Technische Hochschule Rosenheim
Technical University of Applied Sciences

Programming Basics – WiSe21/22

Methods

Prof. Dr. Silke Lechner-Greite

Table of contents – overall

Technische Hochschule Rosenheim

Methods

Technische Hochschule Rosenheim

- Purpose: Encapsulation of functionality for reuse within a programme
- Components of the **method signature**
  - Method name
  - List of input parameters (optional)
  - Return type
  - Visibility & modifier static (optional)
- Methods are defined **within a block**

```
public class HelloWorld {
        public static void main(String[] args) {
                System.out.println("Hello World!");
        }
}
```

Method declaration (definition) in Java

Technische Hochschule Rosenheim

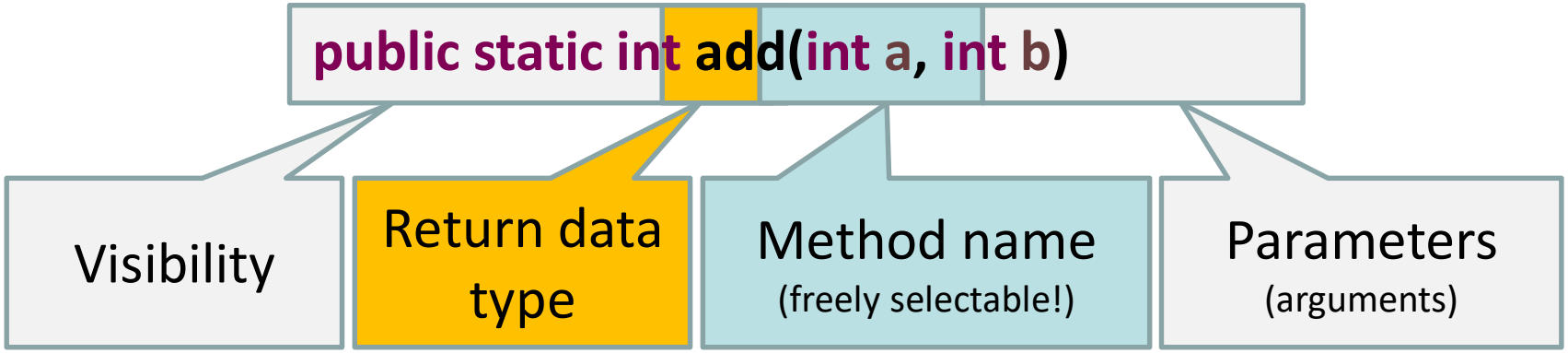
- Declaration is **always within** a class
- Static or object-bound
- Any number of statements, **always as a block**
- *When declaring:*
  return data type, name, arguments as variable declaration
- *When using:*
  name, arguments as literals, variables or expressions

More on this in the chapter on “Classes”

```
public class Addition {
        public static int add(int a, int b) {
                int sum = a + b;
                return sum;
        }
}
```

Return a value with **return**

Method signature



- Visibility: initially always `public`
- Modifier `static`: indicates a method that is bound to a class rather to an object
- Data type of the return value: e.g. `int`, `double`, `String`, etc.
- Method name: freely selectable, rules like those for variables
- Parameters (also: arguments) are…
  - Optional as needed; **no arguments = empty brackets!** ()
  - Declaration of arguments analogous to variables
  - Available as variables within the method

Calling methods

Technische Hochschule Rosenheim

```
public class Addition {
        public static int add(int a, int b) {
                int sum = a + b;
                return sum;
        }
        public static void main(String[] args) {
                int c = add(1, 2); // c == 3
                System.out.println(add(c, 4));
        }
}
```

Output?

- Method name as declared, arguments as required:
  - Literals
  - Variables
  - Complex expressions (arithmetic, other methods, …)
- Return value can be saved or used directly
  - **Exactly one return value** (literal, variable, expression), no more!

Exercise – Method declaration and calling

Technische Hochschule Rosenheim

- Live exercise
  - Complete Task 1 on the live exercises sheet “Methods”
  - You have 10 minutes.

