Level 3

Adventure Games

Riccardo Pucella

October 10, 2014

Adventure games

Adventure games —
    a class of simulation games

Virtual world navigated by the player
Emergent storytelling vs defined narrative

Turn-based

History

Originally text-based:

- *Colossal Cave* (1976), *Zork* (1977)
- Infocom games

Graphical versions:

- *King's Quest* (1984), *Myst* (1993)

Modern reincarnation:

- Interactive fiction

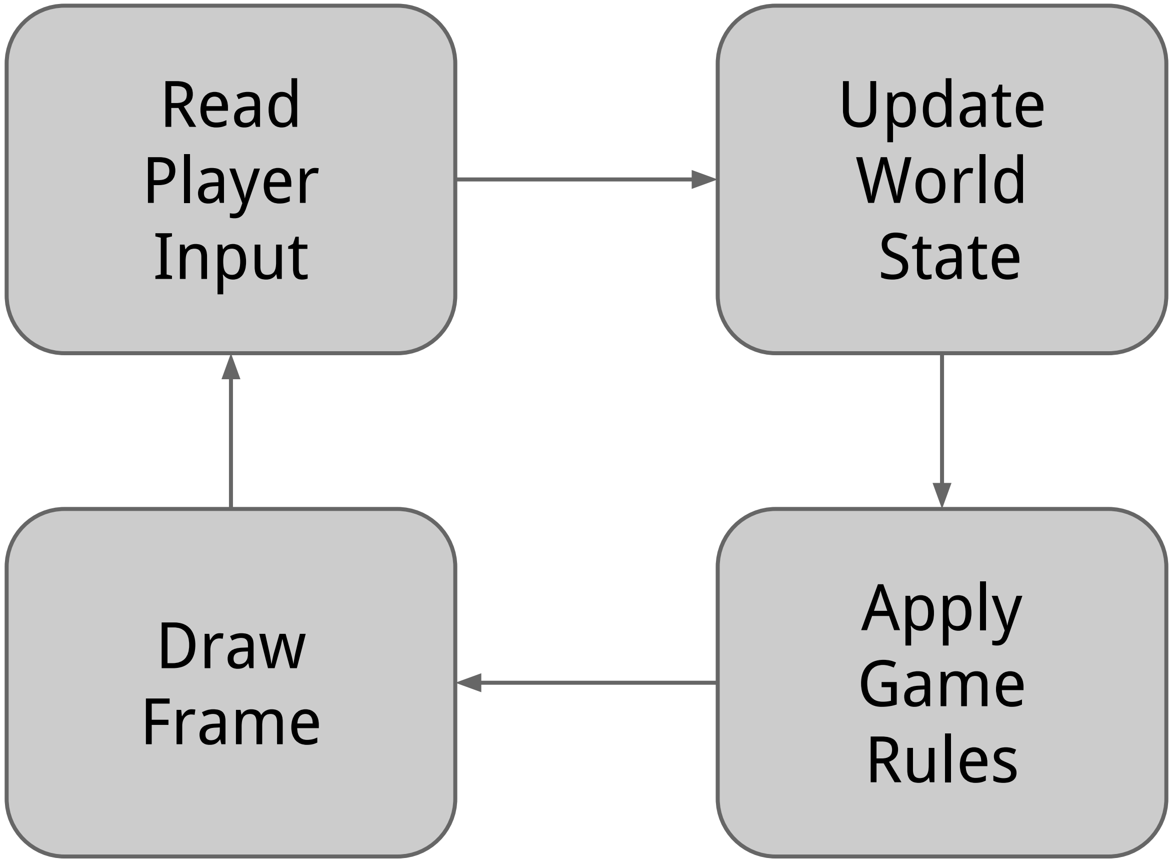
Demo

Level 3 Project

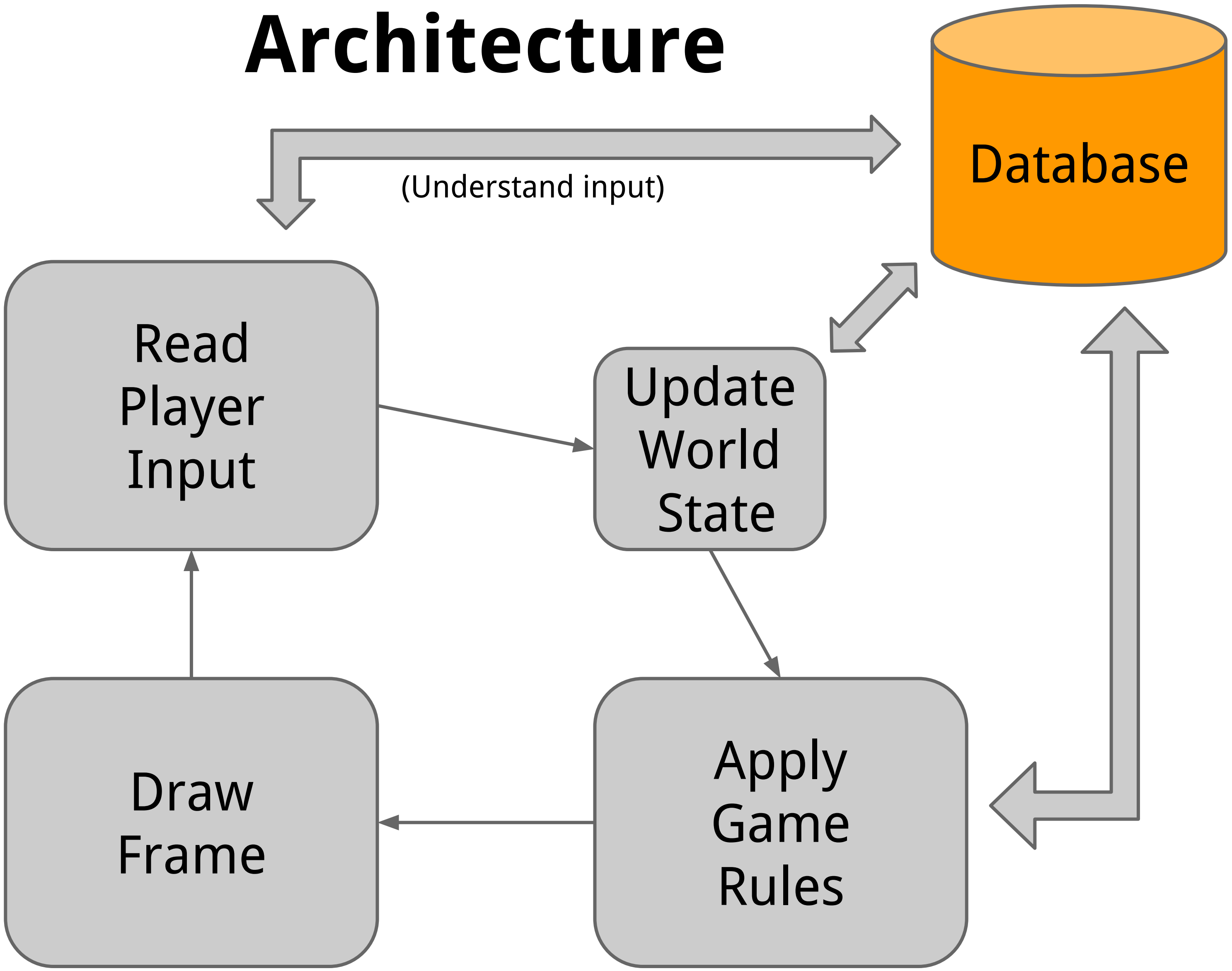
Key Question:

How do you go about programming that?

