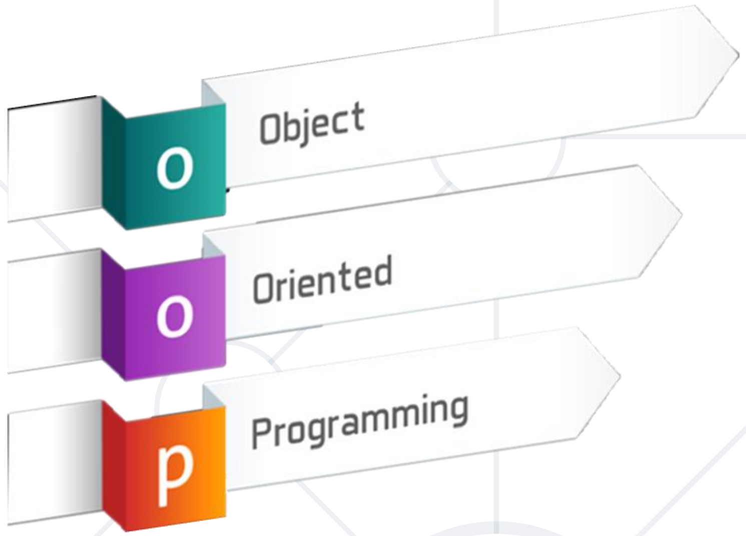
Defining Classes



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg/

Have a Question?

Software University

sli.do

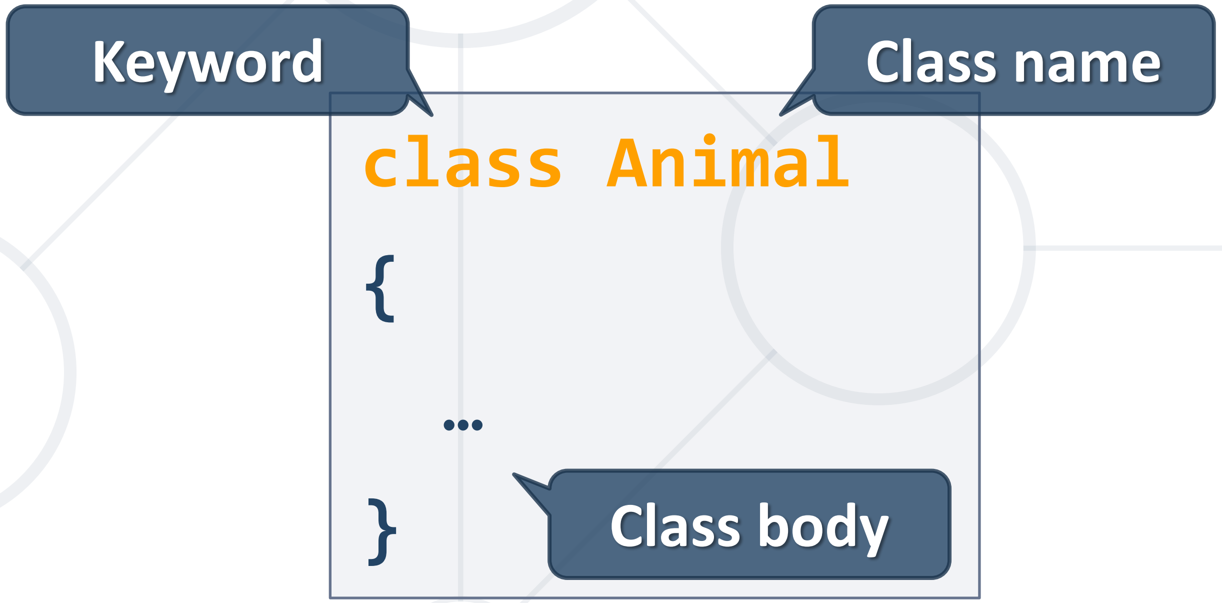
#prgm-for-qa

Table of Contents

Revision: Defining Simple Classes

Defining Simple Classes

- **Class** is a **concrete implementation** of an ADT
- Classes provide **structure** for **describing** and **creating** objects



```
class Animal
{
   …
}
```

Class Members

- **Members** are **declared** in the class and they have certain accessibility
- They can be:
  - Fields
  - Properties
  - Methods

```
class Animal
{
  int legs;
  string Name { get; }
  void Walk() { … }
}
```

