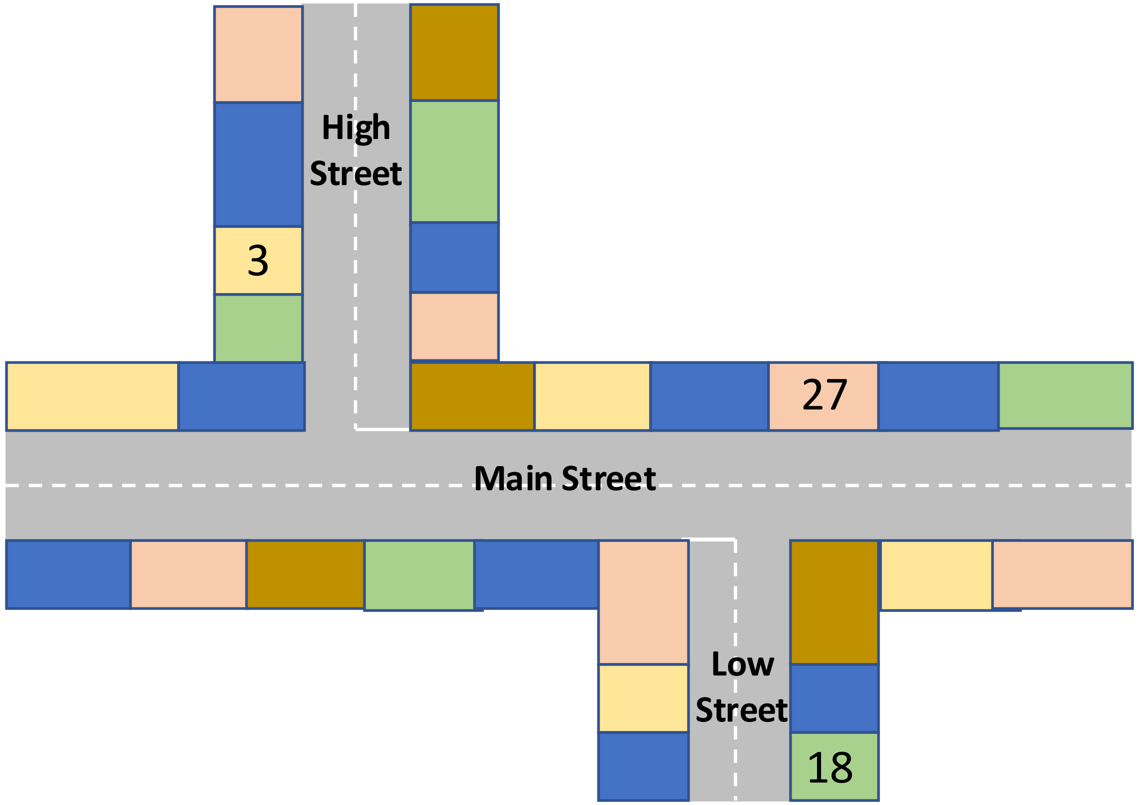
COM410 Programming in Practice

A1.1 Objects and Classes

Anytown Street Plan

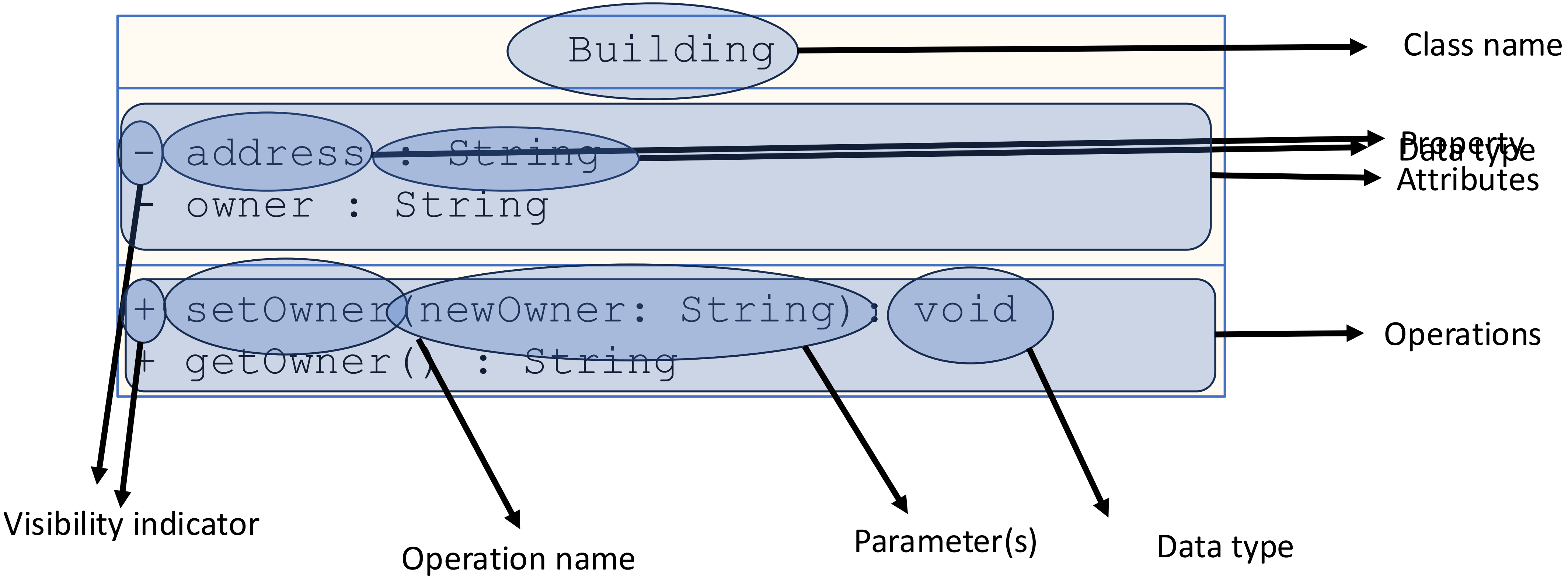


- Anytown – a small village with 3 streets of buildings where each building is described by
  - its address
  - its owner (or business name)

- 3 High Street, Smith's Newsagent
- 27 Main Street, Rex Dog Grooming