Architecture



Architecture



Implementing the database

The database represents the various artifacts in the virtual world of the game

Those artifacts interact and respond to actions

OO languages were created *exactly* for that

Simula: the first object-oriented language

- modeling discrete-event simulations

World objects hierarchy

Every artifact of interest in the game is an object in the database

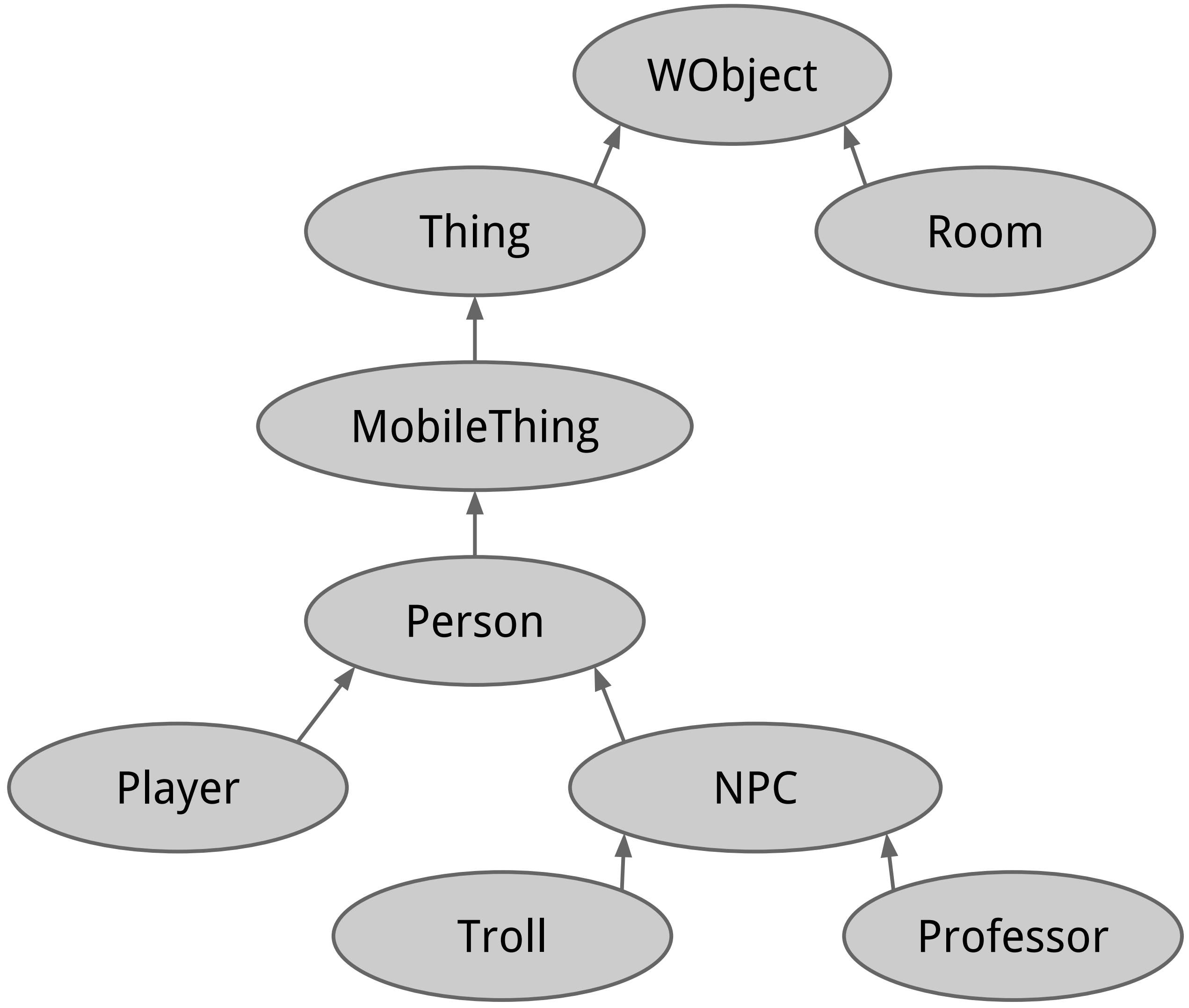
Inheritance relationships between objects based on their kind

Tradeoff between convenience and flexibility

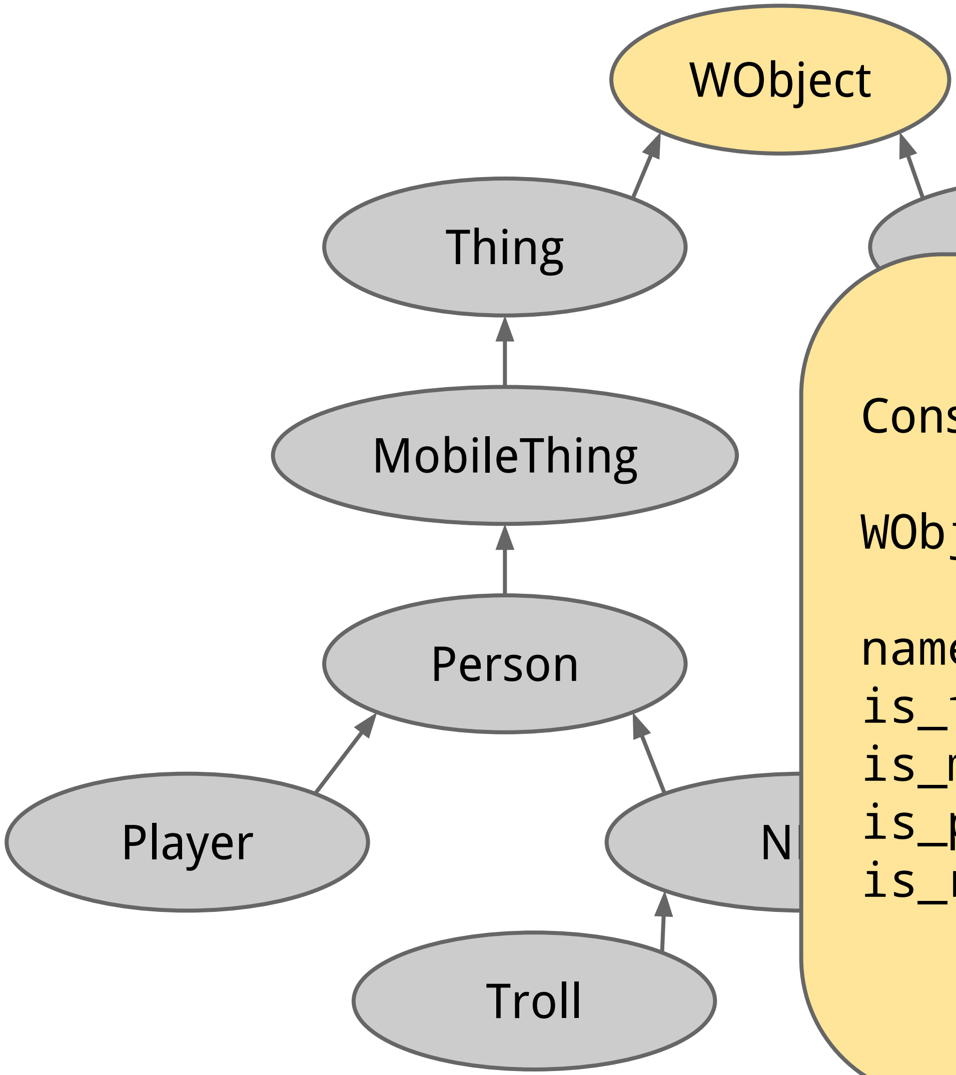
Alternative to general purpose OO language:

