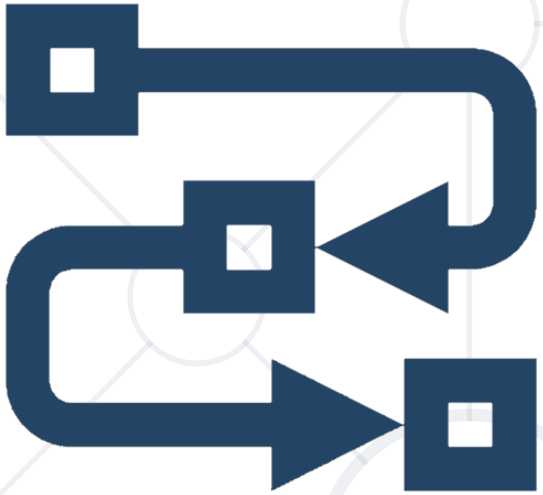
Methods

Defining and Using Methods



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

SoftUni

sli.do

#prgm-for-qa

Table of Contents

What Is a Method?

Void Methods

Simple Methods

- Named block of code, that can be invoked later
- Sample method definition:

```
static void PrintHello()
{
   Console.WriteLine("Hello!");
}
```

Method named printHello

Method body always surrounded by { }

- Invoking (calling) the method several times:

```
PrintHello();
PrintHello();
```

Why Use Methods?

- More **manageable programming**
  - Splits large problems into small pieces
  - Better organization of the program
  - Improves code readability
  - Improves code understandability
- Avoiding **repeating code**
  - Improves code maintainability
- Code **reusability**
  - Using existing methods several times

Void Type Method

- Executes the code between the brackets
- Void methods do **not** return a result

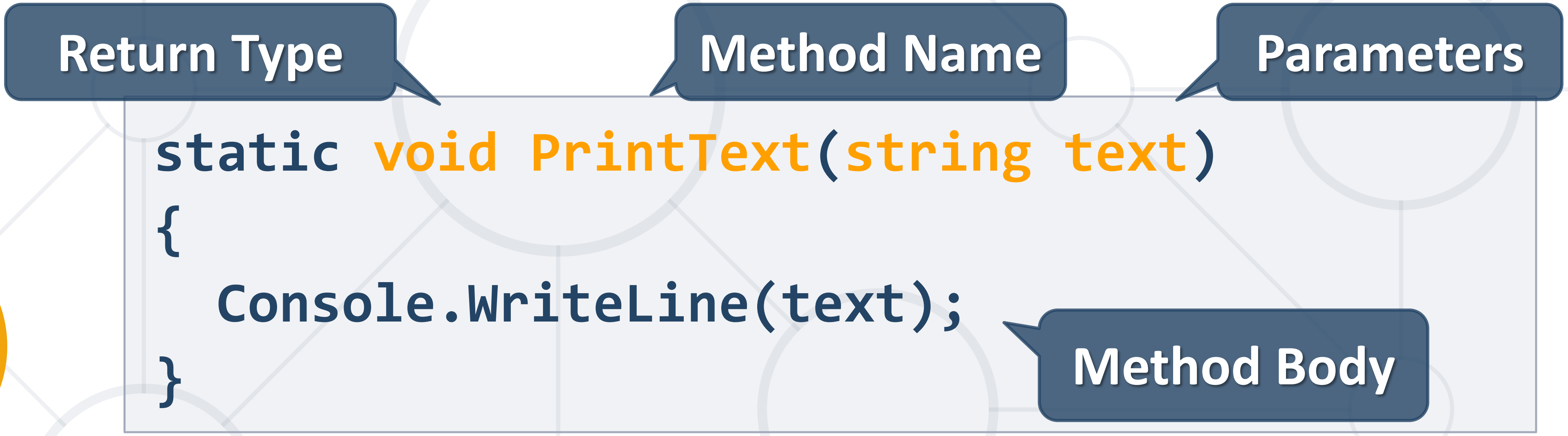
```
static void PrintHello()
{
  Console.WriteLine("Hello");
}
```

Prints "Hello" on the console

Declaring and Invoking Methods

Define Your Own Methods and Invoke Them

Declaring Methods



```
static void PrintText(string text)
{
  Console.WriteLine(text);
}
```

- Variables inside a method are **local**

Invoking a Method

- Methods are first **declared**, then **invoked** (many times)

```
static void PrintHeader()
{
  Console.WriteLine("----------");
}
```