- 18 Low Street, Mary Jones

Definition of a Building

- In UML (Universal Modelling Language)…



Building

- address : String
- owner : String

+ setOwner(newOwner: String) : void
+ getOwner() : String

Class name

Property

Data type

Attributes

Operations

Visibility indicator

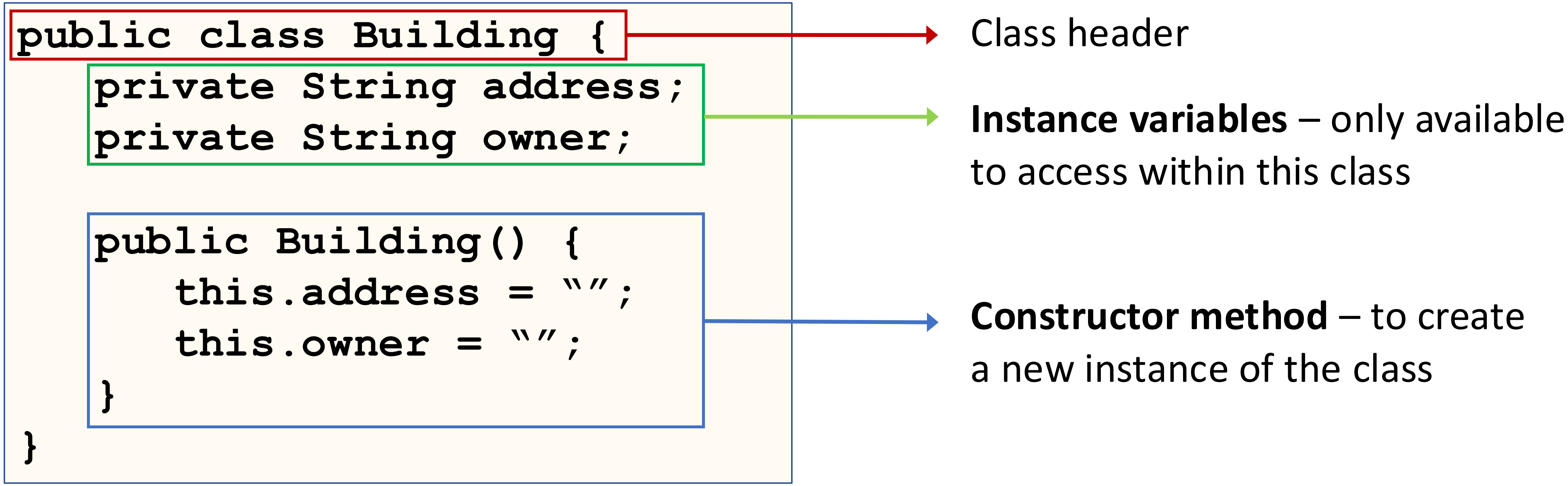
Operation name

Parameter(s)