DSLs for programming adventure games

World objects hierarchy



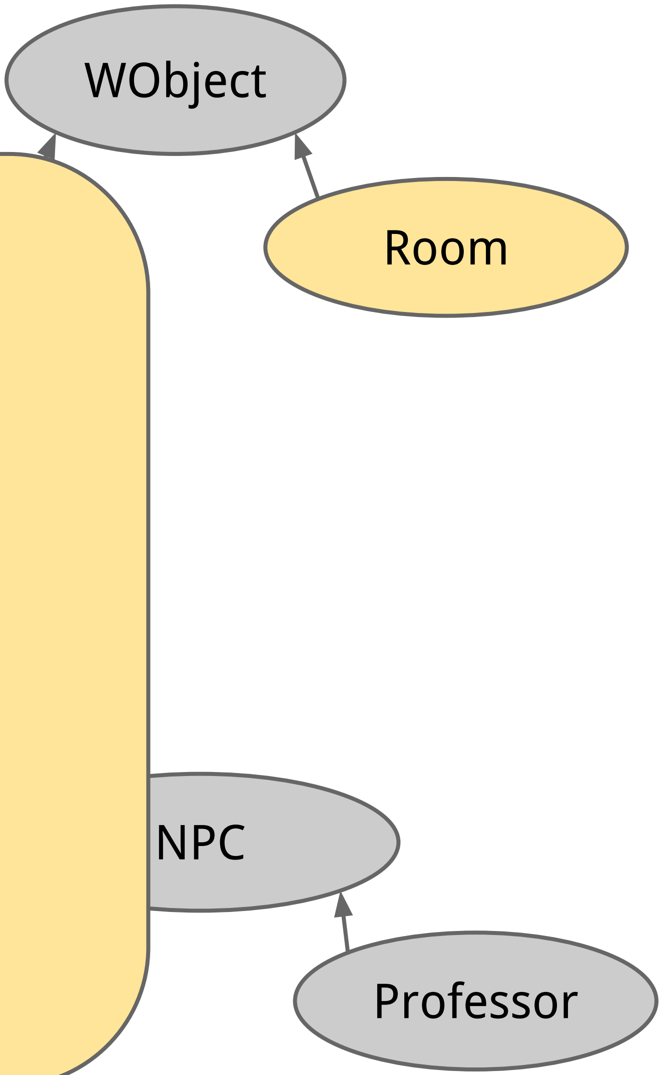
World objects hierarchy



Constructor and methods:

```
WObject (name)

name ()
is_thing ()
is_mobile_thing ()
is_person ()
is_room ()
```