Method **Declaration**

- **Methods** can be **invoked (called)** by their name + **( )**:

```
PrintHeader()
```

double
string
long

Methods with Parameters

Pass Input Data when Invoking Methods

Method Parameters

- Method **parameters** can be of **any data type**

```
static void PrintNumbers(int start, int end)
{
  for (int i = start; i <= end; i++)
  {
    Console.Write("{0} ", i);
  }
}
```

Multiple parameters separated by comma

- Call the method with certain values (**arguments**)

```
PrintNumbers(5, 10);
```

Passing arguments at invocation

Method Parameters

- You can pass **zero** or **several** parameters
- You can pass parameters of **different types**
- Each parameter has **name** and **type**

Multiple parameters of different types

Parameter **type**

Parameter **name**

```
static void PrintStudent(string name, int age, double grade)
{
    Console.WriteLine("Student: {0}; Age: {1}, Grade: {2}",
          name, age, grade);
}
```

Problem: Sign of Integer Number

- Create a method that prints the **sign** of an integer number **n**:

`2`	`The number 2 is positive.`
`-5`	`The number -5 is negative.`
`0`	`The number 0 is zero.`

Solution: Sign of Integer Number

```
static void PrintSign(int number)
{
  if (number > 0)
    Console.WriteLine("The number {0} is positive", number);
  else if (number < 0)
    Console.WriteLine("The number {0} is negative.", number);
  else
    Console.WriteLine("The number {0} is zero.", number);
}
```

Problem: Grades

- Write a **method** that receives a **grade** between **2.00** and **6.00** and prints the corresponding **grade in words**
  - 2.00 - 2.99 → "Fail"
  - 3.00 - 3.49 → "Average"
  - 3.50 - 4.49 → "Good"
  - 4.50 - 5.49 → "Very good"
  - 5.50 - 6.00 → "Excellent"

`3.33`	`Average`
`4.50`	`Very good`
`2.99`	`Fail`

Solution: Grades

```
double grade = double.Parse(Console.ReadLine());
PrintInWords(grade);
static void PrintInWords(double grade)
{
  string gradeInWords = "";
  if (grade >= 2 && grade <= 2.99)
    gradeInWords = "Fail";
  //TODO: make the rest
  Console.WriteLine(gradeInWords);
}
```

Problem: Printing Triangle

- Create a method for printing triangles as shown below:

```
3
```

```
1
1 2
1 2 3
1 2
1
```

```
4
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3
1 2
1
```

Solution: Printing Triangle

- Create a method that **prints a single line**, consisting of numbers from a **given start** to a **given end**:

```
static void PrintLine(int start, int end) {
  for (int i = start; i <= end; i++) {
    Console.Write(i + " ");
  }
 Console.WriteLine();
}
```

Solution: Printing Triangle

- Create a method that prints the **first half (1..n)** and then the **second half (n-1…1)** of the triangle:

```
static void PrintTriangle(int n) {
  for (int line = 1; line <= n; line++)
    PrintLine(1, line);

  for (int line = n - 1; line >= 1; line--)
    PrintLine(1, line);
}
```

Method with **parameter n**

Lines 1…n

Lines n-1…1

Returning Values From Methods

Functions: Take Input, Calculate, Return a Result

The Return Statement

- The **return** keyword immediately stops the method's execution
- Returns the specified value

```
static string ReadFullName()
{
  string firstName = Console.ReadLine();
  string lastName = Console.ReadLine();
  return firstName + " " + lastName;
}
```

Returns a **string**

- Void methods can be **terminated** by just using **return**

Using the Return Values

- Return value can be:
  - **Assigned** to a variable

```
int max = GetMax(5, 10);
```

  - **Used** in expression

```
double total = GetPrice() * quantity * 1.20;
```

  - **Passed** to another method

```
int age = int.Parse(Console.ReadLine());
```

Problem: Calculate Rectangle Area

- Create a method which returns **rectangle area** with given **width** and **length**

```
3
4
```
$\rightarrow$
```
12
```

```
6
8
```
$\rightarrow$
```
48
```

```
5
10
```
$\rightarrow$
```
50
```

```
7
8
```
$\rightarrow$
```
56
```

Solution: Calculate Rectangle Area

```
int width = int.Parse(Console.ReadLine());
int length = int.Parse(Console.ReadLine());
int area = CalcRectArea(width, length);
Console.WriteLine(area);
```

```
static int CalcRectArea(int w, int l)
{
  return w * l;
}
```

Program Execution Flow

The Call Stack: How Does It Work?

