Methods – So far

Methods terminology

Produced by:

Dr. Siobhán Drohan
Mr. Colm Dunphy
Ms Mairead Meagher
Ms Siobhan Roche

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

Department of Computing and Mathematics
setu.ie

Why do we need methods?

- At times, a certain portion of code has to be used many times.
- Instead of re-writing the code many times, it is better to put them into a "*subroutine*", and "call" this "*subroutine*" many time - for ease of maintenance and understanding.
- Subroutine is called method (in Java) or function (in C/C++).

What are methods?

- A **method** is a reusable block of code that performs a specific task.
- It helps organise a program by breaking it into smaller, manageable parts.
- Methods can be called whenever needed, reducing repetition and making the code easier to read and maintain.
- Methods help us so we don’t have to write the same instructions over and over—they make our code **cleaner**, **faster**, and **more fun**!

Benefits of using methods?

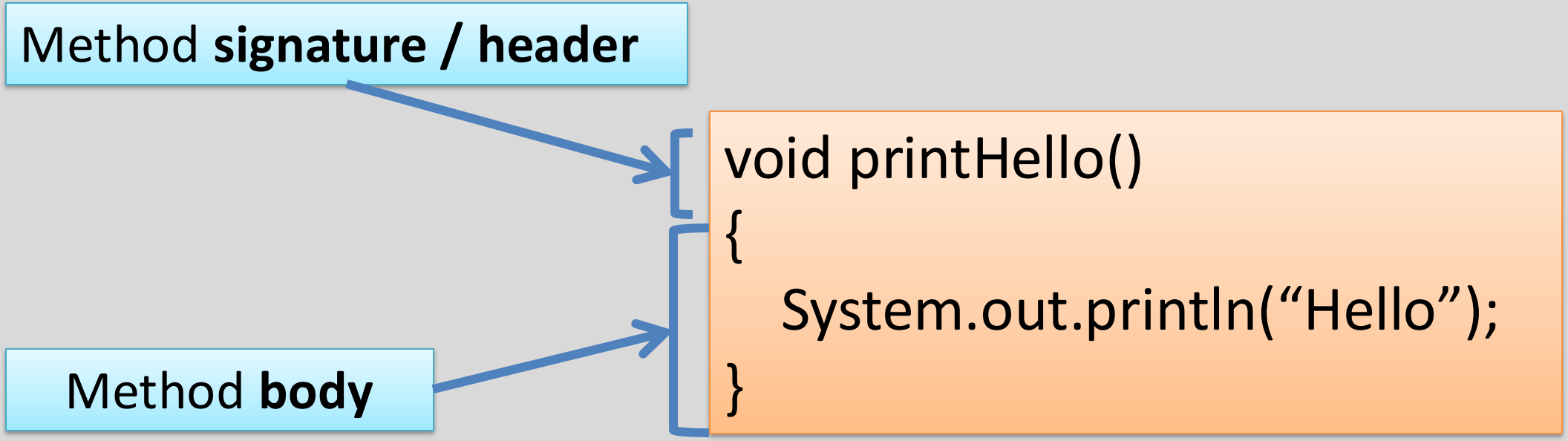
- ***Divide and conquer***: Construct the program from simple, small pieces or components. Modularise the program into self-contained tasks.
- ***Avoid repeating code***: It is easy to copy and paste, but hard to maintain and synchronise all the copies.
- ***Software Reuse***: You can reuse the methods in other programs, by packaging them.