Field

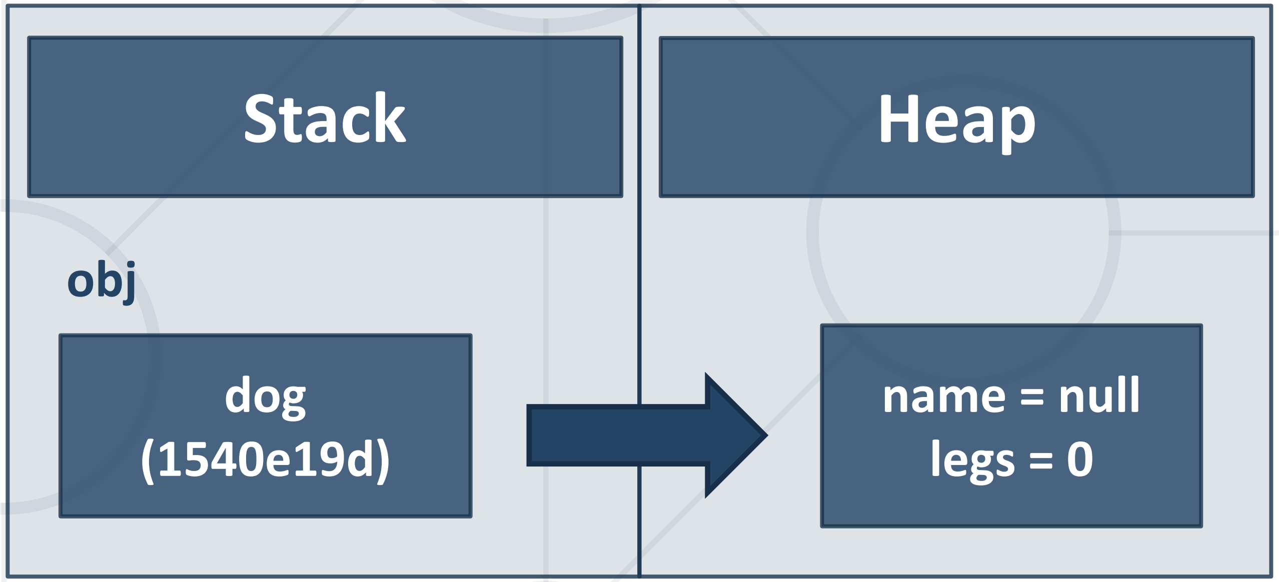
Property

Method

Object Reference

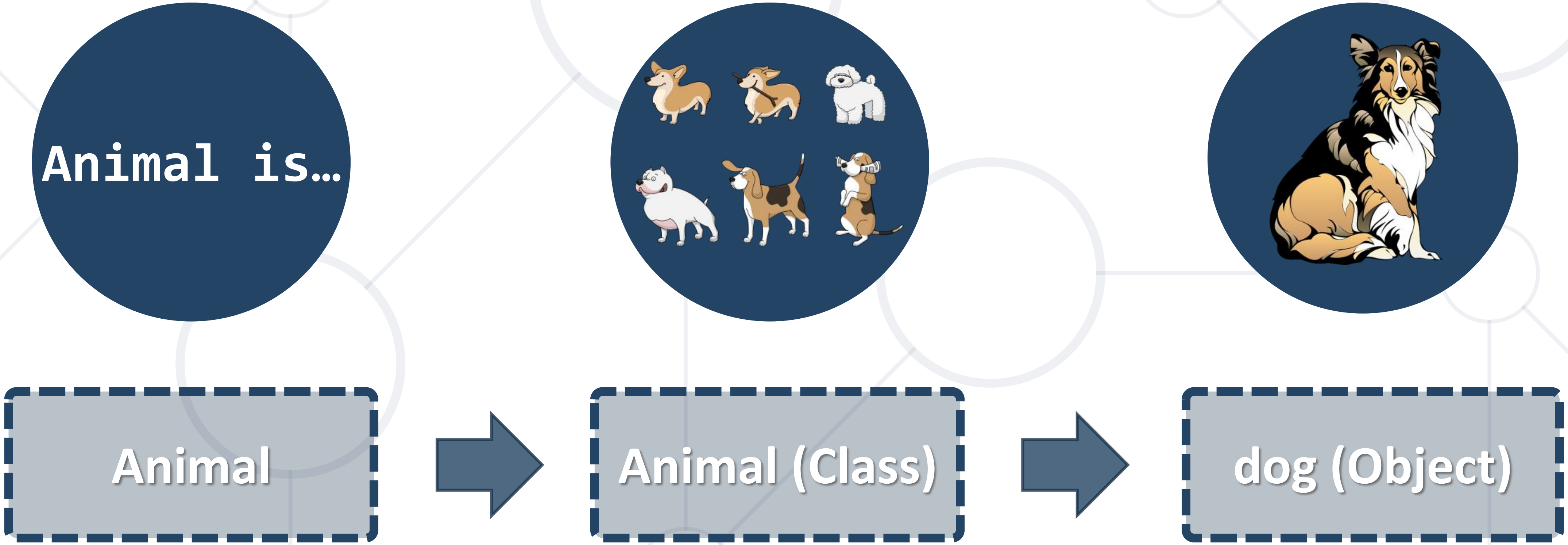
- Declaring a variable creates a **reference** in the stack
- The **new** keyword allocates memory on the heap

```
Animal dog = new Animal();
```



