W4156

OOAD IV: OO Design

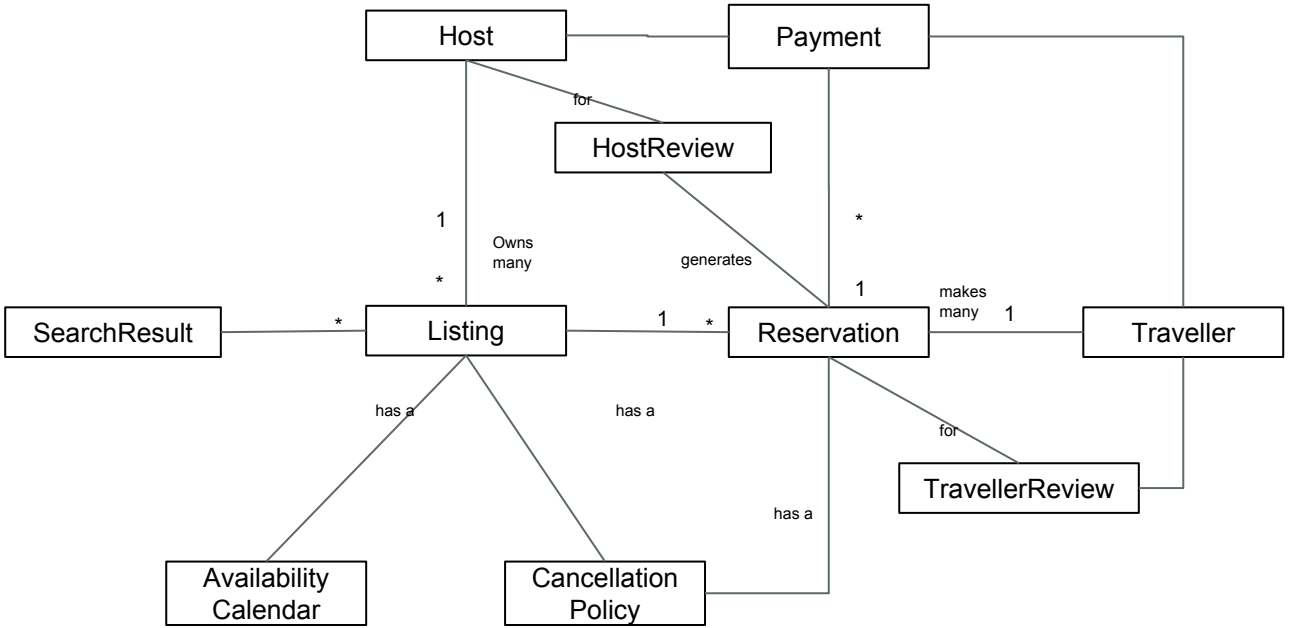
Recap

- OOAD is a *paradigm* to analyze and design software solutions

- In OO analysis we produced a conceptual model by
  1. Using use-cases and noun identification to identify ***conceptual classes***
  2. Identifying the ***relationships*** between conceptual classes
     (Process was also fantastic to gain deeper understanding of problem and to create common language amongst the development team)