Data type

Definition of a Building

- In Java…



```
public class Building {
    private String address;
    private String owner;

    public Building() {
        this.address = “”;
        this.owner = “”;
    }
}
```

Class header

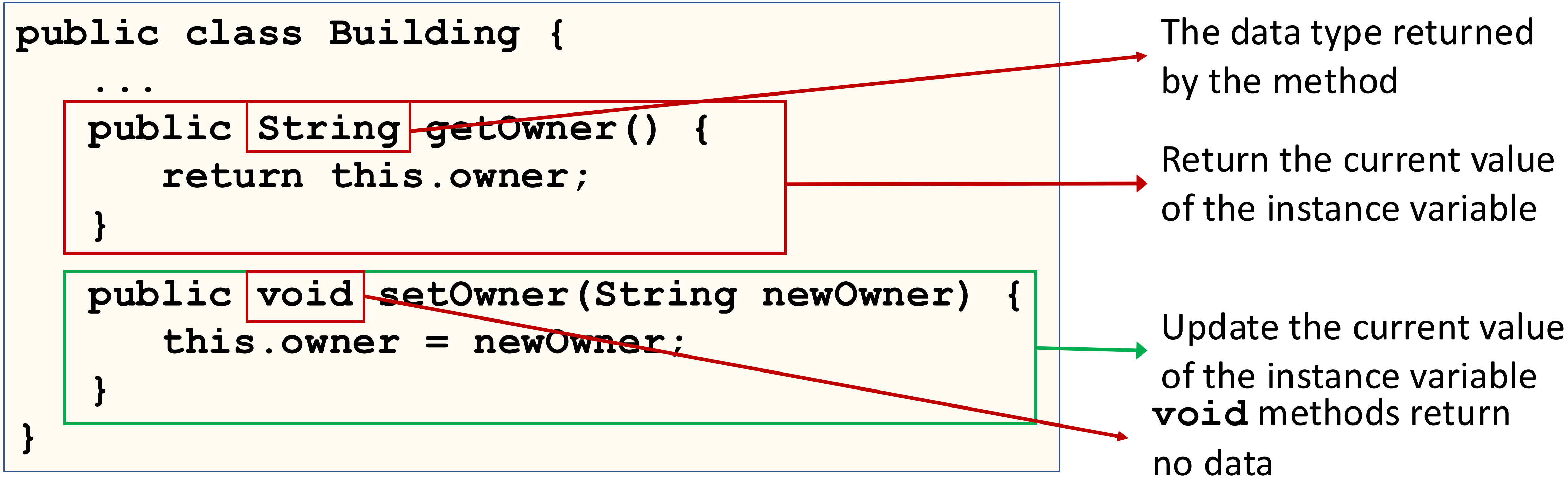
Instance variables – only available to access within this class

Constructor method – to create a new instance of the class

- Since instance variables are private – how can they be accessed from outside the class?

Accessor and Mutator Methods

- “Getter” and “Setter” methods to return and update the values of the instance variables



```
public class Building {
    ...
    public String getOwner() {
        return this.owner;
    }

    public void setOwner(String newOwner) {
        this.owner = newOwner;
    }
}
```