Topics list

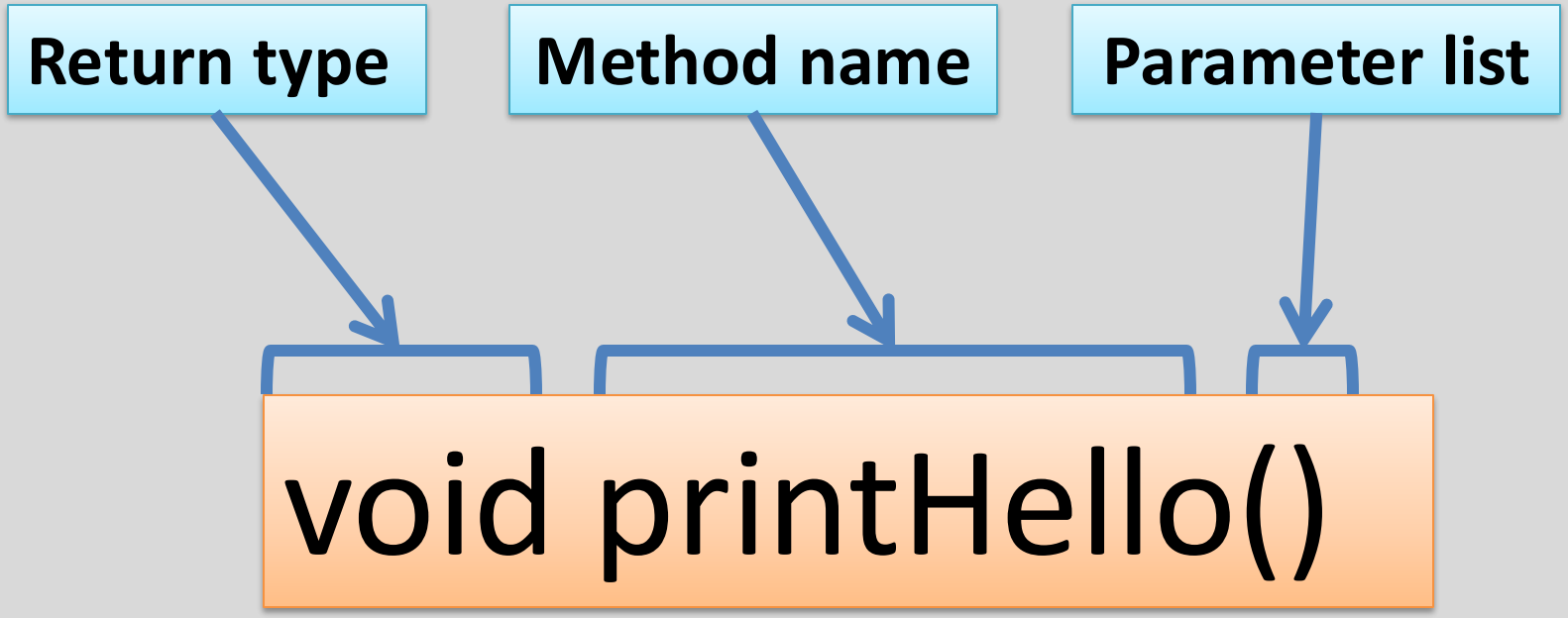
Method terminology:

1. Return type
2. Method names
3. Parameter list

Method terminology



Method signature

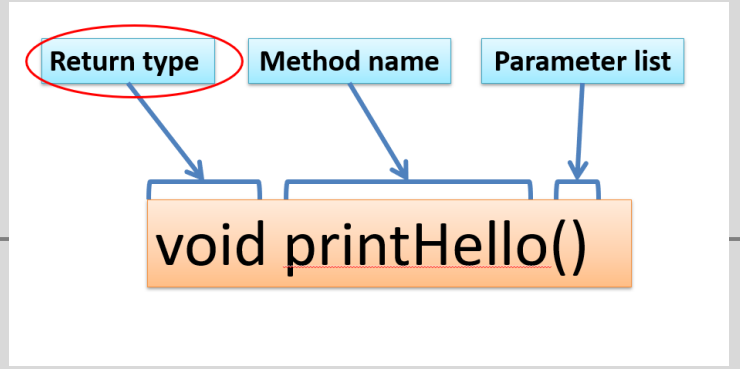


Topics list

Method terminology:

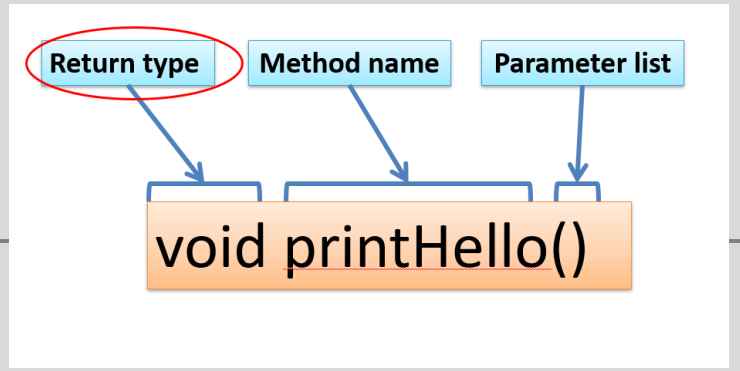
1. Return type
2. Method names
3. Parameter list

Return Type: **void**



- Methods can return information.
- The void keyword just before the method name means that **nothing is returned** from the method.
- void is a return type
  and must be included in the method signature,
  if your method returns no information.