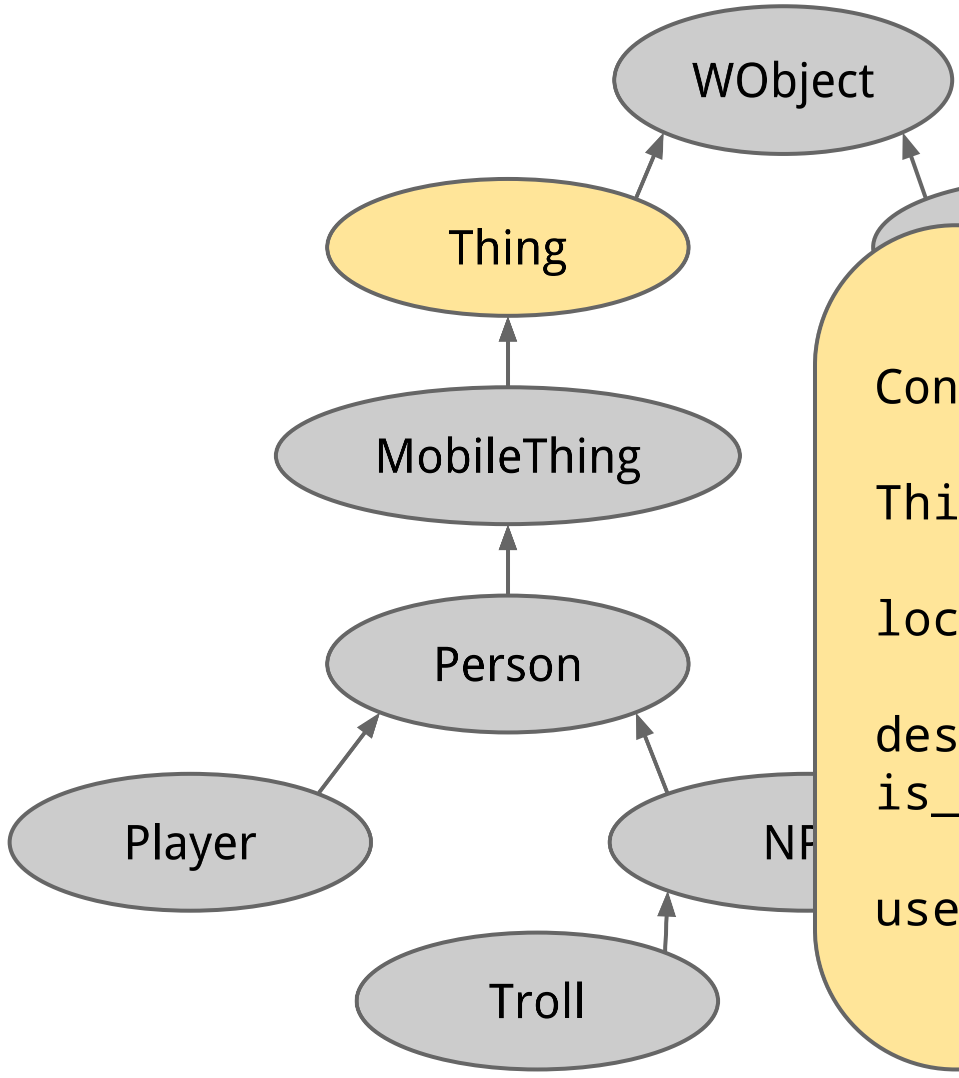
World objects hierarchy



Constructor and methods:

```
Room (name)

exits ()

report (message)
broadcast (message)

contents ()
have_thing (thing)
add_thing (thing)
del_thing (thing)
```