Name conflicts and overloading

Technische Hochschule Rosenheim

```
public class Addition {
        public static int add(int a, int b) {
                int sum = a + b;
                return sum;
        }
        public static double add(double a, double b) {
                return a + b;
        }
        public static void main(String[] args) {
                int c = add(1, 2);
                System.out.println(add(c, 12.0));
        }
}
```

- **Same method name** just with **different argument list**
- Compiler selects the variant that matches the arguments when method is called

Methods without return value

Technische Hochschule Rosenheim

- There are often methods that do not have (or need) to return a value, e.g. data output
- Special pseudo data type: **void** (*i.e. returns nothing*)
- Example:

  public static void main(String [] args)
  - Starting point of the programme ("here's the start")
  - Without return (the caller is the operating system)

Methods to create structure within a program (1)

Technische Hochschule Rosenheim

```
public class Structure {
   public static void main(String[] args) {
      double wage1 = 15.0, wage2 = 17.50, factor = 1.25;
      int month = 10, day = 3, hour = 10;
      double salary;
      if (month == 1 && (day == 1 || day == 6)
            || month == 10 && day == 3
            || month == 12 && (day == 6 || day == 25 || day == 26)) {
         if (hour > 8.0)
            salary = wage2 * 8.0 + factor * (hour - 8.0);
         else
            salary = wage2 * hour;
      } else {
         if (hour > 8.0)
            salary = wage1 * 8.0 + factor * (hour - 8.0);
         else
            salary = wage1 * hour;
      }
      System.out.println("Your salaray in EUR " + salary);
   }
}
```

Complex test on public holidays!

Almost identical statements!

Methods to create structure within a program (2)

Technische Hochschule Rosenheim

```
public class Structure {

   public static boolean isPublicHoliday(int m, int d) {
      return (m == 1 && (d == 1 || d == 6)
            || m == 10 && d == 3
            || m == 12 && (d == 6 || d == 25 || d == 26));
   }

   public static double calculateEarnings(double h, double w, double f) {
      if (h <= 8.0)
         return w * h;
      else
         return w * (8.0 + f * (h - 8.0));
   }

   public static void main(String[] args) {
      double wage1 = 15.00, wage2 = 17.50, factor = 1.25;
      int month = 10, day = 3, hours = 10;
      double earnings;
      if (!isPublicHoliday(month, day))
         earnings = calculateEarnings(hours, wage1, factor);
      else
         earnings = calculateEarnings(hours, wage2, factor);
      System.out.println("Your earnings are EUR " + earnings);
   }
}
```

Public holiday logic

Wage calculation

"Actual" programme

Increase re-usability!

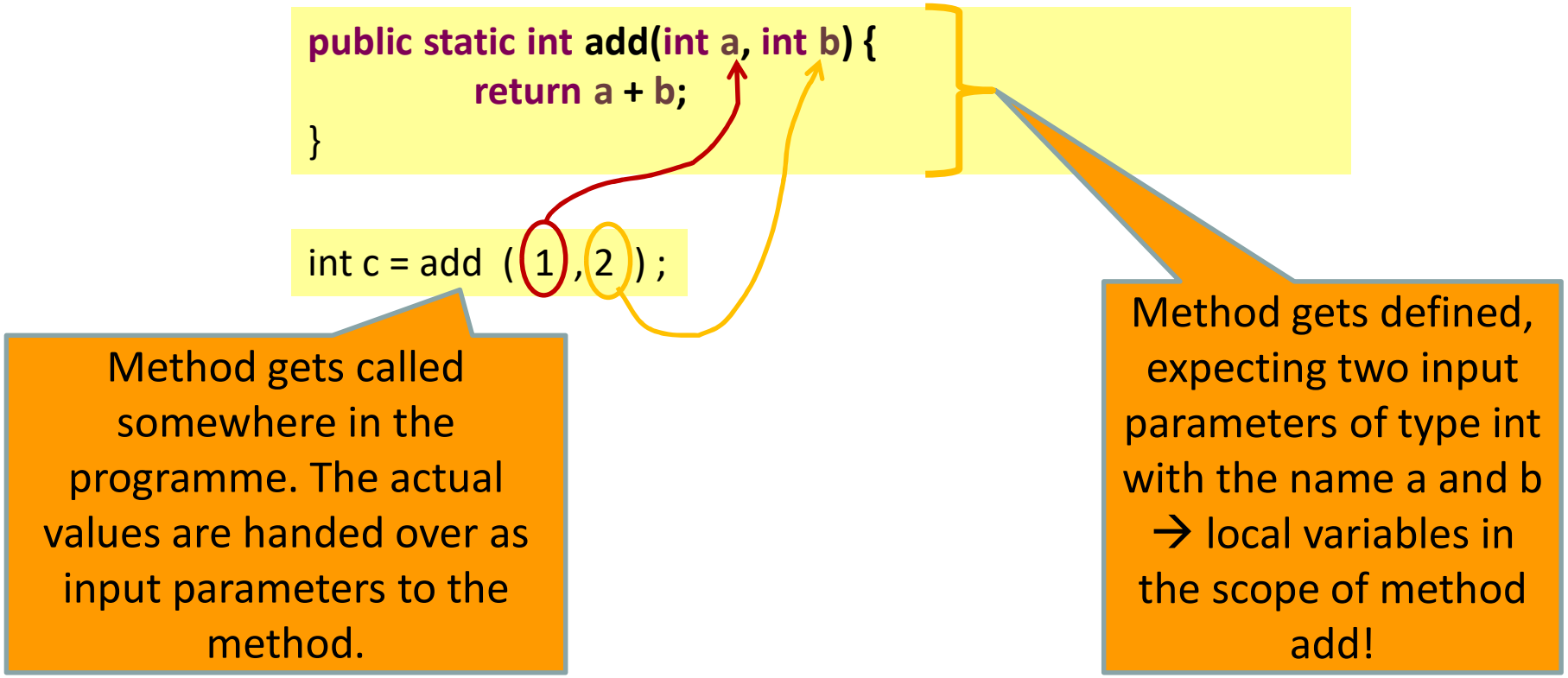
Exercise – Method declaration and calling