Program Execution

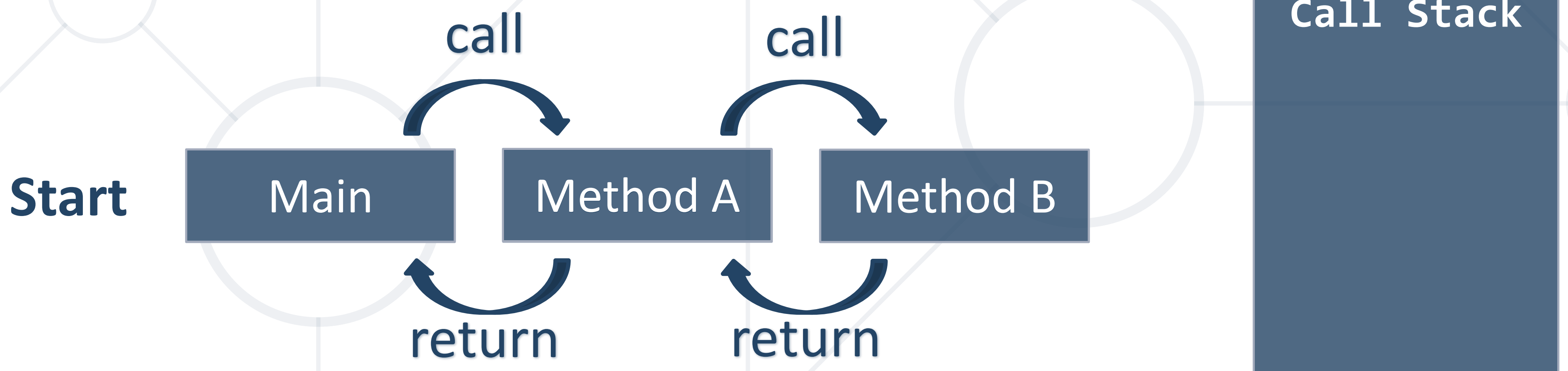
- The program continues, after a method execution completes:

```
Console.WriteLine("before method executes");
PrintLogo();
Console.WriteLine("after method executes");
```

```
static void PrintLogo()
{
   Console.WriteLine("Company Logo");
   Console.WriteLine("http://www.companywebsite.com");
}
```

Program Execution – Call Stack

- "The stack" **stores information** about the **active subroutines** (methods) of a computer program
- Keeps track of **the point** to which each active subroutine should **return control** when it **finishes executing**



Naming and Best Practices

Good Method Name Explains What It Does

Naming Methods

- Methods naming guidelines
  - Use **meaningful** method names
  - Method names should answer the question:
    - **What does this method do**?

✓
```
FindStudent, LoadReport, Sine
```

  - If you cannot find a good name for a method, think about whether it has a **clear intent**

🚫
```
Method1, doSomething, HandleStuff, sample
```

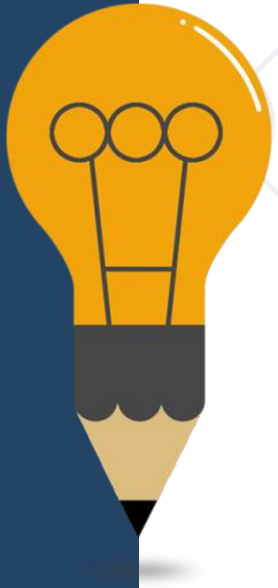
Naming Method Parameters

- Method parameters names
  - Preferred form: [**Noun**] or [**Adjective**] + [**Noun**]
  - Should be in **`camelCase`**
  - Should be **meaningful**

✓
```
firstName, report, usersList, font
```

  - Unit of measure should be obvious

✓
```
speedKmH, fontSizeInPixels, inchesLength
```



Methods – Best Practices

- Each method should perform a **single**, well-defined task
  - A method's name should **describe that task** in a clear and non-ambiguous way
- **Avoid** methods **longer than one screen**
  - **Split them** to several shorter methods

```
static void printReceipt() {
   PrintHeader();
   PrintBody();
   PrintFooter();
}
```

Self documenting and **easy to test**

Summary

- Break large programs into simple methods that solve small sub-problems
- Methods consist of declaration and body
- Methods are invoked by their name + ()
- Methods can accept parameters
- Methods can `return` a value or nothing (`void`)

Software University

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