Classes vs. Objects

- Classes provide **structure** for describing and creating objects
- An **object** is a **single instance of a class**



Fields and Properties

Storing Data Inside a Class

Fields and Modifiers

- Class **fields** have type and name
- Access modifiers (**public** / **private**) define accessibility

Class modifier

Fields should always be private

Fields can be of **any type**

```
public class Animal
{
  private string name;
  private int legs;
  private Person owner;
  public void Walk () { … }
}
```

Properties

- Used to **create accessors** and **mutators** (**getters** and **setters**)

```
public class Animal
{
  private int legs;
  public int Legs
  {
    get { return this.legs; }
    set { this.legs = value; }
  }
}
```

The field is hidden

The getter provides access to the field

The setter provides field change

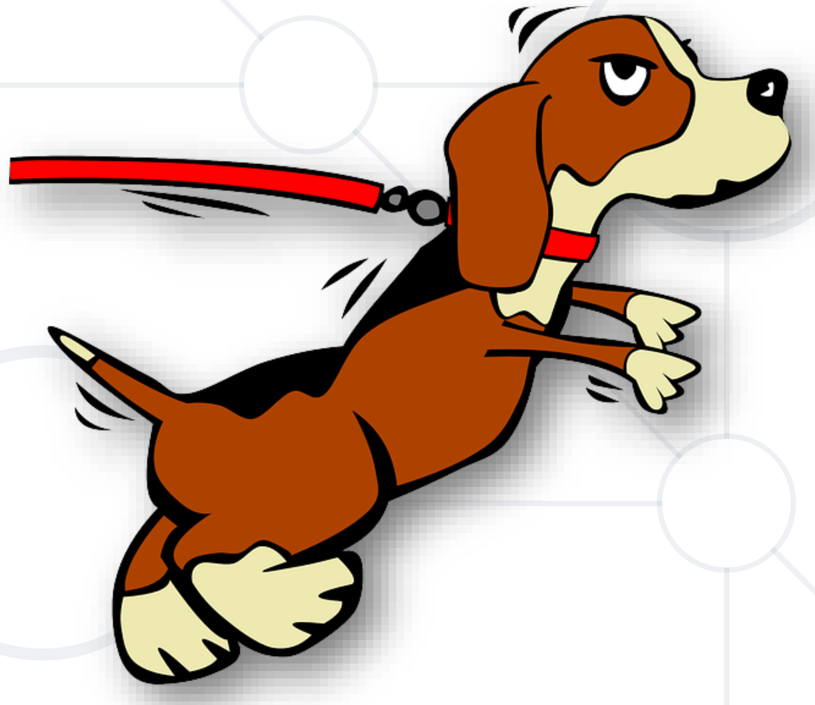
Methods

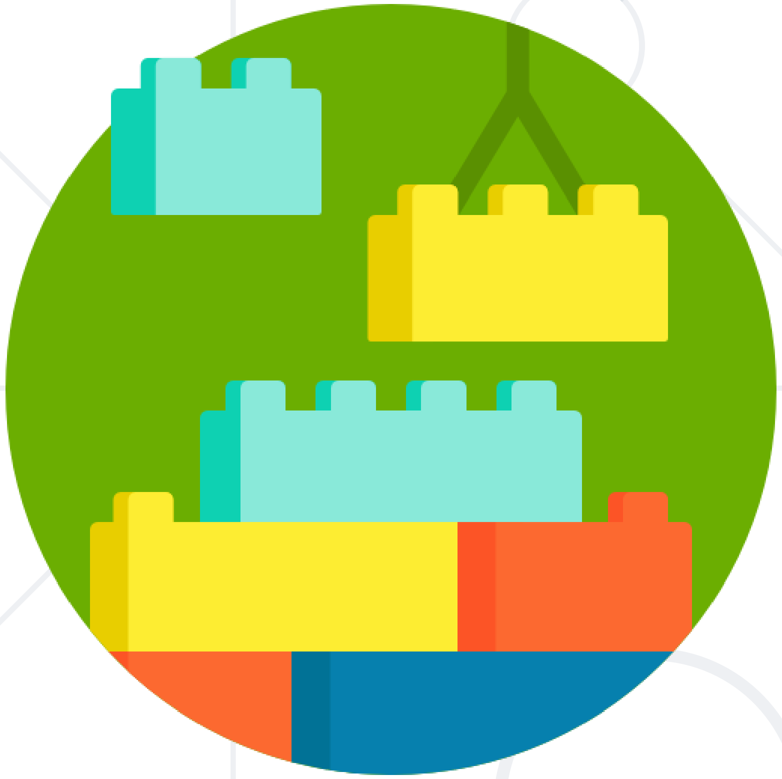
Defining a Class Behaviour

Methods

- Store executable code (an algorithm)

```
public class Animal
{
  private int legs;
  public int Walk() {
      // implement behavior
  }
}
```