Return Type: **int**

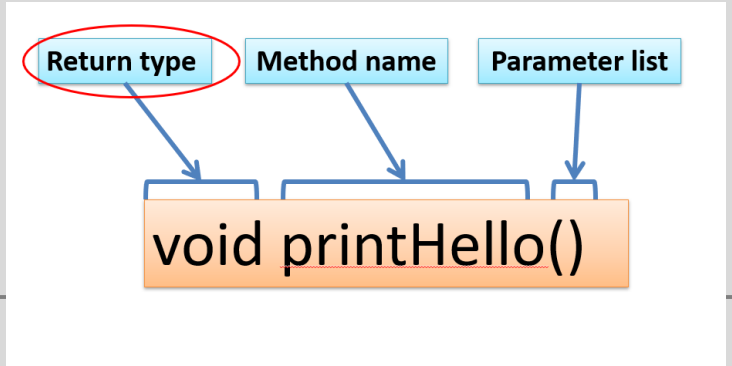


- When a data type (e.g. int)
  appears before the method name,
  this means that **something is returned** from the method.

- Within the body of the method,
  you use the **return** statement to **return the value**.

- E.g. `return(1);`
- E.g. `return(answer);`

Return Type: **int**




```
int val = 30;


void draw()
{
    int result = timesTwo(val);
    System.out.println(result);
}
```

Method **call**

```
int timesTwo(int number)
{
    number = number * 2;
    return number;
}
```

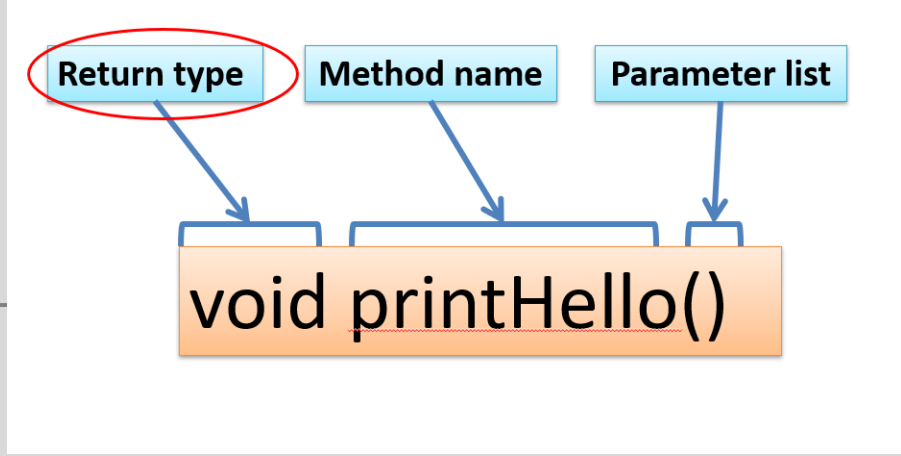
Method **header**

// The red **int** in the method declaration
// specifies the type of data to be returned.