Technische Hochschule Rosenheim

- Live exercise
  - Complete Task 2 on the live exercises sheet “Methods”
  - You have 15 minutes.

Methods: passing parameters (1)

- When the method is called, the arguments are **copied sequentially (and by value):**

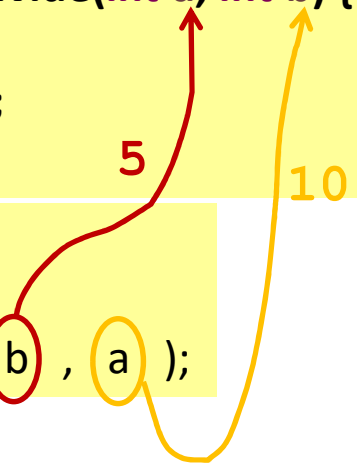


Methods: passing parameters (2)

- **Names are irrelevant for method calls with variables!**
- The values are only copied sequentially
- This is due to the work with scopes

```
public static double divide(int a, int b) {
        // 5 / 10
        return a / b;
}
```

```
int a = 10;
int b = 5;
double c = divide ( b , a );
```



Value of c?

c == 0.5

When the method is called, the arguments are copied into the method sequentially!

Exercise – Method declaration and calling

Technische Hochschule Rosenheim

- Live exercise
  - Complete Task 3 on the live exercises sheet “Methods”
  - You have 10 minutes.

Outlook: recursion

Technische Hochschule Rosenheim

- Methods can call other methods
  - For example: **main → add**
- Methods can also call themselves: **recursion** or **recursive calling**.
- Many problems can be described more easily *recursively* than *iteratively* (i.e. with **for** or **while**)

```
public class GGT {

   public static int ggT(int a, int b) {
      if (a == b) return a;
      else {
         if (a > b) return (ggT(a - b, b));
         else return (ggT(b - a, a));
      }
   }

   public static void main(String[] args) {
      int a = 143, b = 65;
      System.out.println(ggT(a, b));
   }
}
```

Methods – summary (1)

Technische Hochschule Rosenheim

- Methods are used to bundle functionality so that the functionality can be reused
- Bundling code into methods also reduces code redundancies --- error susceptibility decreases
- Consequently, maintainability is increased – functionality only exists in one place; in the event of changes, only this one place must be modified and tested.
- Methods create structure in the programme --- everything becomes clearer and easier to read
- **Valid for static methods**:
  - If a method is declared within the same class, it can be accessed with the method name.
  - If a method is declared within another class and called in its own programme, it can be accessed with the dot operator: ClassName.method(.)
    - Example: Math.random()
    - Example: Strucutre.calculateEarnings(.) --- since it was called in its own class by the main method, a call via calculateEarnings(.) is sufficient
- Methods can be overloaded.

Methods – summary (2)

Technische Hochschule Rosenheim

- Methods are declared by the method signature:
  - Methods can accept arguments
    - If the method signature contains a list of arguments, then precisely these arguments must be passed
  - Methods usually have a return value of a specific type, otherwise: **void**
    - Return for example a calculated value with **return** statement
- Methods with the **same name** must have **different arguments**.
- Random name overlapping of arguments after signature and variables when calling are **irrelevant**!
  - The arguments are copied sequentially by the call into the method
- Methods can call methods, even themselves!