Constructors

Object Initialization

Constructors

- When a **constructor** is invoked, it creates an instance of its class and usually initializes its members
- Classes in C# are instantiated with the **keyword new**

```
public class Animal
{
  public Animal() {}
}
```

```
public class StartUp
{
  static void Main()
  {
    Animal cat = new Animal();
  }
}
```

Multiple Constructors

- You can have multiple constructors in the same class

```
public class Animal
{
  private int legs;
  public Animal() { }
  public Animal(int legs)
  {
    this.legs = legs;
  }
}
```

Constructor **without** parameters

Constructor **with** parameters

Constructor Chaining

- Constructors can call each other

```
public class Person {
  private string name;
  private int age;
  public Person()
  {
    this.age = 18;
  }
  public Person(string name) : this()
  {
    this.name = name;
  }
}
```

Calls default constructor

Enumerations

Syntax and Usage

Enumerations

- **Represent** a numeric value from a fixed set as a text
- We can use them to pass **arguments** to **methods** without making code confusing

```
enum Day { Mon, Tue, Wed, Thu, Fri, Sat, Sun }
```

```
GetDailySchedule(0)
```

$\Rightarrow$

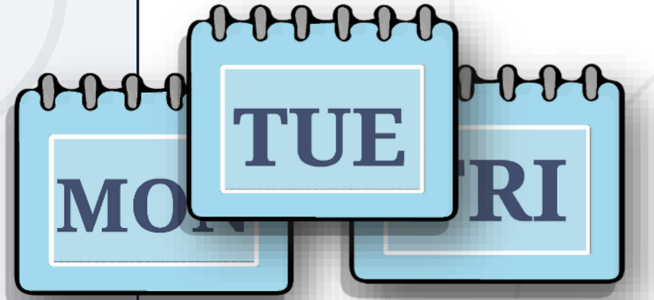
```
GetDailySchedule(Day.Mon)
```

- By default **enums** start at 0
- Every next value is incremented by 1

Enumerations

- We can **customize** enum **values**

```
enum Day {
  Mon = 1,
  Tue,   // 2
  Wed,   // 3
  Thu,   // 4
  Fri,   // 5
  Sat,   // 6
  Sun    // 7
}
```



```
enum CoffeeSize
{
  Small = 100,
  Normal = 150,
  Double = 300
}
```

Static Classes

Static Class Members

Static Class

- A **static** class is declared by the `static` keyword
- It **cannot** be **instantiated**
- You **cannot declare** variables from its **type**
- You access its **members** by using the **its name**

```
double roundedNumber = Math.Round(num);
int absoluteValue = Math.Abs(num);
int pi = Math.PI;
```

Static Members

- Both **static** and **non-static** classes can contain **static** members:
  - Methods, fields, properties, etc.
- A **static member** is **callable** on a class even when no instance of the class has been created
- Accessed by the **class'** name, not the **instance** name
- Only **one copy** of a static member **exists**, regardless of how many **instances** of the class are created

Static Members

- Static methods can be overloaded but not overridden
- A **const field** is essentially **static** in its **behavior** and it belongs to the **type**, **not** the **instance**
- Static members are initialized **before** the static member is **accessed** for the **first time** and **before** the static **constructor**

```
Bus.Drive();
int wheels = Bus.NumberOfWheels;
```

Example: Static Members

```
public class Engine
{
  public static void Run() {
    Console.WriteLine("This is a static method"); }
}
```

```
public static void Main() {
  Engine.Run();
}
// Output: This is a static method
```

Namespaces

Definition and Usage

Namespaces

Software University

- Used to organize classes
- The using keyword allows us not to write their names
- Declaring your own namespaces can help you control the scope of class and method names

```
System.Console.WriteLine("Hello world!");
List<int> list = new
  System.Collections.Generic.List<int>();
```

Summary

- Classes define **structure** for objects
- Objects are **instances of a class**
- Classes define **fields**, **methods**, **constructors** and other members
- Constructors:
  - **Invoked** when creating **new instances**
  - **Initialize** the **object's state**

Software University

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни БУДИТЕЛ

Finance Academy

SoftUni Diamond Partners

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