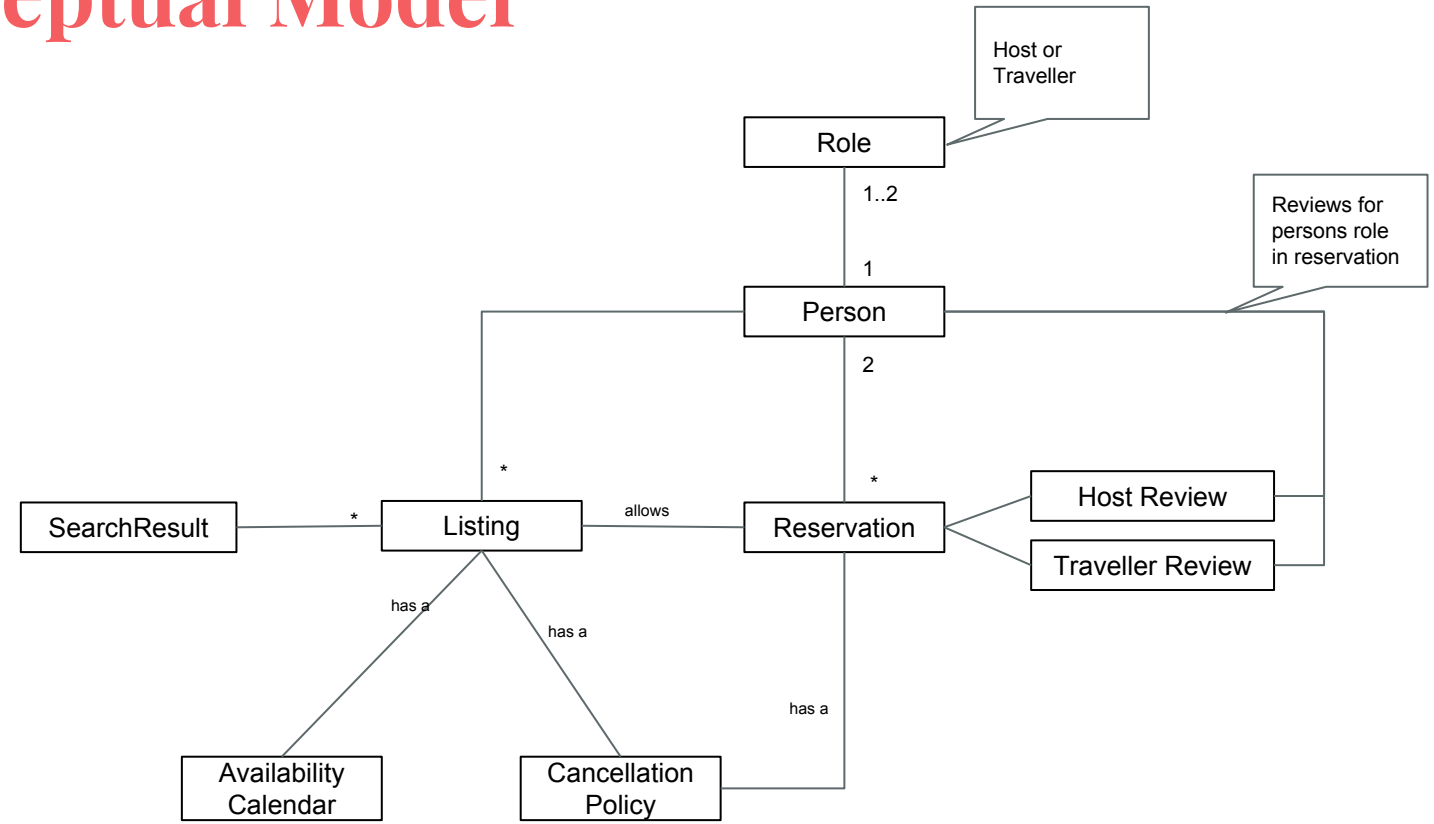
- But we want real working code!
  - The conceptual model did not address many concerns
  - We need to process the conceptual model into an implementation!

Conceptual Model



(I have taken the design and *domain language* from the last lecture forward for continuity)

Conceptual Model



(I decided since people can fulfill both roles there are really ‘people’ with roles)

Agenda

- Design phase
  a. Objective
  b. Process
- Worked Example
- Newsflash!

Design Phase

Generate an implementable set of classes by:

