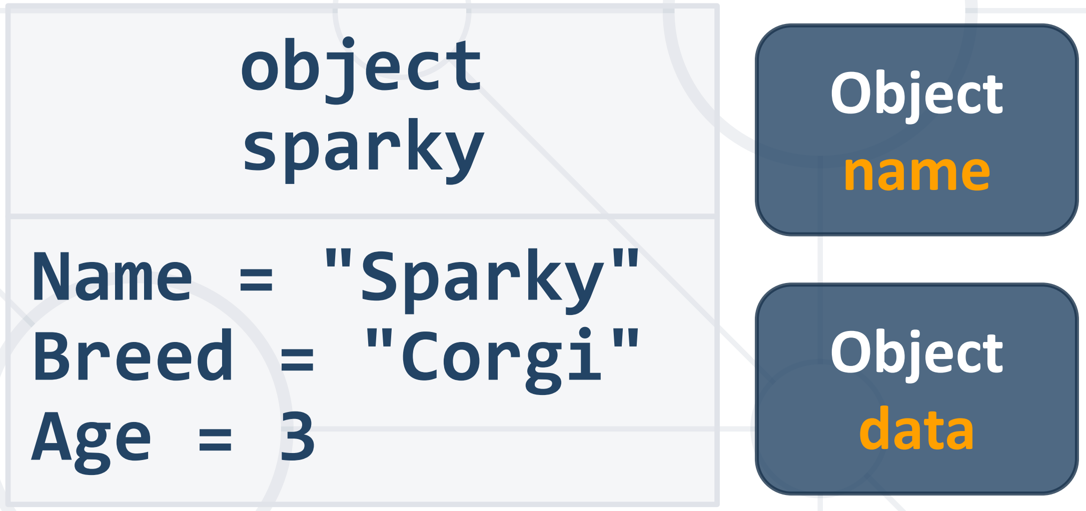


```
class Dog

Name: string
Breed: string
Age: int

Bark(…)
Eat(…)
```

- An object is a single instance of a class



```
object
sparky

Name = "Sparky"
Breed = "Corgi"
Age = 3
```

`Math.Max()`

Using the Built-in API Classes

Built-in API Classes in .NET Core

- .NET Core provides thousands of ready-to-use classes
  - Packaged into namespaces like **System**, **System.Text**, **System.Collections**, **System.Linq**, **System.Net**, etc.
- Using static .NET class members

```
double cosine = Math.Cos(Math.PI);
```

- Using non-static .NET classes

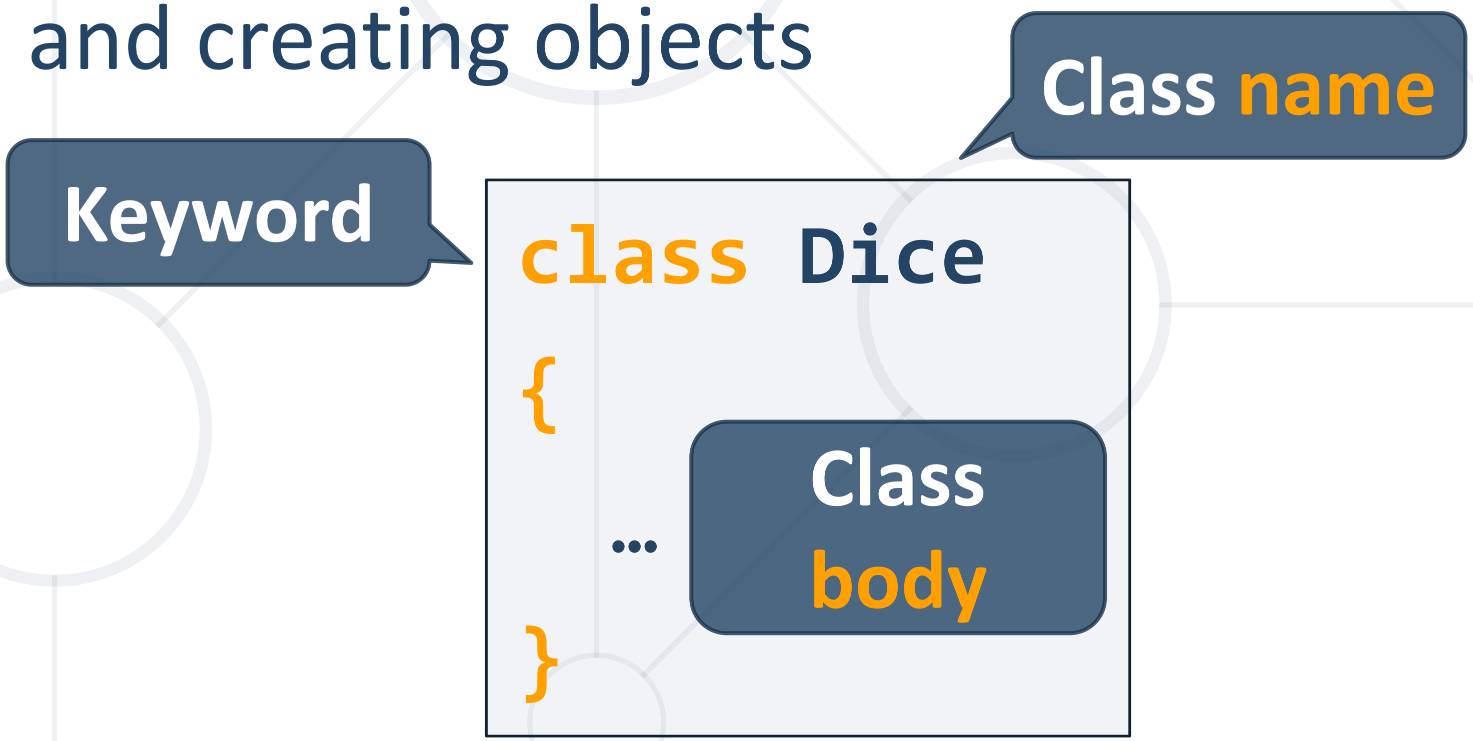
```
Random rnd = new Random();
int randomNumber = rnd.Next(1, 99);
```

