Defining Classes

Classes, Fields, Constructors, Properties, Methods

SoftUni Team

Technical Trainers

Software University

SoftUni Foundation

Software University

http://softuni.bg

Table of Contents

Have a Question?

SoftUni Foundation

sli.do

#csharp-advanced

Abstract Data Type

Hide Details from the Client

Abstract Data Type

- Data type whose **representation** is **hidden** from the client

```
string ADT – indexed sequence of
chars:
string()
int Length()
char CharAt(int index)
bool IsEmpty()
// many others…
```

ADTs are defined by their **usage**

Abstract Data Type (2)

- You **don't need** to know the **implementation** to use an ADT

```
Dog:
        Dog()
 string Name()
   void Bark()
   void Sleep()
```

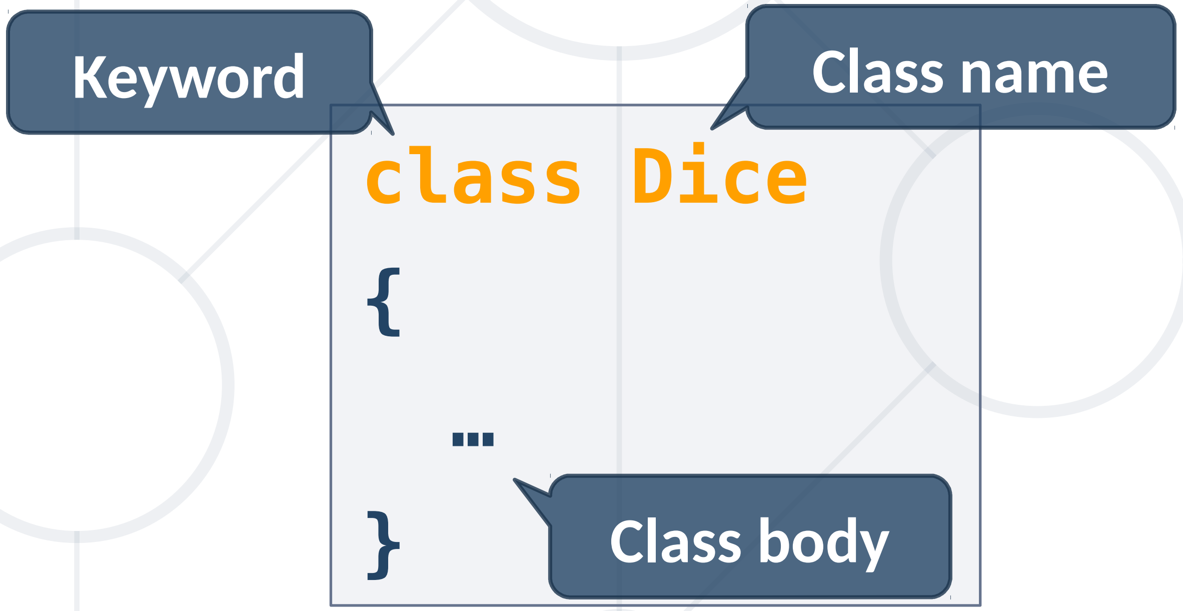
```
Computer:
        Computer()
   void TurnOn()
   void TurnOff()
 string Spec()
```

Defining Classes

Creating Class for an ADT

Defining Simple Classes

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



```
class Dice
{
    …
}
```

Naming Classes

- Name classes with nouns using **PascalCasing**
- Use **descriptive nouns**
- **Avoid abbreviations** (except widely known, e.g. URL, HTTP, etc.)