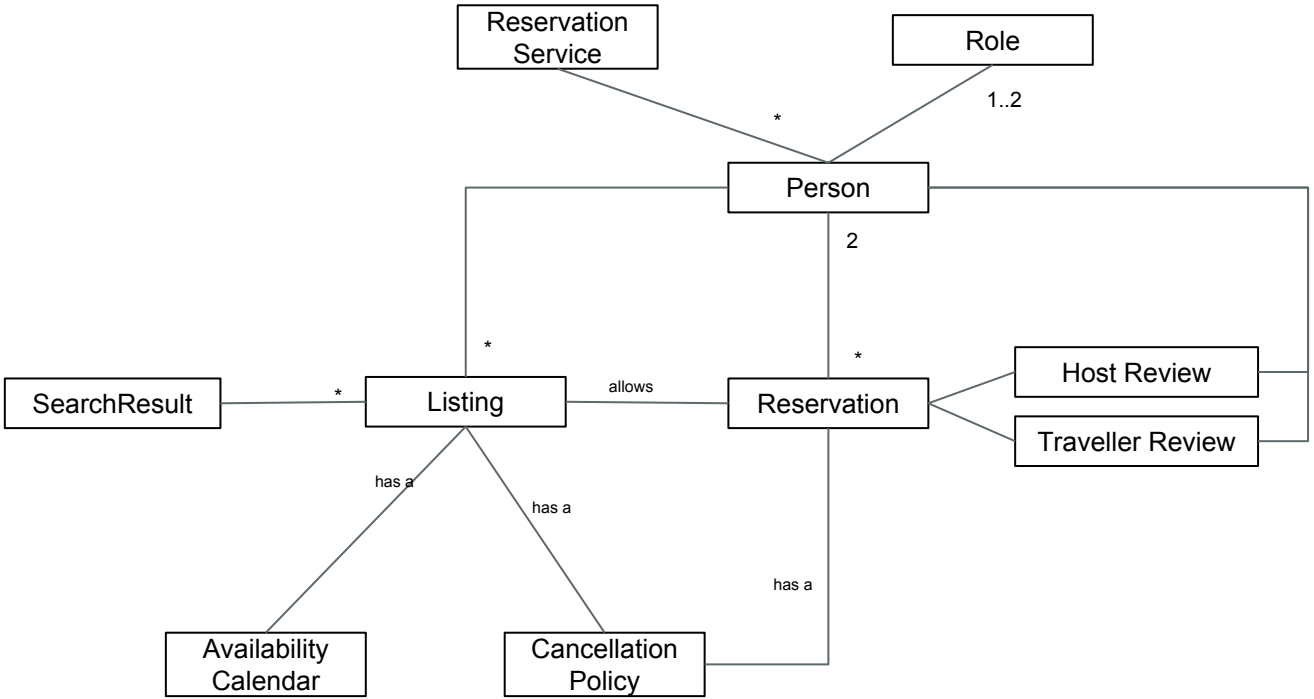
1. Refine the ***set of classes***
    a. add/delete/split vs the conceptual model
    b. Sanity check: “What is the ***responsibility*** of each class?”

2. Use scenarios (use-cases) to design ***each class***
    a. What ***methods*** does it ***need*** to provide?
    b. What ***data*** does it ***need*** to hold?
    c. How do we implement the ***relationships***

NOTE - 1-2 will be iterative. Use-cases will expose deficiencies in design

(note: we will ignore section 7.1.1 of the book per previous lecture about architectural style)

1a. Refine Set of Classes



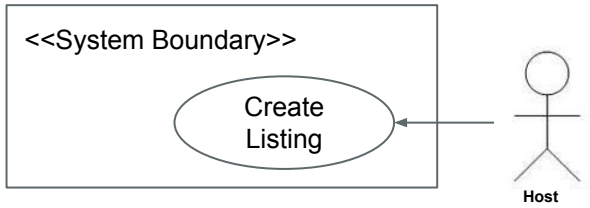
(I have taken the design and *domain language* from the last lecture forward for continuity)

1b. Sanity Check Responsibility

For each class ask “what is this class responsible for?”

- A central aspect of OO design is encapsulation and abstraction
- Objects should have a single responsibility (SRP)
- Answer should not be ‘does X *and* Y’ or ‘does ‘A *or* B’

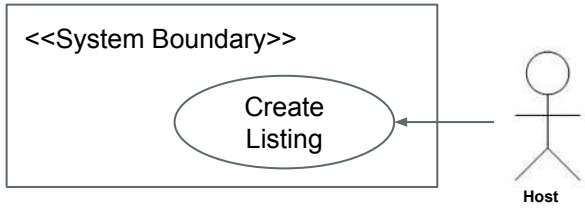
2a. Test Design w/ Use-Case

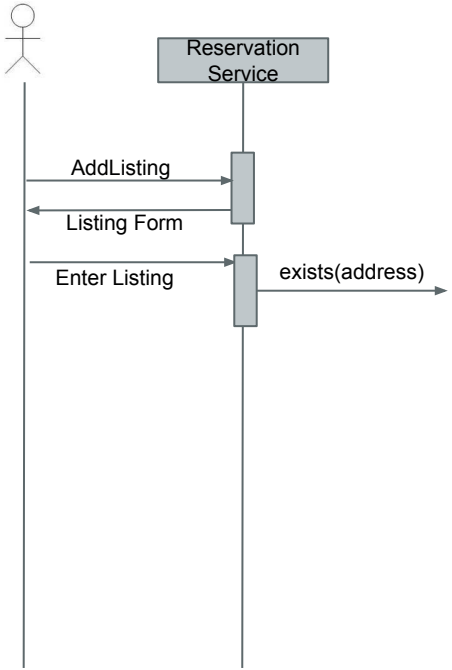


Title	Create Listing
Actor	Host
Precondition	Host is registered and in good standing
Basic Flow	1. Select list property 2. Enter property details 3. Select house rules and cancellation policy 4. List Property
Alternate	4b. If property is duplicate, can not be registered or address cannot be validated the listing will be rejected

We want to establish “does design support use-case?”