Creating Custom Classes

Defining Classes

Defining Simple Classes

- Specification of a given type of objects from the real-world
- Classes provide structure for describing and creating objects



```
class Dice
{
      …
}
```

Naming Classes

- Use **PascalCase** naming
- Use descriptive nouns
- Avoid abbreviations

✓

```
class Dice { … }
class BankAccount { … }
class IntegerCalculator { … }
```

✗

```
class TPMF { … }
class bankaccount { … }
class intcalc { … }
```

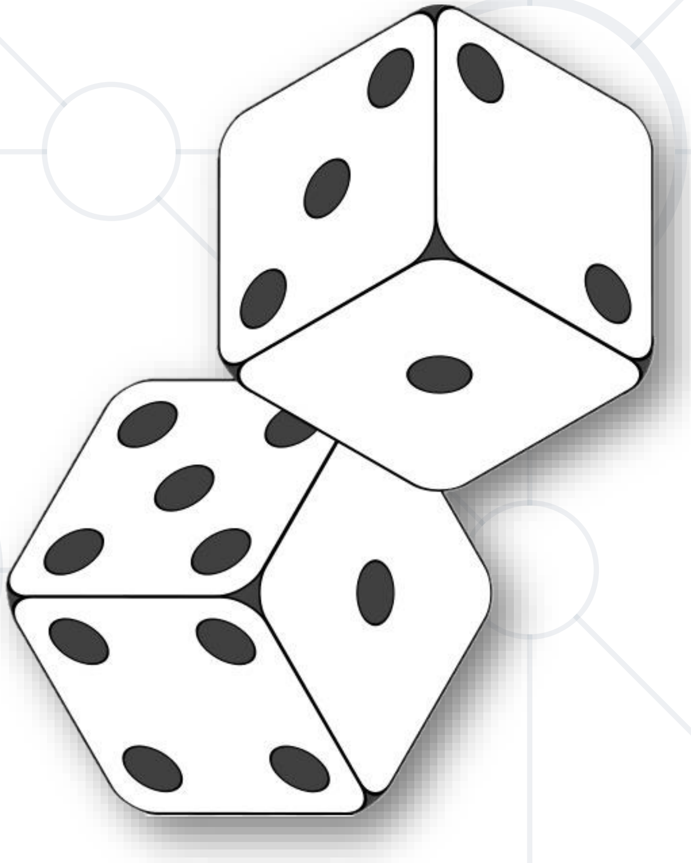
Class Members

- Class is made up of **state** and **behaviour**
- Properties **store state**
- Methods **describe behaviour**

```
class Dice
{
  public int Sides { get; set; }
  public string Type { get; set; }
  public void Roll() { }
}
```

Properties

Method



Creating an Object

- A class can have **many instances** (objects)

```
class Program
{
  public static void Main()
  {
    Dice diceD6 = new Dice();
    Dice diceD8 = new Dice();
  }
}
```

Use the **new** keyword

Properties

- Describe the characteristics of a given class

```
class Student
{
  public string FirstName { get; set;
  public string LastName { get; set; }
  public int Age { get; set; }
}
```