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

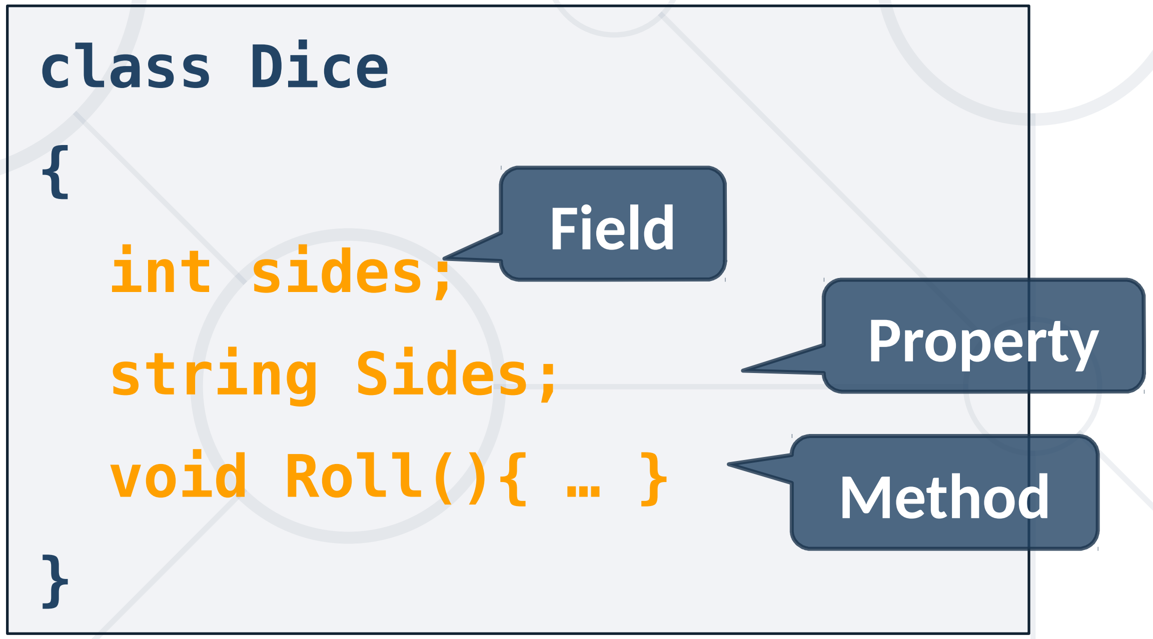
✓

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

✗

Class Members

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



```
class Dice
{
  int sides;
  string Sides;
  void Roll(){ … }
}
```