2a. Test Design w/ Use-Case





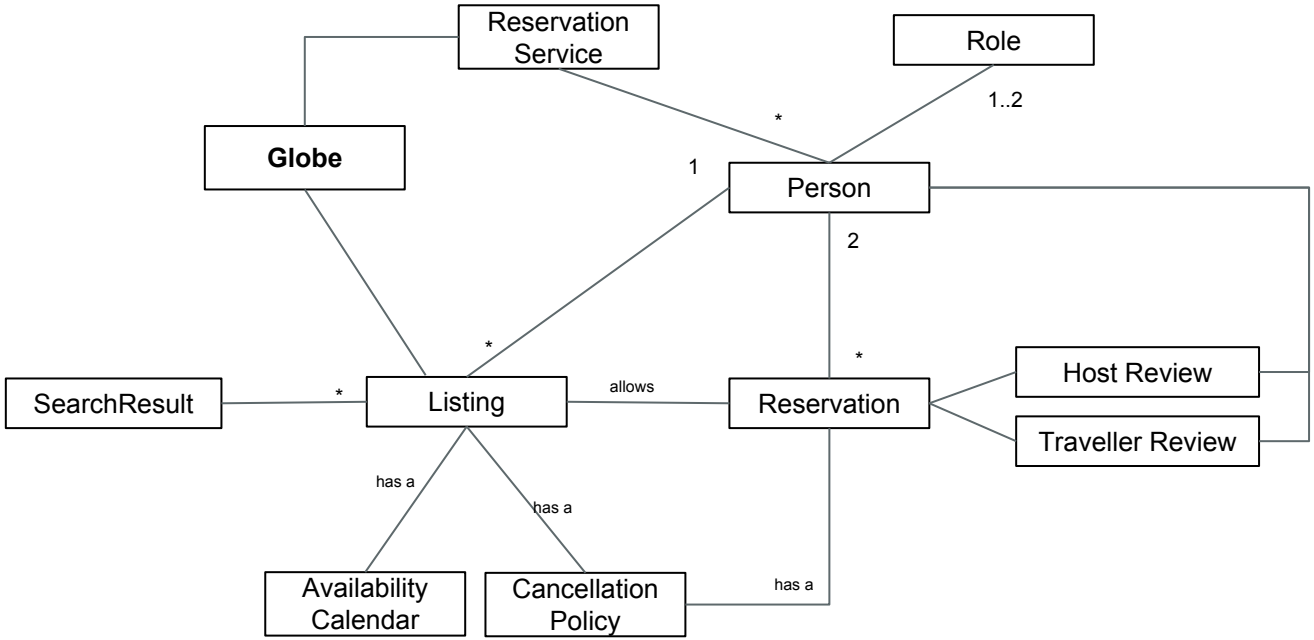
Oh. This is kind of awkward. How can I check it an address is unique?

In my conceptual model people have listings.

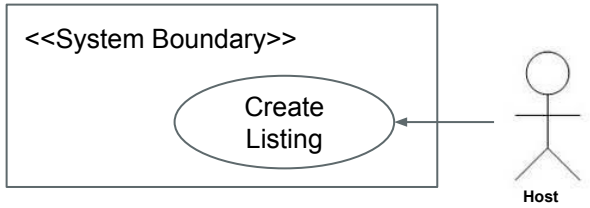
I *could* iterate through all the people and all their listings to see if the address is already registered but this is ugly.At the implementation level I need to know the set of registered listings….