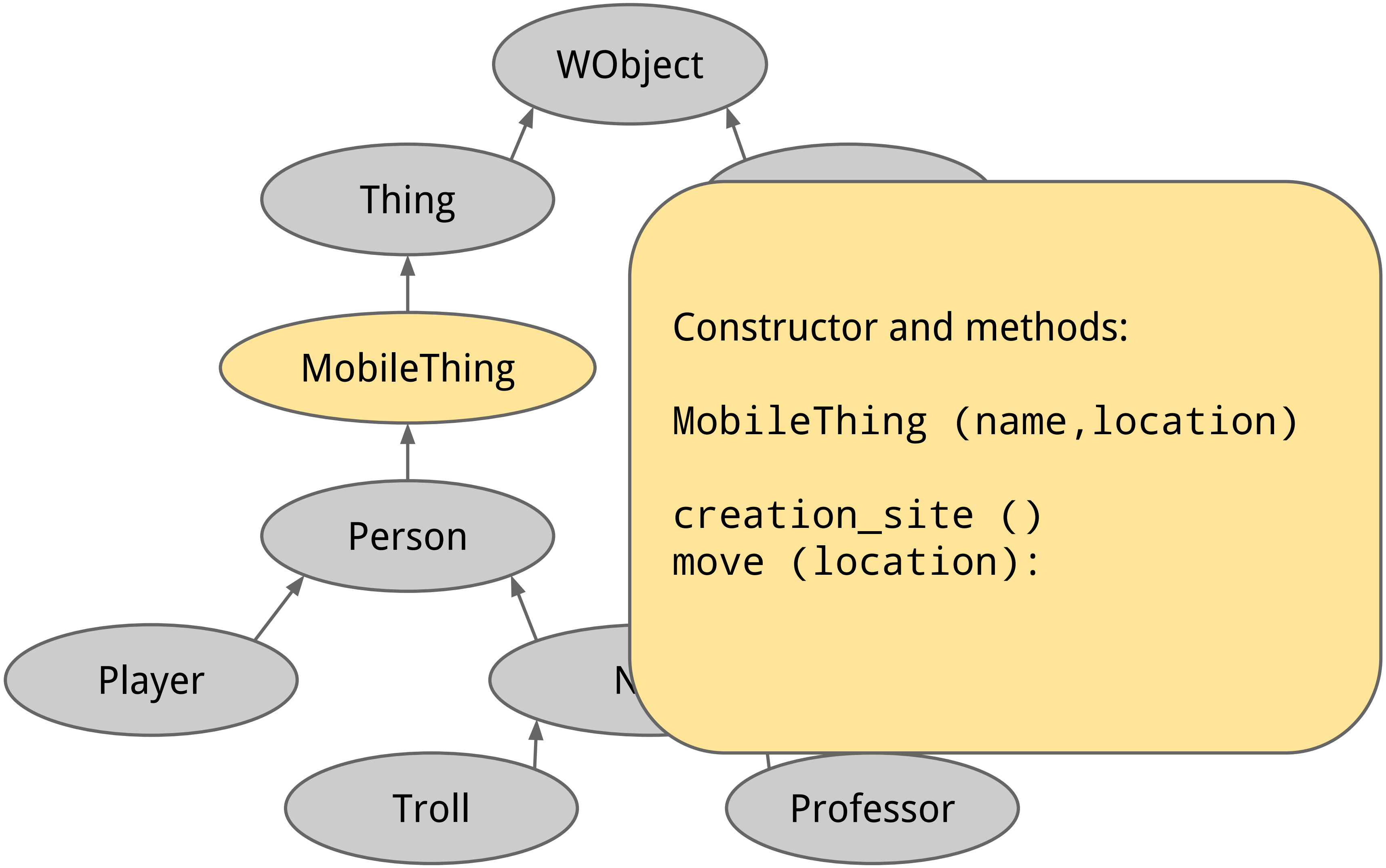
World objects hierarchy



Constructor and methods:

```
Thing (name,location)

location ()

destroy ()
is_in_limbo ()

use (person)
```

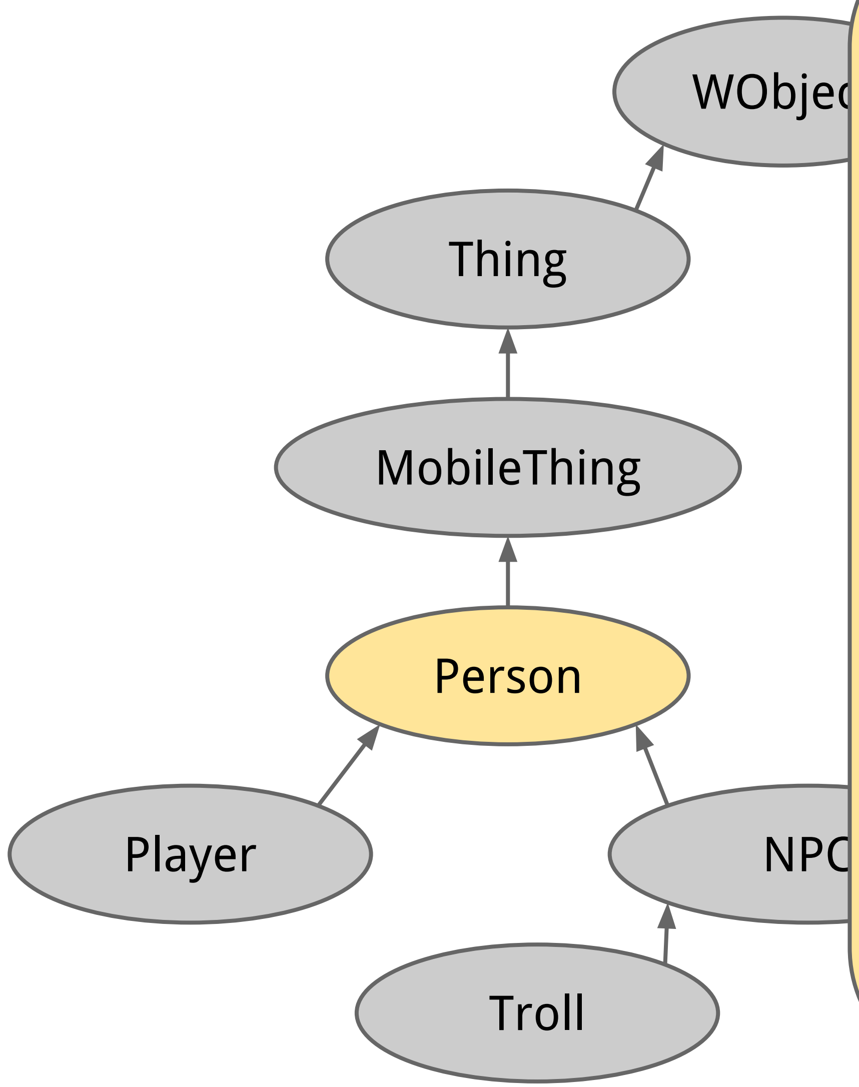
World objects hierarchy



Constructor and methods:

```
MobileThing (name,location)

creation_site ()
move (location):
```

