Appendix 2: Unified Modelling Language (UML) — class diagrams (SDD)

To model a system, it is important to capture the static behaviour of the system.

A class diagram is used for a quick overview of the system. It describes the structure of a system by showing its:

- classes
- variables, structures and types
- methods of the class
- relationships between the classes

The purpose of a class diagram is to model the static aspect of the system.

Drawing a class diagram

A class is a blueprint for an object. A class diagram describes each class and the relationships between the classes.

UML class notation

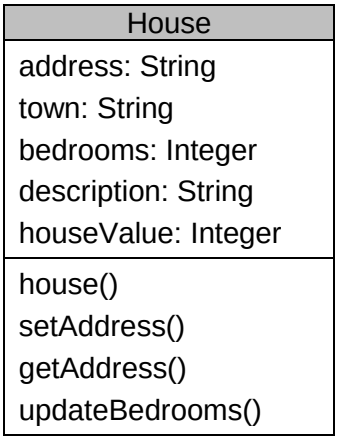
A class diagram consists of:

- a class name
- instance variables and data types:
  - public
  - private
- methods:
  - public
  - private
  - constructor
- inheritance between classes

Example
A program is being written for an estate agency to store the details of houses for sale or available to rent.

Class diagram for House

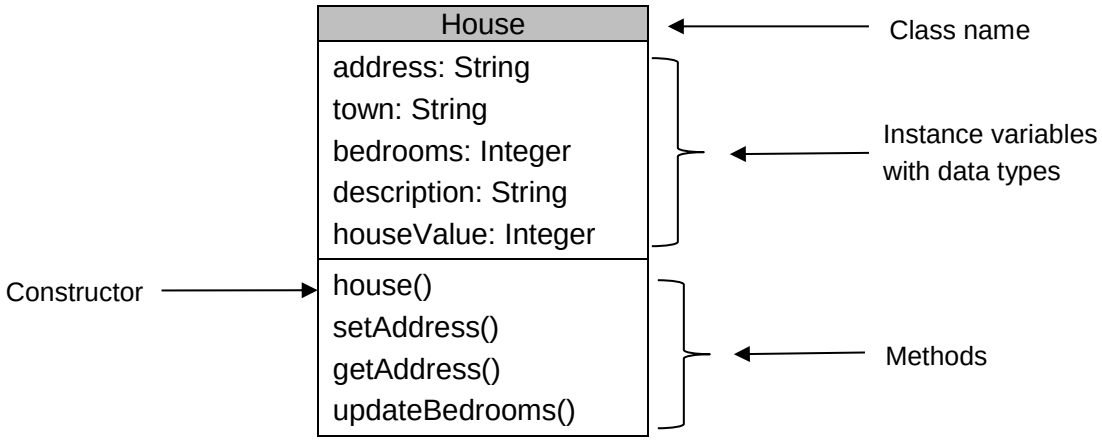
Part of the class diagram for the House class is shown below.



Explanation

The class diagram indicates the:

- class name
- instance variables with data types in the class (instantiation variables)
- methods associated with the class (including the constructor method)



Constructor

A constructor is shown on a UML class diagram in the methods section. The constructor will have the same name as the class name. The constructor method is used to create an individual object that belongs to the class.

Public and private

The instance variables and methods within a class can be public or private elements.

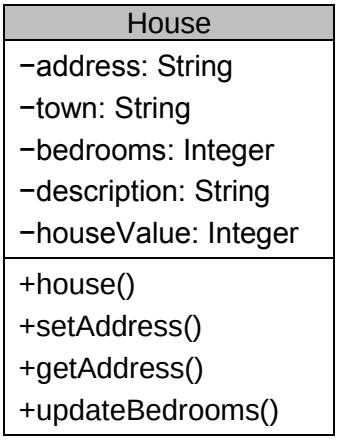
Public elements can be used by any class; however, private elements can only be used by the owning class.

UML allows any variable or method to be shown as public or private.

In a class diagram:

- public elements are preceded with a + sign
- private elements are preceded with a − sign

The House class, with public and private elements, will look as follows.



The set and get methods (sometimes called mutators and accessors) are needed to retrieve (get) or edit (set) the values held in private variables.

Example code: `setAddress()`

Used to edit the value stored in the private instance variable `address`.

```
PROCEDURE setAddress(STRING newAddress)
      SET THIS.address TO newAddress
END FUNCTION
```

Example code: `getAddress()`

Used to retrieve the value stored in the private instance variable `address`.

```
FUNCTION getAddress() RETURNS STRING
      RETURN THIS.address
END FUNCTION
```

Inheritance

UML allows the object-oriented construct of inheritance to be exemplified.

A subclass can inherit all of the properties and methods of a superclass.

On a UML class diagram, this type of inheritance is indicated by an arrow from the subclass to the superclass.

Array of objects

The instance variables of a class or subclass can include an array data structure. This can be used to store instances of another class.

An array of objects is written as:

```
scores: Array of Score[ ]
```

where Score is another class. On a UML class diagram, the connection between the array of objects and the object (class) is also indicated by an arrow.

Example

The program below is for an estate agency to store the details of houses available for sale or to rent.