Creating an Object

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

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

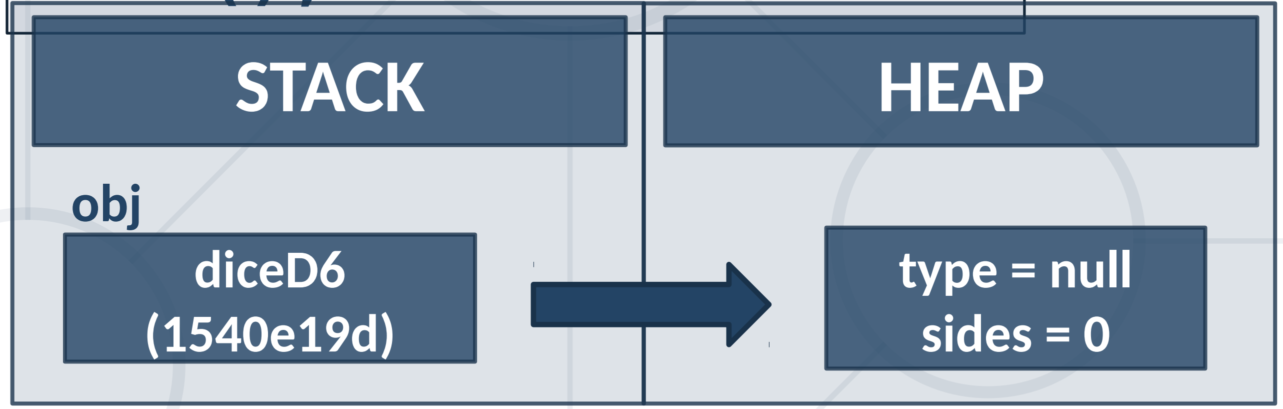
Use the **new** keyword

A variable stores a **reference**

Object Reference

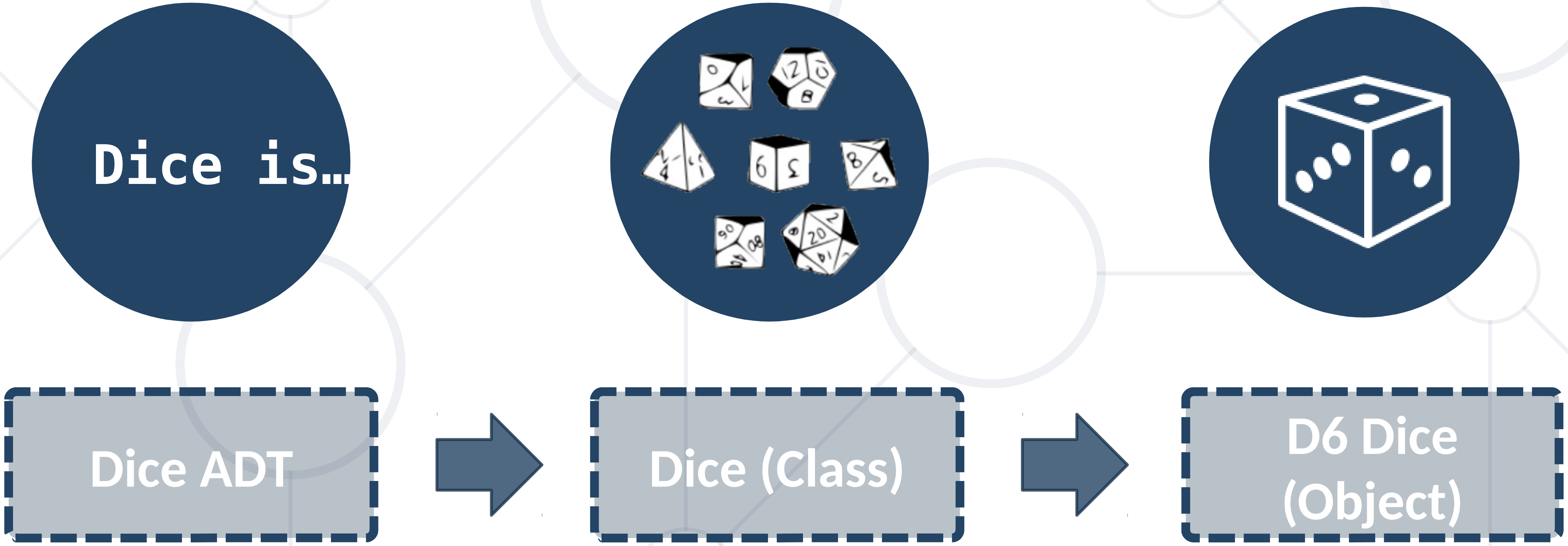
- Declaring a variable creates a **reference** in the stack
- The **new** keyword allocates memory on the heap

```
Dice diceD6 = new
Dice();
```