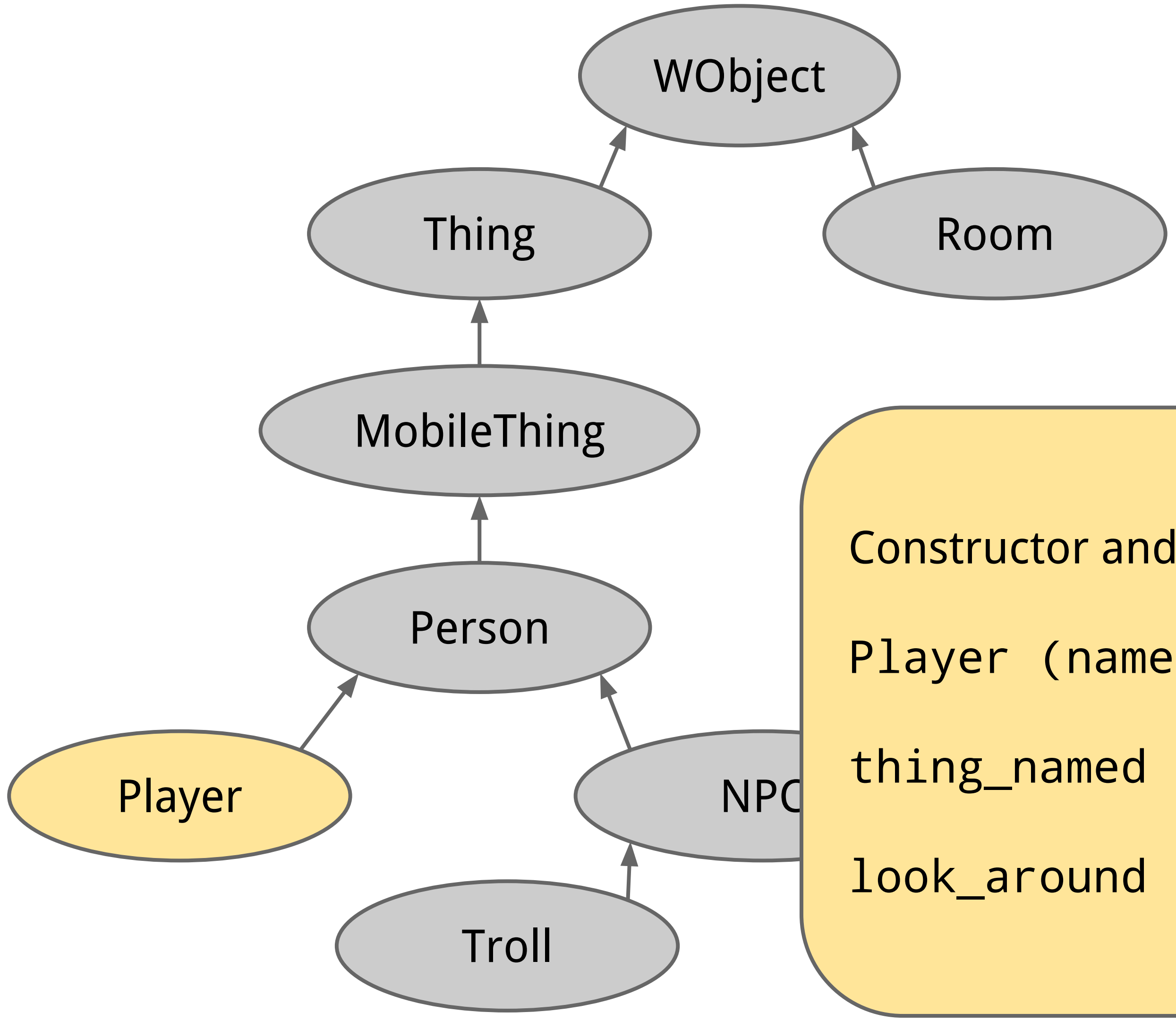
World objects hierarchy



Constructor and methods:

```
Person (name,location)

health ()

say (message)
go (direction)

enter_room ()
leave_room ()

people_around ()
stuff_around ()

suffer (hits)
die ()
```

