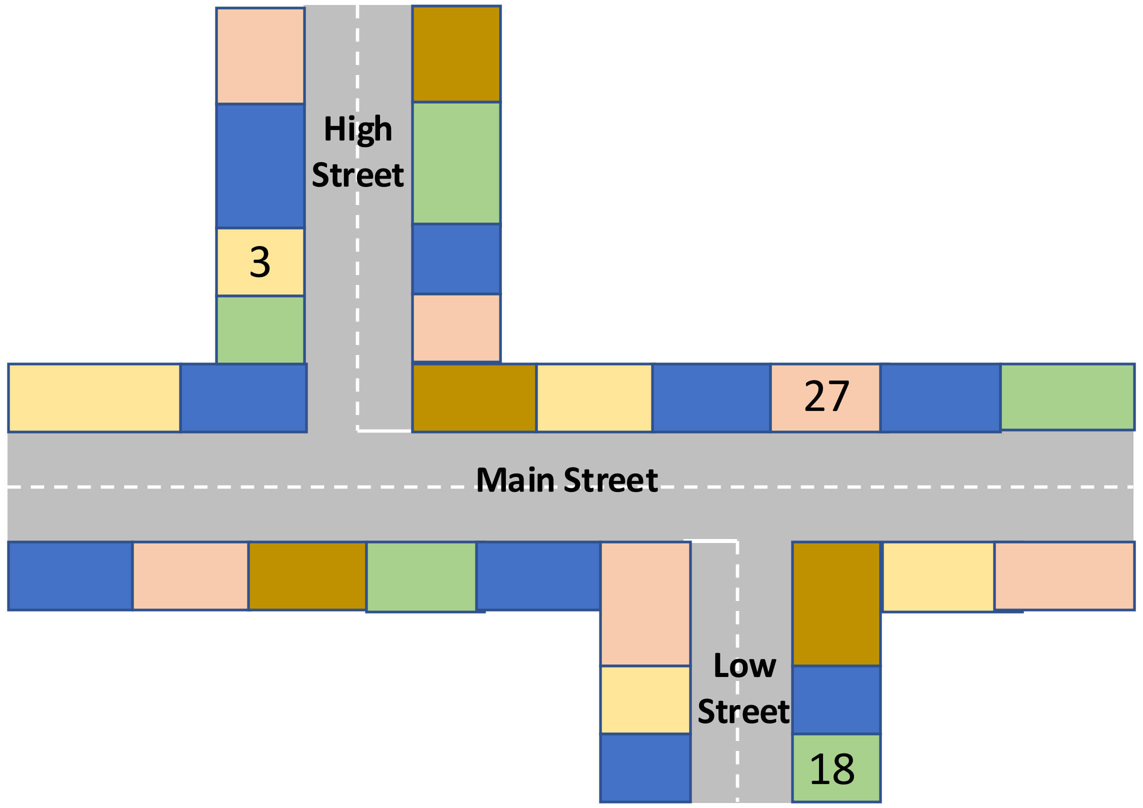
COM410 Programming in Practice

A1.2 Inheritance

Anytown Street Plan

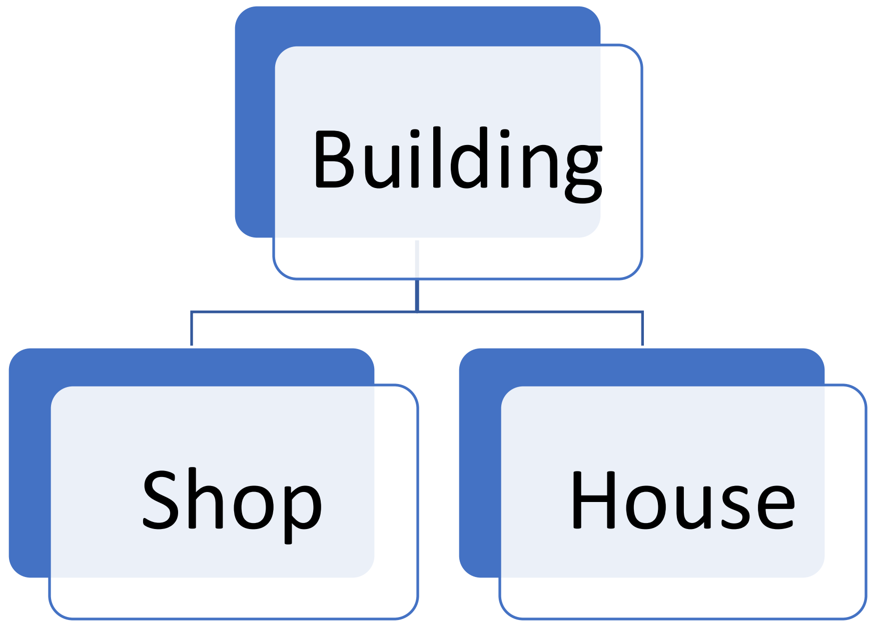


- Anytown – a small village with 3 streets of buildings where each building is described by
    - its address
    - its owner (or business name)