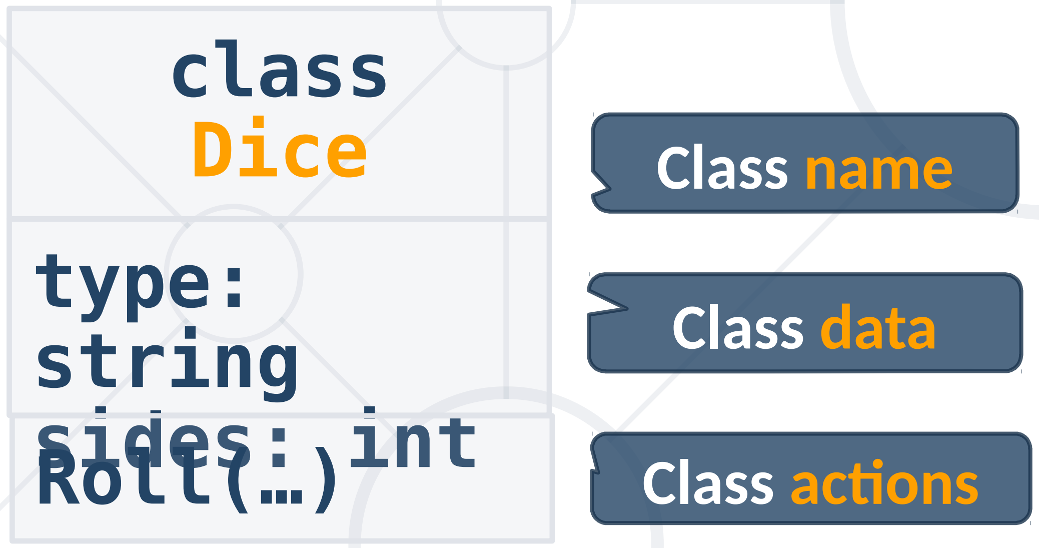
Classes vs. Objects

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

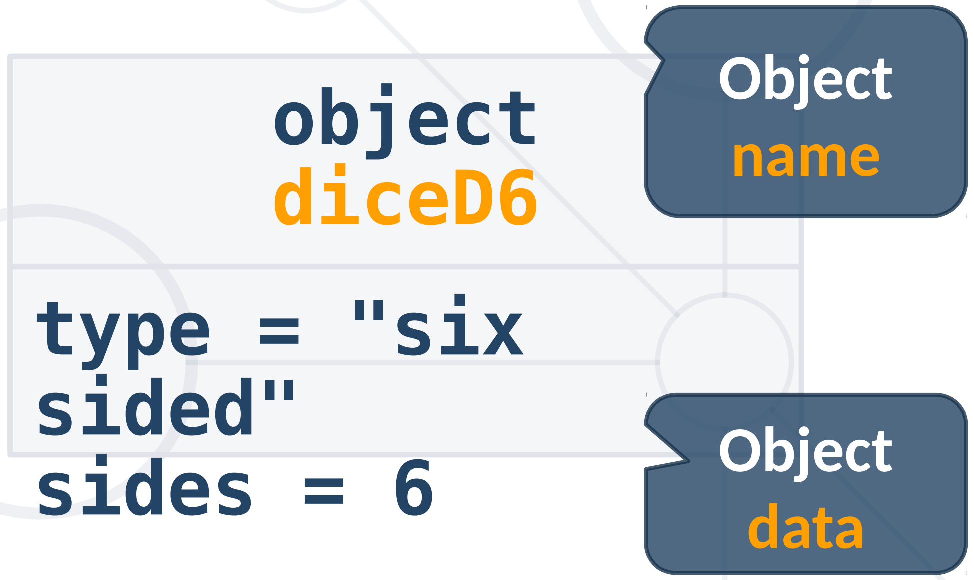


Classes vs. Objects

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



```
class
Dice
type: string
sides: int
Roll(…)
```



```
object
diceD6
type = "six sided"
sides = 6
```

Class Data

Storing Data Inside a Class

Fields and Modifiers

- Class fields have type and name
- Modifiers define accessibility

Class modifier

Fields should always be private

Fields can be of any type

```
public class Dice
{
  private string type;
  private int sides;
  private int[] rollFre-
quency;
  private Person owner;
  public void Roll () { … }
}
```

Properties

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

```
public class Dice
{
  private int sides;
  public int Sides
  {
    public get { return
this.sides; }
    public set { this.sides = value;
}
  }
}
```

The field is hidden

The getter provides access to the field

The setter provides field change

Problem: Car

- Create a class **Car**

```
Car
-make:string
-model:string
-year:int
(no actions)
```

```
private string make;
private string model;
private int year
public string Make
{
  get { return this.make; }
  set { this.make = value; }
}
// TODO: Balance and Year Getter &
Setter
```