The data type returned by the method

Return the current value of the instance variable

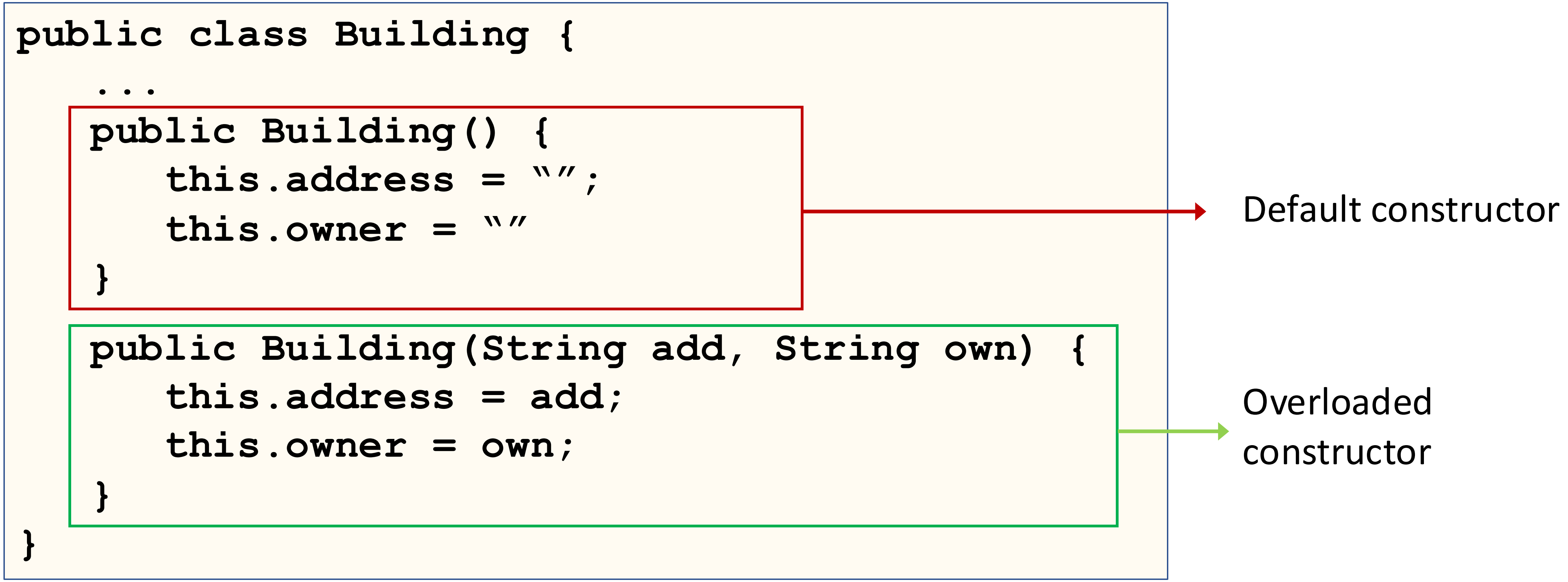
Update the current value of the instance variable

void methods return no data

- Can we provide the values when the Building is being created?

Overloading Constructors

- Can provide multiple versions of constructors (or other methods) that are differentiated by their parameter lists



```
public class Building {
    ...
    public Building() {
        this.address = “”;
        this.owner = “”
    }
    public Building(String add, String own) {
        this.address = add;
        this.owner = own;
    }
}
```

Default constructor

Overloaded constructor

toString()

- A method commonly provided to represent the state of an object as a string

```
public class Building {
    ...

    public String toString() {
        return this.address + “ occupied by “ + this.owner;
    }

}
```

Scenario

- Create a new project called **Anytown** in IntelliJ Idea and create the new file *Building.java*

  i. In the new file, implement the `Building` class with instance String variables `address` and `owner`, overloaded constructor methods, accessor and mutator methods for both instance variables and a `toString()` method

  ii. Test your implementation by adding a `main()` method that creates 3 new `Building` objects (using each constructor at least once) and verifies the operation of the accessor, mutator and `toString()` methods.

  iii. When you are happy that the application works as expected, move the `main()` method to a new Java file called *AnytownTest.java* and verify that running this file returns the same result.

Key OOP Concepts

- The small application developed so far demonstrates some of the key concepts in OOP (Object-Oriented Programming)
    - **Classes** and **objects** – a class is a template from which objects are created. Each object has a collection of states (properties) as well as a collection of behaviours. States and behaviours are implemented in Java as instance variables and methods.
    - **Abstraction** – the user (client) is only aware of the operations that are provided on an object, not how they are implemented.
    - **Encapsulation** – variables and methods are wrapped in one single unit. Variables are kept hidden (private) and can only be accessed from outside through setter and mutator methods