# Local Variables and Scope of Variables


- Produced by: Ms. Mairead Meagher, Ms. Siobhán Roche.



Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

Department of Computing and Mathematics
http://www.setu.ie/

# Topics list

1. Local variables.

2. Scope of variables.

# To think about

- How could we write a method to ‘refund’ an excess balance?
  - We need to return the value in balance.
  - We need to set the balance to zero.
- How can we do the first without losing the value in balance?

# Unsuccessful attempt

```
public int refundBalance()
{
    // Return the amount left.
    return balance;
    // Clear the balance.
    balance = 0;
}
```

It looks logical, but the language does not allow it.
A method will terminate immediately after executing a return statement!

# Local variables

- Methods can define their own, *local* variables:
  - Short lived, like parameters.
  - The method sets their values – unlike parameters, they do not receive external values.
  - Used for ‘temporary’ calculation and storage.
  - They exist only as long as the method is being executed.
  - They are only accessible from within the method.
  - They are defined within a particular *scope*.

# Local variables

A local variable


```
public int refundBalance()
{
    int amountToRefund;
    amountToRefund = balance;
    balance = 0;
    return amountToRefund;
}
```

# Topics list

1. Local variables.

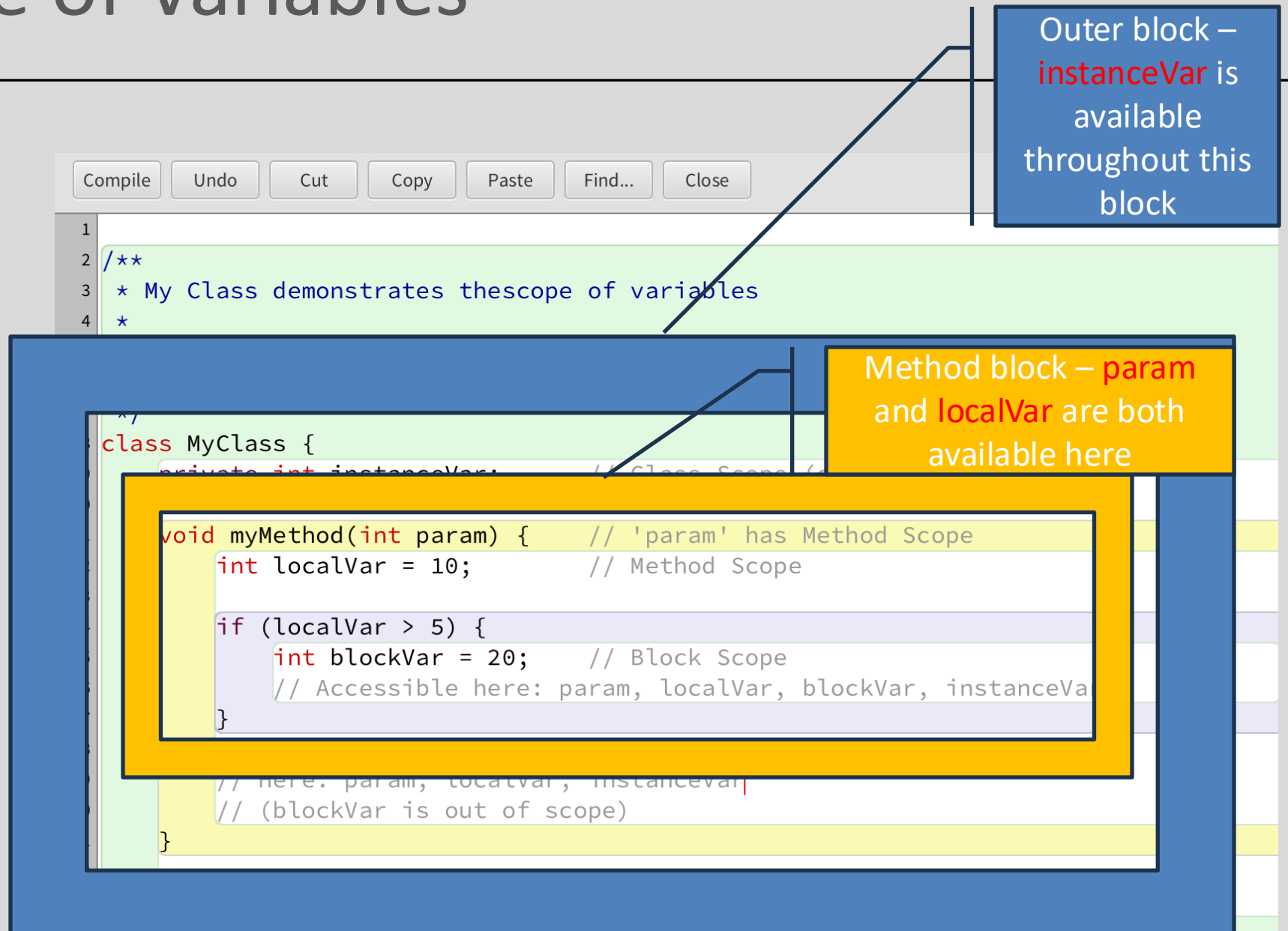
2. Scope of variables.

# Scope of Variables

Compile | Undo | Cut | Copy | Paste | Find... | Close | Source Code

```java

/**
 * My Class demonstrates thescope of variables
 *
 * @author Mairead & Siobhan
 * @version 25-09-2025
 */
class MyClass {
    private int instanceVar;       // Class Scope (object-level)

    void myMethod(int param) {     // 'param' has Method Scope
        int localVar = 10;         // Method Scope

        if (localVar > 5) {
            int blockVar = 20;     // Block Scope
            // Accessible here: param, localVar, blockVar, instanceVar
        }

        // Here: param, localVar, instanceVar
        // (blockVar is out of scope)
    }

    // Here: instanceVar
}

```

# Scope of Variables



# Scope and lifetime

- Each block defines a new scope.
  - Class, method and statement.
- Scopes may be nested:
  - statement block inside another block inside a method body inside a class body.
- Scope is static (textual).
- Lifetime is dynamic (runtime).

# Scope and lifetime

- The scope of a field is its whole class.
- The lifetime of a field is the lifetime of its containing object.
- The scope of a local variable is the block in which it is declared.
- The lifetime of a local variable is the time of execution of the block in which it is declared.

# Questions?