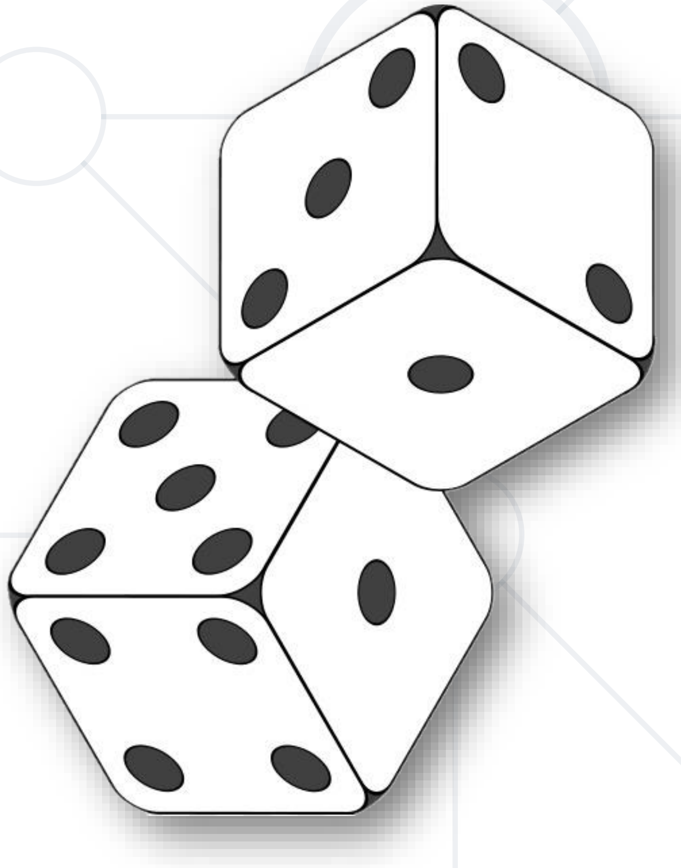
The **getter** provides access to the field

The **setter** provides field change

Methods

- Store **executable code** (algorithm)

```
class Dice
{
  public int Sides { get; set; }
  public int Roll()
  {
    Random rnd = new Random();
    return rnd.Next(1, Sides + 1);
  }
}
```



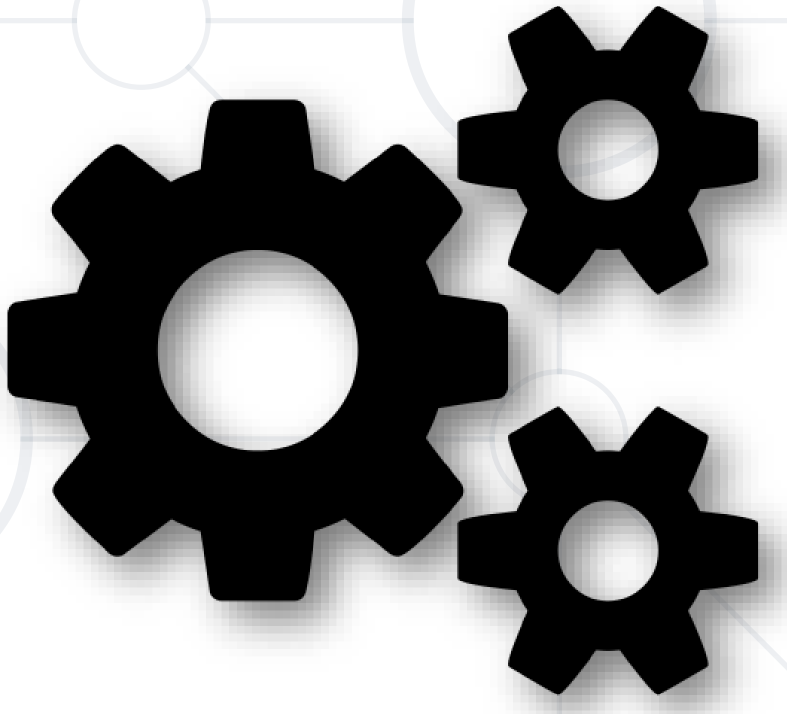
Constructors

- Special methods, executed during object creation

```
class Dice
{
    public int Sides { get; set; }
    public Dice()
    {
        this.Sides = 6;
    }
}
```

Constructor name is the same as the name of the class

Overloading default constructor



Constructors

- You can have multiple constructors in the same class

```
class Dice
{
  public int Sides { get; set; }
  public Dice() { }
  public Dice(int sides)
  {
    this.Sides = sides;
  }
}
```

```
Dice dice1 = new Dice();
Dice dice2 = new Dice(7);
```

Class Operations

- Classes can define **data** (state) and **operations** (actions)

```
class Rectangle
{
   public int Top { get; set; }
   public int Left { get; set; }
   public int Width { get; set; }
   public int Height { get; set; }

  int CalcArea()
  {
    return width * height;
  }
```

Classes may hold data (**properties**)

Classes may hold operations (**methods**)

Summary

- Objects
  - Holds a set of named values
  - Instance of a class
- Classes define templates for object
  - Methods
  - Constructors
  - Properties

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни БУДИТЕЛ

Finance Academy

SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

Решения за твоето утре

DRAFT KINGS

VIVACOM

AMBITIONED

INDEAVR

Serving the high achievers

SOFTWARE GROUP

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License