Method **definition**

val is *passed in* becoming **number**

Return Types



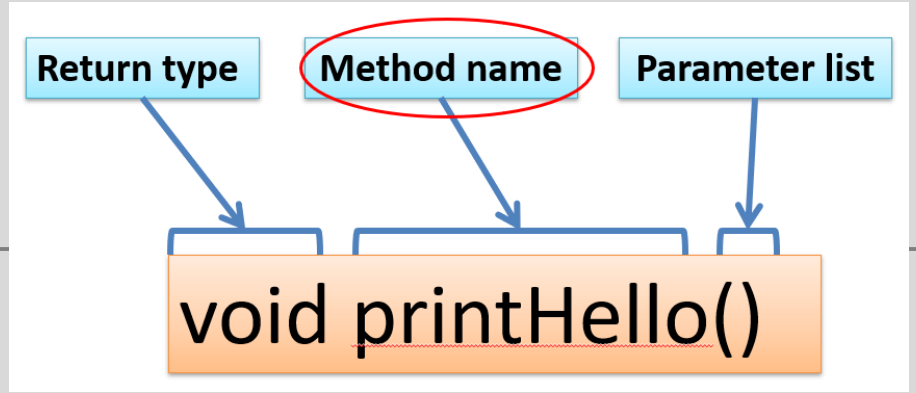
- Methods can return any **type of data** e.g.
  - boolean
  - byte
  - char
  - int
  - float
  - String
  - etc.
- You can only have **one return type, per method**.

Topics list

Method terminology:

1. Return type
2. Method names
3. Parameter list

Method name



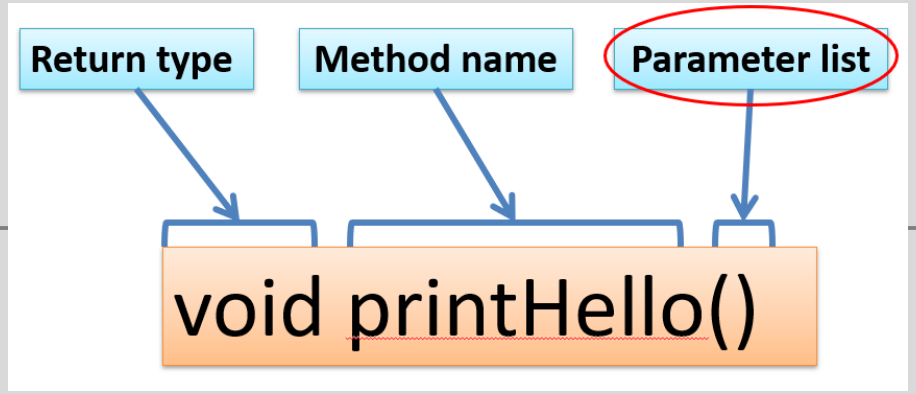
- Method names should:
  - Use **verbs** (i.e. actions) to describe what the method does
    e.g.
    - calculateTax
    - printResults
  - Be **mixed case** with the first letter lowercase
    and the first letter of each internal word capitalised.
    i.e. **camelCase**

Topics list

Method terminology:

1. Return type
2. Method names
3. Parameter list

Parameter list



- Methods **take in data** via their **parameters**.

- Methods do not have to pass parameters
  e.g. setup() has **no parameters**.

Methods **with NO parameters**

void printFirstName()
void printFullName()

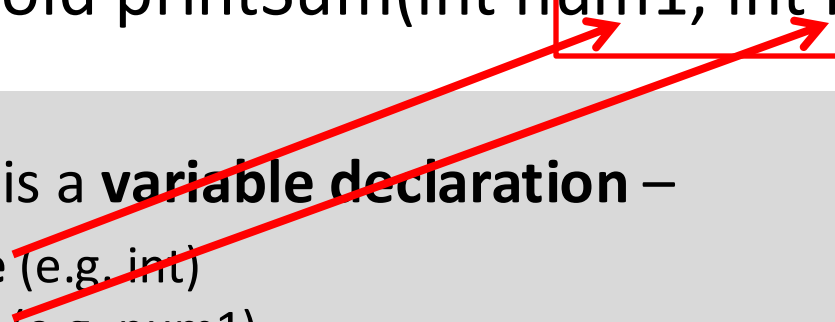
- Methods do not have to pass parameters.
- These methods have **no parameters**;
  note how no variable is passed in the parenthesis i.e. ().
- These methods don't need any additional information to do its tasks.
  - E.g. `outputHello();`

```
void outputHello(){
    println("Hello');
}
```

Methods **with Parameters**

```
void println(String myString)

void printSum(int num1, int num2)
```



- A **parameter** is a **variable declaration** –
  - it has a **type** (e.g. int)
    and a **name** (e.g. num1).
- If a method needs additional information to execute,
  we provide a parameter/s, so that the information can be passed into it.
- The first method, *println*, above has **one parameter**.
- The second method *printSum* has **two parameters**
- Methods can have **any number of parameters**

Questions?