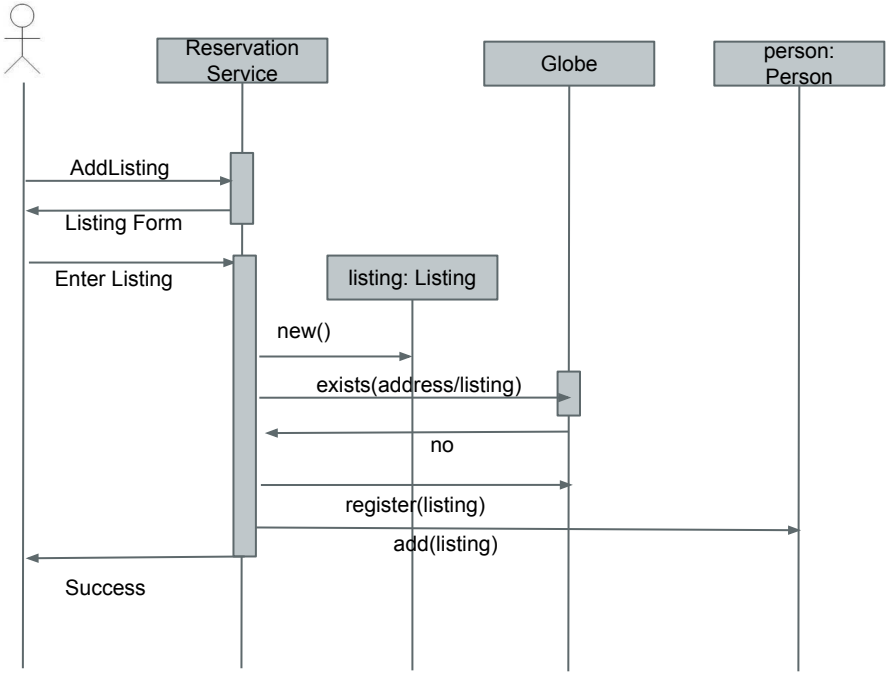
Other thought - wow - walking through use-cases really helps understand if my design meets my requirements!!!