Methods

Defining a Class Behaviour

Methods

- Store **executable code** (an algorithm)

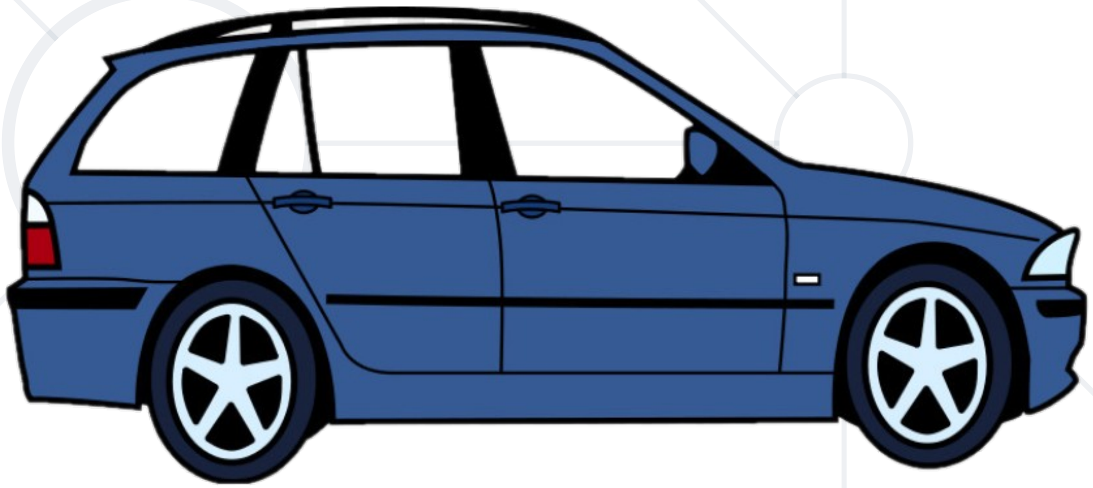
```
public class Dice
{
   private int sides;
   private Random rnd = new Random();
   public int Roll() {
      int rollResult = rnd.Next(1, this.sides +
1);
      return rollResult;
   }
}
```

this points to the current instance

Problem: Car Extension

- Create a class **Car**

```
Car
-make:string
-model:string
-year:int
-fuelQuantity:double
-fuelConsumption:double
+Drive(double distance):void
+WhoAmI():string
```



Solution: Car Extension

```
// TODO: Get the other fields from previous prob-
lem
private double fuelQuantity;
private double fuelConsumption;
// TODO: Get the other properties from previous
problem
public double FuelQuantity {
  get { return this.fuelQuantity; }
  set { this.fuelQuantity = value; }}
public double FuelConsumption {
  get { return this.fuelConsumption; }
  set { this.fuelConsumption = value; }}
```

Check your solution here: https://judge.softuni.bg/Contests/1478/Defining-Classes-Lab

Solution: Car Extension (2)

```
public void Drive(double distance)
{
    bool canContinue = this.FuelQuantity – (distance *
                 this.FuelConsumption) >= 0;
    if(canContionue)
        this.FuelQuantity -= distance * this.FuelConsumption;
    else
        Console.WriteLine("Not enough fuel to perform this
trip!");
}
```

Check your solution here: https://judge.softuni.bg/Contests/1478/Defining-Classes-Lab

Solution: Car Extension (3)

```
public string WhoAmI()
{
    StringBuilder sb = new StringBuilder();
    sb.AppendLine($"Make: {this.Make}");
    sb.AppendLine($"Model: {this.Model}");
    sb.AppendLine($"Year: {this.Year}");
    sb.Append($"Fuel: 
{this.FuelQuantity:F2}L");
    return sb.ToString();
}
```

Check your solution here: https://judge.softuni.bg/Contests/1478/Defining-Classes-Lab

