Ulster University

COM410 Algorithms & Data Structures

A1.3 Static Variables and Methods

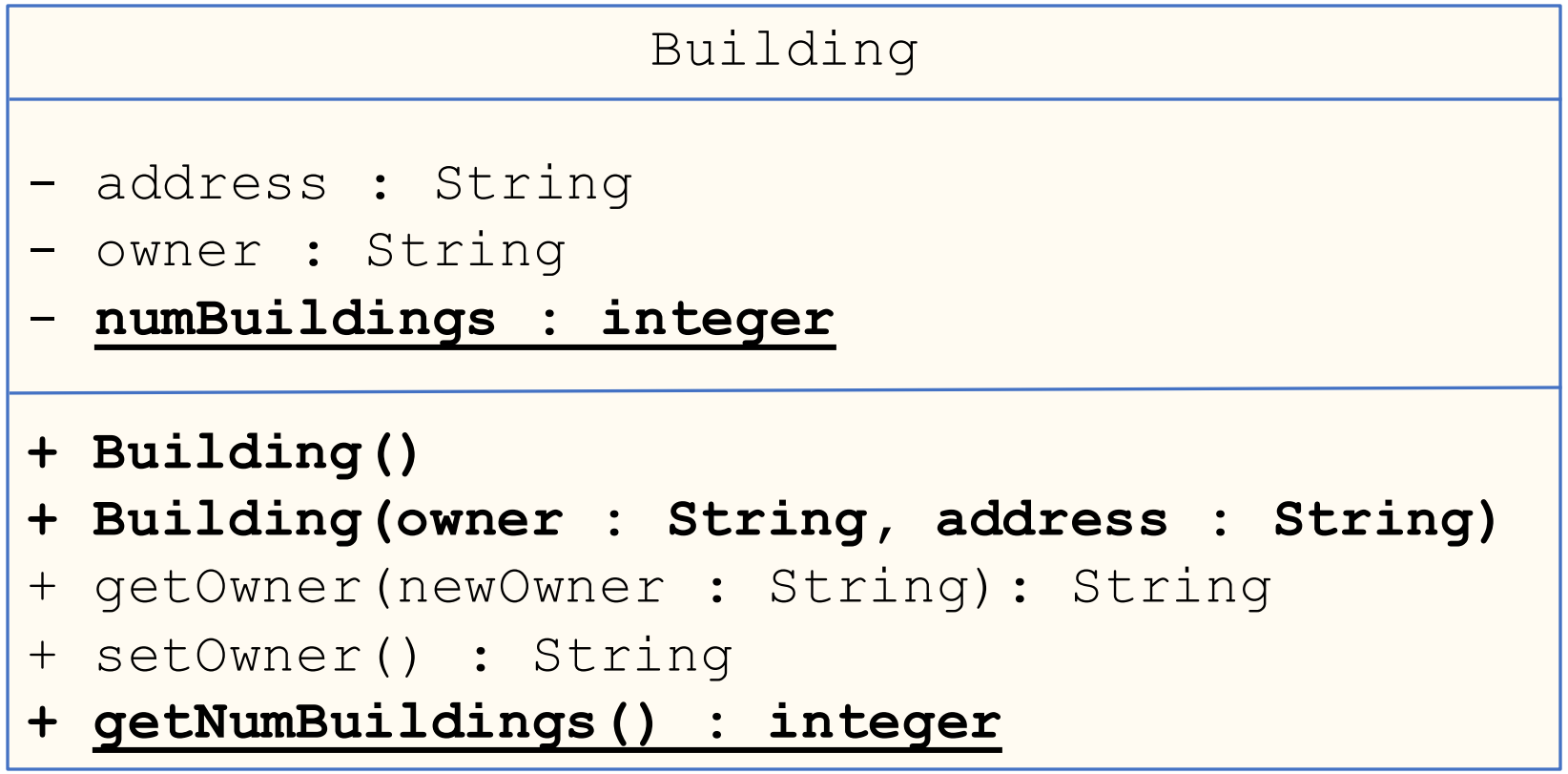
Static Variables and Methods

- So far, the variables and methods we have used have been **instance** variables and **instance** methods
  - a copy of the variable exists for each instance of the class
  - methods are applied to instances of the class

- **Static** variables and **static** methods belong to the class – not to a particular instance
  - They are often “helper” methods and associated data that provide a general service to the application
  - E.G. in our **Anytown** project, we might want to keep track of the total number of buildings that can be generated

Definition of a Building

- In UML (Universal Modelling Language)…



By convention, static variables and methods are indicated by underlining

We have also added the constructors to complete the UML diagram

Static Variables and Methods

```
public class Building {
    private String  address;
    private String owner;
    private static int numBuildings = 0;

    public static int getNumBuildings() {
        return numBuildings;
    }

   // ...
}
```

Static (class) variable

Static (class) method

Scenario

Make the following modifications to your **`Anytown`** project…

i. Add a static integer variable **`numBuildings`** to the **`Building`** class along with the static method **`getNumBuildings()`** that returns it. Also, update both constructor methods so that the value of **`numBuildings`** is incremented every time a new **`Building`** is created.

ii. Add code to the **`AnytownTest`** class so that the number of buildings created is displayed as the final output.

iii. Now, provide a similar static variable and method for each of the **`House`** and **`Shop`** classes and make appropriate modifications to their constructors.

iv. Add code to the **`AnytownTest`** class to modify the final output message so that it takes the form (e.g.)

"There are a total of 6 buildings of which 4 are houses and 2 are shops"

Next Steps

- So far, we have…
  - Identified the main concepts of OOP
  - Introduced UML as a modelling technique to help specify objects as their attributes and operations
  - Implemented our initial object hierarchy in Java
  - Built and run the application using IntelliJ IDEA

- **To go further, we need to know more Java,** in particular
  - Loop structures, conditional statements
  - Variables, data types, arrays and strings
  - Input, Output and Files
  - Generating random values (useful for testing)