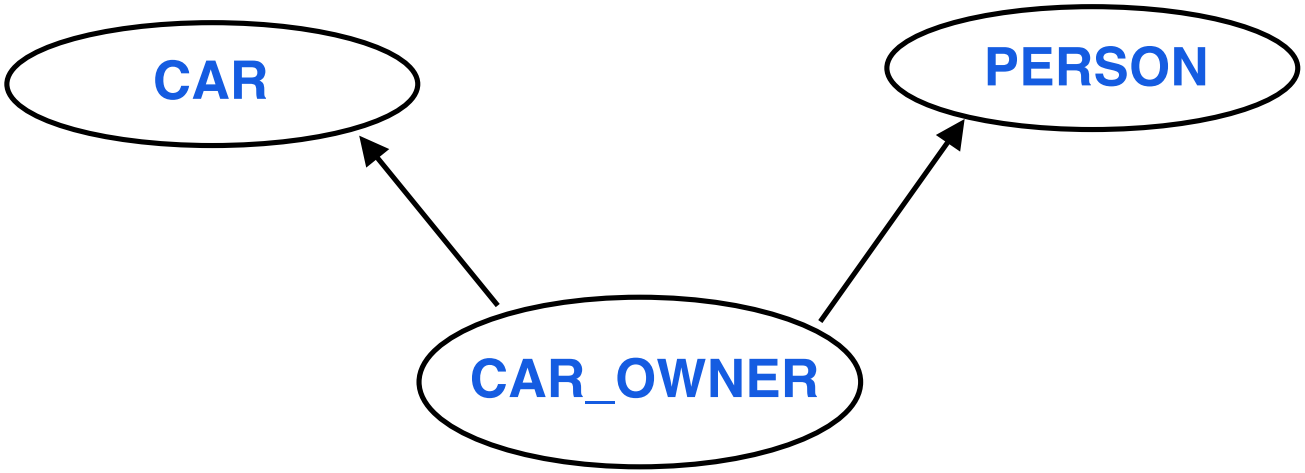
Designing Classes
Part 2

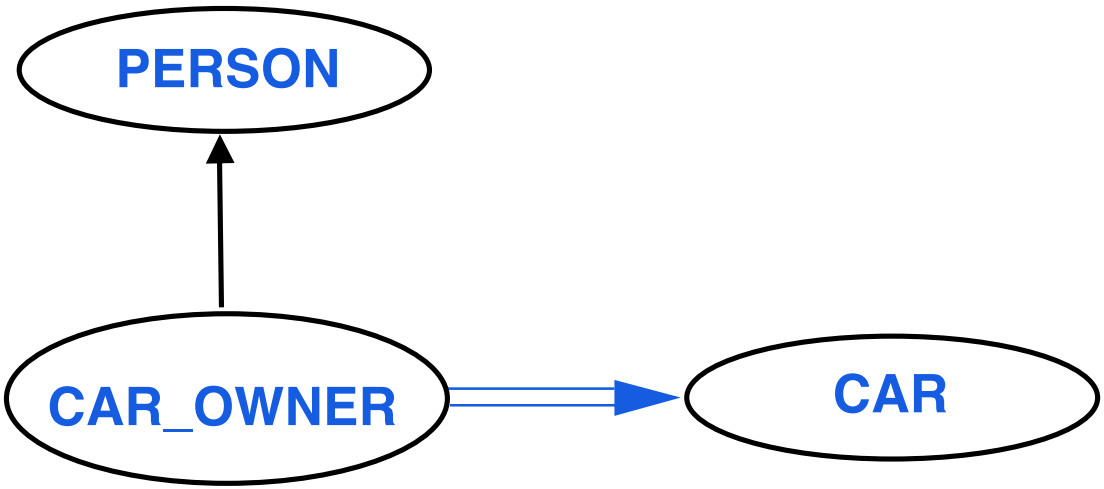
How Not to Use Inheritance

- Class CAR, class PERSON put together to define a new class CAR_OWNER
- Every CAR_OWNER is both a PERSON and a CAR ?



How Not to Use Inheritance – 2

- Correct relationship is client–supplier



Do not make a class B inherit from class A unless you can somehow make the argument that one can view every instance of B also as an instance of A

Use versus Inheritance – 1

- When the **is view** is legitimate, the **has view** can be taken instead

  The reverse is not usually true

  CAR_OWNER

- Two criteria help us resolve such arguments

  Though they sometimes fail to give a clear cut solution

Rule of Change

- Client relations permit change, while inheritance does not
  - » **If B inherits from A then every B object is an A object and no object can change this property**
  - » **If a B object has a component of type A it is possible to change that component (up to the constraints supplied by the type system)**

Use versus Inheritance – 2

- Basic rule
  - » **Client is a has relationship**
  - » **Inheritance is an is_a relationship**
- It is a **wicked problem** to decide

  due to difficulties of system modelling
- Compare the following
  - » **Every software engineer is an engineer**
  - » **In every software engineer there is an engineer**
  - » **Every software engineer can have an engineer component**

Rule of Change – Example

```
class SWENG inherit ENGINEER ...
class SWENG2 feature me : ENGINEER ...
class SWENG3 feature me : VOCATION ...
```

- In the first, object relationship cannot be changed dynamically
- In the other two, new values can be assigned to **me** – up to type constraints
  - » **Software engineer is also a juggler**

Do not use inheritance for a perceived is_a relation if the corresponding object components may have to be changed at run time.

Polymorphism Rule

Inheritance is appropriate for is_a relations if data structure components of a more general type may need to be attached to objects of more specific type