1a. Refine Set of Classes

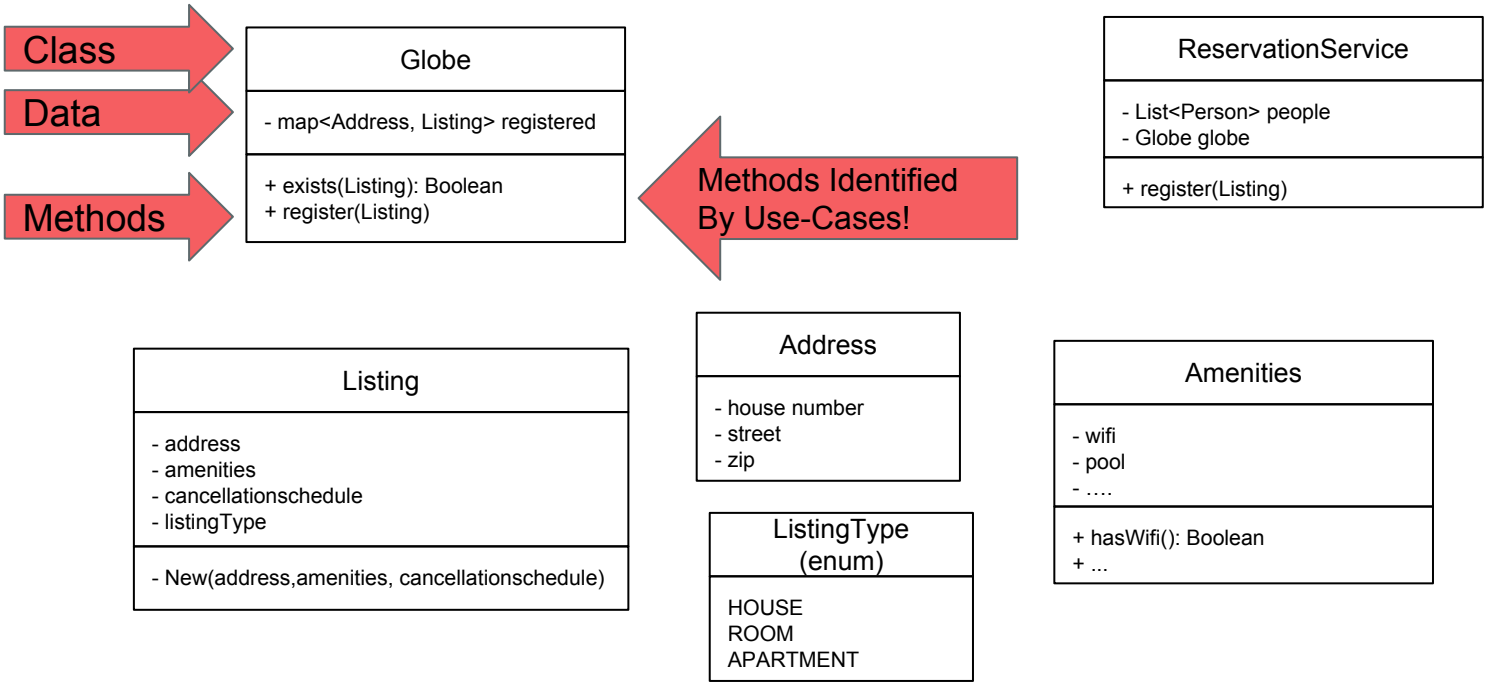


2a. Test Design w/ Use-Case



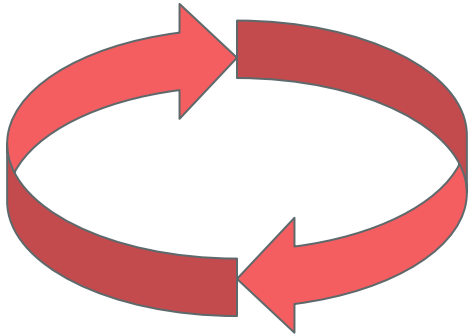


2a-c. Methods, Data, Relationships



1. Drastically over simplifying ‘address’ particular for global usage. Furthermore, we are skipping the geocoding of the address entered by a person into the address
2. Amenities could be implemented as a map of capabilities rather than a structure of Booleans

Repeat for Use-Cases



1. Design will evolve/iterate as we go through use-cases, add/delete/refine.
2. Stop once we have met immediate requirements
3. *Balance* thinking ahead vs unneccessary over-engineering
4. In the second sprint we will add functionality

Newsflash!
Changing Requirements in Iteration II

Newsflash! Changing Requirements!

1. Law change in NYC
2. Hosts are allowed <=1 NYC located listing
3. Where listing is whole apartment!!!

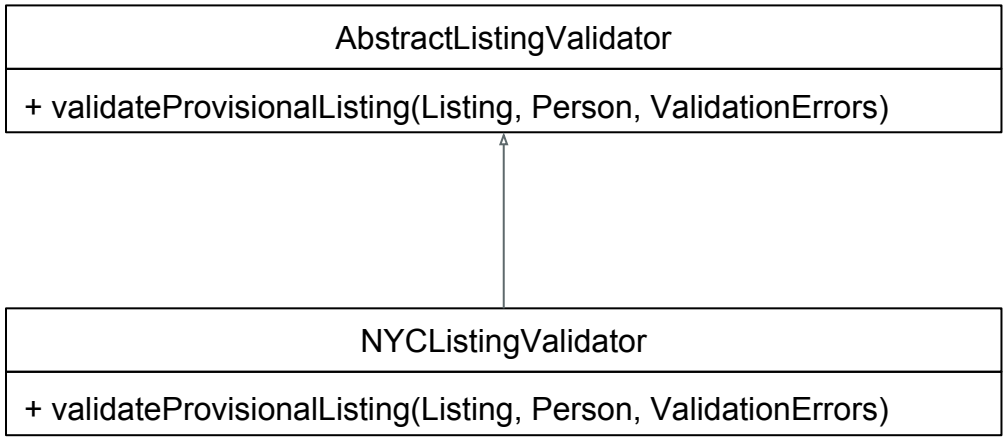
Litmus test of our design ....

A) How can we implement this requirement?
B) Is it *easy* to add OR does it break entire solution?

How to solve this requirement?

1. We could add it as a blob of code within RegistrationService or Globe within register()
    a. But what if we add 50 of this type of requirement? (Bangalore, Glasgow, Blah)
    b. Adding it as a ‘blob’ also seems to be procedural….
    c. Adding it to Globe or Registration seems to violates SRP …..
2. I am remembering something about Objects ……
3. What is this ‘thing’?

It’s a ListingValidationRule!!!