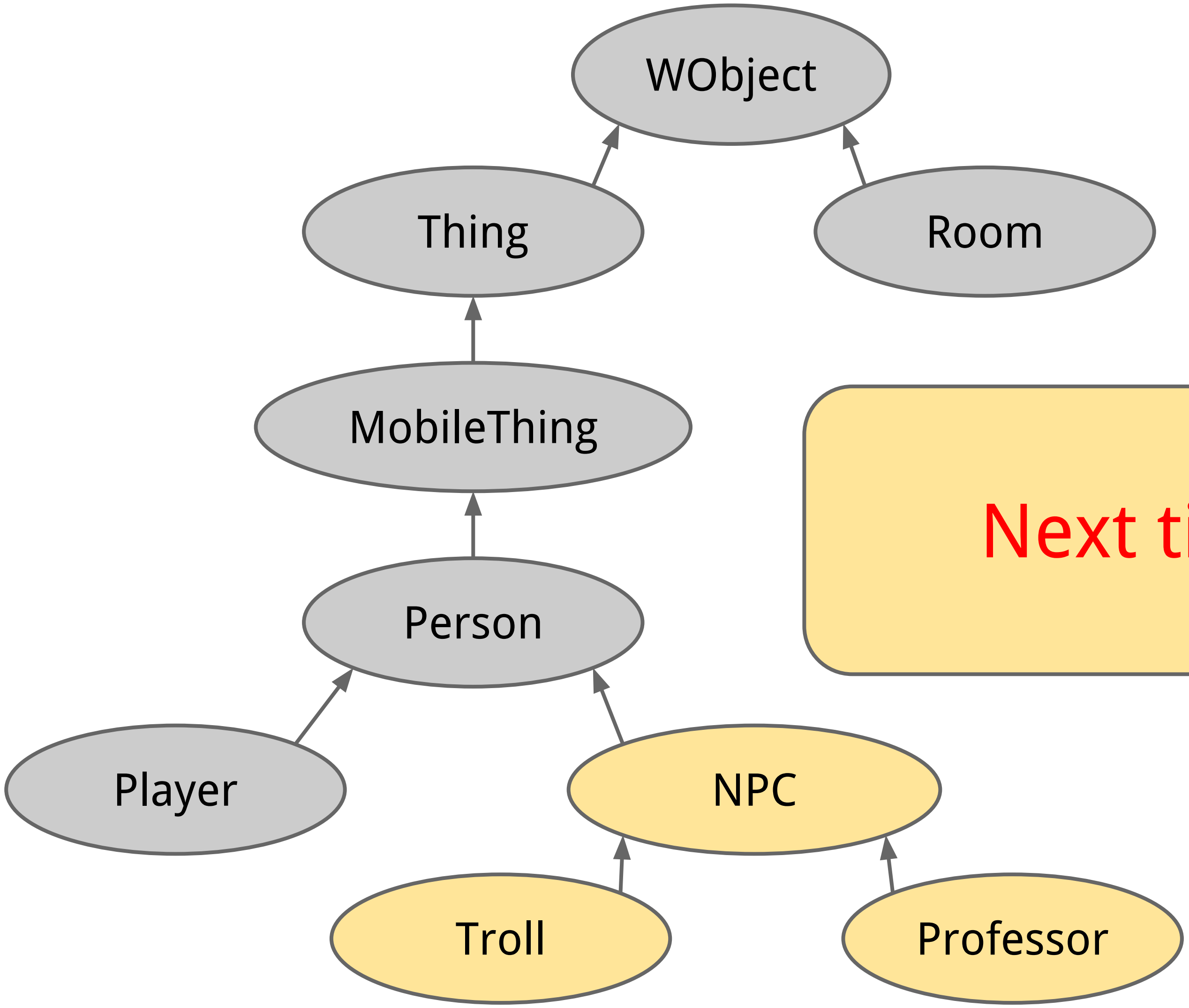
World objects hierarchy



Constructor and methods:

```
Player (name,location)

thing_named (name)

look_around ()
```

World objects hierarchy



Next time…

Global information

Some global information is maintained in static fields (aka class variables):

`Room.rooms` : list of all rooms created

`Player.me` : the current player (as an object)

`Player.god_mode` : for cheating

Understanding player input

Player input of the form:

```
verb
verb name
verb name name
```

Convert this input into actions on the database

Understanding player input

Player input of the form:

```
verb
verb name
verb name name
```

References to a name resolved into a `thing` based on what's around the player (method `thing_named`)

Understanding player input

Player input of the form:

```
verb
verb name
verb name name
```

- Look through list of registered verbs
- Upon a match, resolve names and call verb’s action method
- Action should call suitable method in the database

Example

```
class Use (Verb):
    def __init__ (self):
        Verb.__init__ (self,'use')

    def action1 (self,obj):
        obj.use(Player.me)
        return SAME_ROUND
```

Example

Initialize with syntax for verb

```
class Use (Verb):
    def __init__ (self):
        Verb.__init__ (self,'use')

    def action1 (self,obj):
        obj.use(Player.me)
        return SAME_ROUND
```

Example

`action1` invoked when there's one name after the verb

called with the resolved name's thing

return whether to go to next round or not

```
def action1 (self,obj):
    obj.use(Player.me)
    return SAME_ROUND
```

Next time: adding NPC behavior

Current description covers reactive behavior

- world objects react to player actions

NPCs: proactive behavior

- behaviors not prompted by player actions