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 Dice
{
  public Dice() {
}
}
```

```
public class StartUp
{
  static void Main()
{
  var dice = new
Dice()
}
```

Object Initial State

- Constructors **set object's initial state**

```
public class Dice
{
  int sides;
  int[] rollFrequency;
  public Dice(int sides) {
    this.sides = sides;
    this.rollFrequency = new
int[sides];
  }
}
```

Always ensure **correct state**

Multiple Constructors

- You can have multiple constructors in the same class

```java
public class Dice
{
  private int sides;
  public Dice() { }
  public Dice(int sides)
  {
    this.sides = sides;
  }
}
```

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

Problem: Car Constructors

- Extend the previous problem and **create 3 constructors**
- Default values are:
  - Make – VW
  - Model – Golf
  - Year – 2025
  - FuelQuantity = 200
  - FuelConsumption = 10

```
Car
---------------------------------------------
+Car()
+Car(string make, string
model, int year)
+Car(string make, string
model, int year, double
fuelQuantity, double
fuelConsumption)
```

Solution: Car Constructors

```
public Car() {
  this.Make = "VW";
  this.Model = "Golf";
  this.Year = 2025;
  this.FuelQuantity = 200;
  this.FuelConsumption = 10;}
  public Car(string make, string model, int year) : this()
{
      this.Make = make;
      this.Model = model;
      this.Year = year;}
```

Check your solution here: https://judge.softuni.bg/Contests/1478/Defining-Classes-Lab

Problem: Car Constructors(2)

```
public Car(string make, string model, int 
year,
double fuelQuantity, double fuelConsumption)
    : this(make, model, year)
{
    this.FuelQuantity = fuelQuantity;
    this.FuelConsumption = fuelConsumption;
}
```

Check your solution here: https://judge.softuni.bg/Contests/1478/Defining-Classes-Lab

Problem: Car Engine And Tires

- Create the two classes and extend the Car class

Engine
-horsePower:int -cubicCapacity:double
+Engine(int horsePower, double cubicCapacity)

Tire
-year:int -pressure:double
+Engine(int year, double pressure)

Car
+Car(string make, string model, int year, double fuelQuantity, double fuelConsumption, Engine engine, Tire[] tires)

Solution: Car Engine And Tires

```
private int horsePower;
private double cubicCapacity;
public Engine(int horsePower, double cubicCapac-
ity){
    this.HorsePower = horsePower;
    this.CubicCapacity = cubicCapacity;}
public int HorsePower {
    get { return this.horsePower; }
    set { this.horsePower = value; } }
public double CubicCapacity{
    get { return this.cubicCapacity; }
    set { this.cubicCapacity = value; }}
```

Check your solution here: https://judge.softuni.bg/Contests/1478/Defining-Classes-Lab

Solution: Car Engine And Tires(2)