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

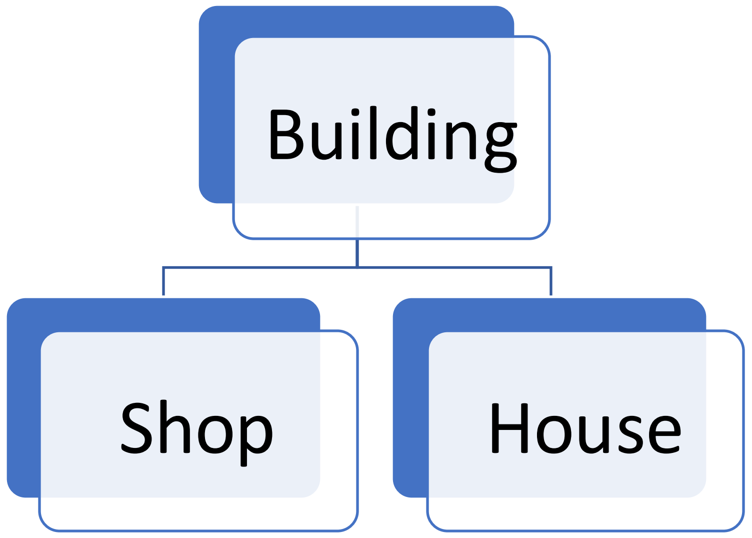
- We identify 2 types of building
    - Shops
    - Houses

Hierarchy of Objects



- Shops and houses are types of building
    - Some properties (attributes) will be common to all buildings (e.g. all buildings have an address, all buildings have an owner)
    - Other properties may be specific to shops or houses (e.g. number of bedrooms in a house or number of employees for a shop)
- A class hierarchy allows us to specify properties at the appropriate level

Hierarchy of Objects



Building 1
Address: 3 High Street
Owner: Smith's Newsagent

Is a: Shop
Number of employees: 5
Annual turnover: £250K

Building 2
Address: 18 Low Street
Owner: Mary Jones

Is a: House
Number of bedrooms: 3
Has a garage: false

- Shops and houses are defined as types of building, but with additional properties
    - Will be implemented as additional instance variables and methods specific to each

Creating the subclass

```
public class House extends Building {
   private int  numBedrooms;
   private boolean hasGarage;

   public House(String add, String own,
                int beds, boolean garage) {
      super(add, own);
      this.numBedrooms = beds;
      this.hasGarage = garage;
   }
   // plus accessor and mutator methods

}
```

a subclass of `Building`

`House`-specific values

Superclass constructor

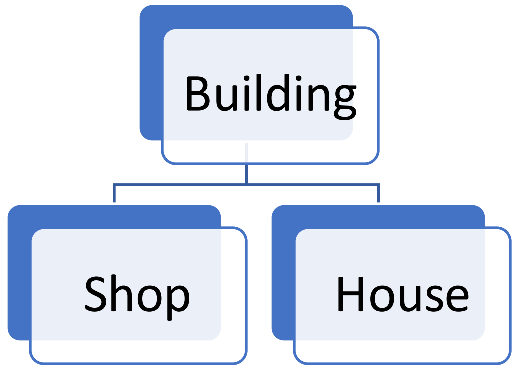
Set `House`-specific values

Scenario

- Return to your **Anytown** project and make the following modifications:
  1. add the new classes `House` (in a new file ***House.java***) and `Shop` (in a new file ***Shop.java***) where both are subclasses of the class `Building`, `House` has instance variables `numBedrooms` (integer) and `hasGarage` (boolean) and `Shop` has instance variables `numEmployees` and `averageTurnover` (both integer values)
  2. Create constructors for `House` and `Shop` that cater for (i) no parameters provided and (ii) all parameters provided.
  3. Implement accessor and mutator methods for all instance variables in both new classes
  4. Implement a `toString()` method for each class that generates a bespoke message for instances of the class, but which uses the `toString()` method in the superclass
  5. Test your implementation by adding code to the existing a `main()` method in the class `AnytownTest` that creates new `House` and `Shop` objects (using each constructor at least once) and verifies the operation of the accessor, mutator and `toString()` methods.

Key Concepts

- This example introduces more key concepts of object-oriented programming
  - **Inheritance** sharing of attributes and methods between classes arranged in a hierarchy.
  - Sub classes will inherit all attributes and methods from their super class
  - Attributes and methods can be over-ridden if desired



- Classes **`Shop`** and **`House`** have inherited address and owner properties…
- …as well as their own individual properties

- **Polymorphism** is the name given to the **is-a** relationship that a class hierarchy produces
  - E.g. A **`Shop`** **is a** **`Building`** and **`Shop`** objects have access to all of the methods of class **`Building`**

Challenge

Pet Hierarchy

i. In a new Java project called Pets, define a new class `Pet` with instance variables `name` and `age`. Provide a suitable constructor and accessor/mutator methods for the class.

ii. Now, define two new classes `Cat` and `Dog` as subclasses of `Pet`. Each sub-class has an additional instance variable `breed` that can store the particular type of cat (e.g. Persian, Tabby, etc.) or dog (e.g. Spaniel, terrier, etc.).

iii. Each sub-class should contain a method called `speak()` that returns a typical animal noise, plus a description of the animal such as

“*Miaow! I am Pixel, a 4 year old tabby*”, or

“*Woof! I am Rex, a 9 year old terrier*”.

iv. Provide a further class `PetTest` that implements a `main()` method to test your new class hierarchy.