```
private int year;
private double pressure;
public Tire(int year, double pressure){
    this.Year = year;
    this.Pressure = pressure;}
public int Year{
    get { return this.year; }
    set { this.year = value; }}
public double Pressure{
    get { return this.pressure; }
    set { this.pressure = value; }}
```

Check your solution here: https://judge.softuni.bg/Contests/1478/Defining-Classes-Lab

Solution: Car Engine And Tires(3)

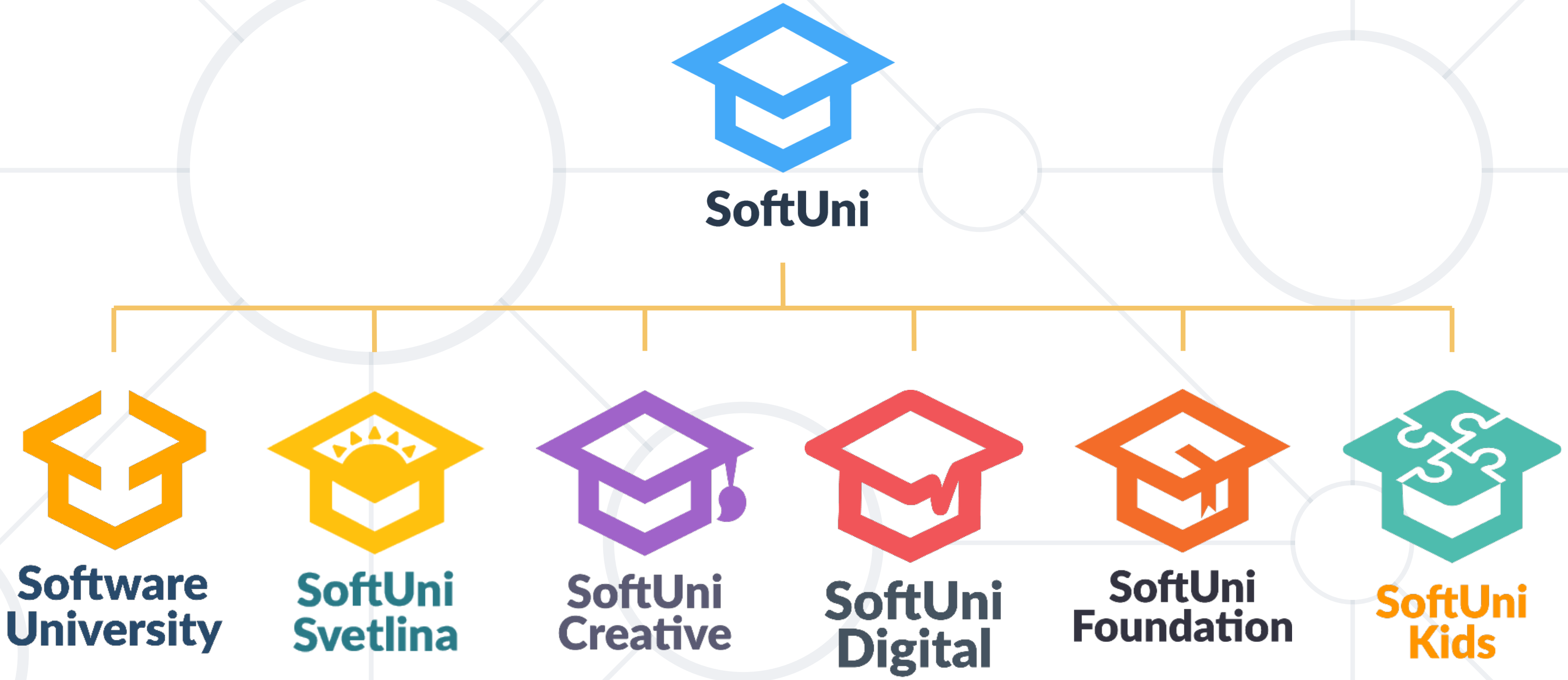
```
public Car(string make, string model, int year,
double fuelQuantity, double fuelConsumption, Engine en-
gine,
Tire[] tires)
    : this(make, model, year, fuelQuantity, fuelCon-
sumption)
{
    this.Engine = engine;
    this.Tires = tires;
}
```

Check your solution here: https://judge.softuni.bg/Contests/1478/Defining-Classes-Lab

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**

Questions?


SoftUni
Software University
SoftUni Svetlina
SoftUni Creative
SoftUni Digital
SoftUni Foundation
SoftUni Kids

SoftUni Diamond Partners

SoftUni Foundation

XSsoftware

SBTech

telenor

SoftwareGroup
doing it right

NETPEAK

SmartIT

Postbank
Решения за твоето утре

SUPER HOSTING .BG

INDEAVR
Serving the high achievers

INFRAGISTICS

STEMO
Computer Systems & Software

SoftUni Organizational Partners

SoftUni Foundation

ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

Lukanet.com

WORLD OF MYTHS

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License