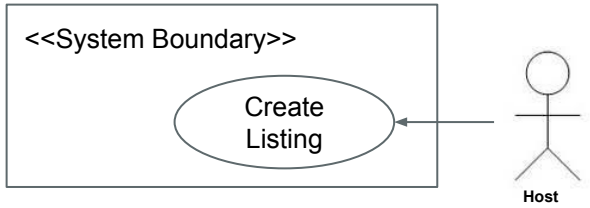


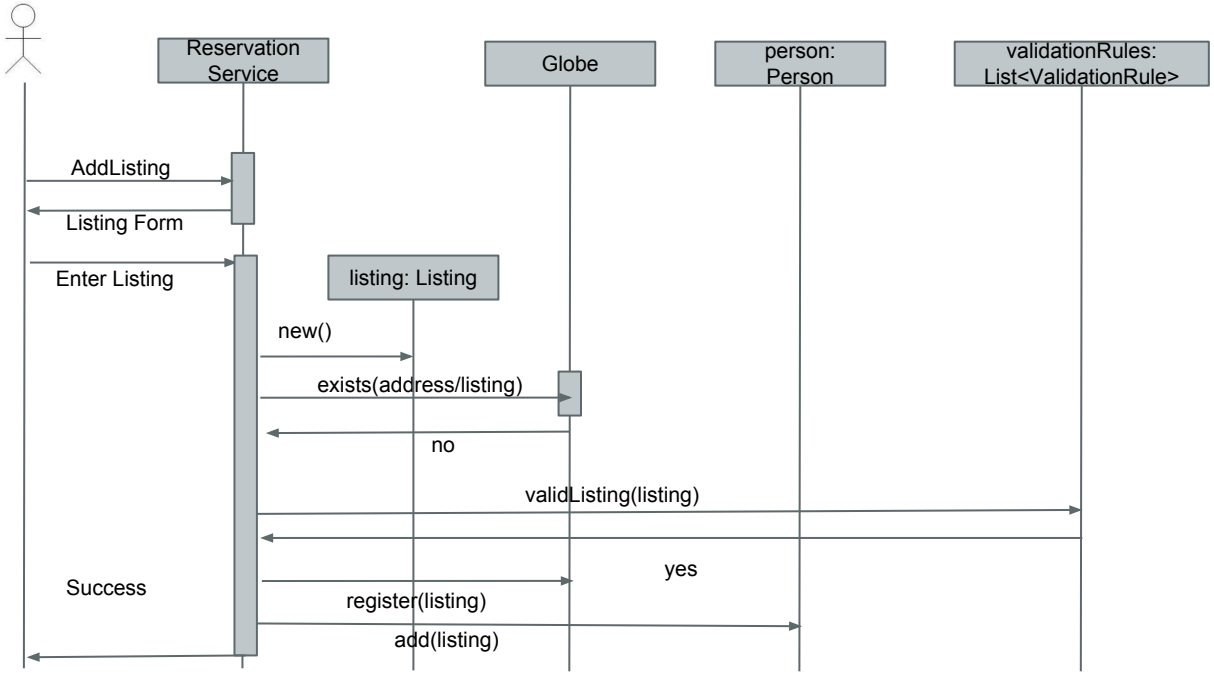
Something like ...

```
class ValidationErrors:

    def __init__(self):
        self.__v_errors = []

    def add_error(self, reason):
        self.__v_errors.add(reason)

    def count(self):
        return len(self.__v_errors)

class AbstractListingValidator(ABC):

    @abstractmethod
    def validate(self, provisional_listing: Listing, person: Person, v_errors: ValidationErrors):
        pass


class NYCListinvValidator(AbstractListingValidator):

    def validate(self, provisional_listing: Listing, person: Person, v_errors: ValidationErrors):
        ny_listings = list(filter(lambda l: l.address._state is "NY", person.listings))
        if (len(ny_listings) > 1):
            v_errors.add_error("Already have a NY listing and Cuomo says NoNo")
```

```
class ReservationService:


    def __init__(self):
        self.__validators = List[AbstractListingValidator]
        self.__people = []
        self.__globe = Globe()

    def register(self, person: Person, listing: Listing):
        if self.__globe.address_exists(listing.address):
            raise Exception("Oh no")

        v_errors = ValidationErrors()
        for validator in self.__validators:
            validator.validate(listing, person, v_errors)
        if v_errors.non_empty():
            raise Exception("Oh no")

        ...
```

2a. Test Design w/ Use-Case





Pop Quiz

Question	Answer
The conceptual class model is <ready/not ready> to implement?	
The design phase translates an _____ into a _____ [sequence diagram, conceptual model, implementable class diagram]	
A class needs to contain *all* methods and data that an entity could conceivably hold? OR A class need to contain the minimal set of data and methods required to support the required functionality?	
We can build the design through iteratively working through _____ and identifying if they are accomodated by the _____. [use-case, class design]	

Warning 1

- The Springer OOAD book also implies a classic 2-tier architecture

We will cover architectural styles later:

- We will assume a single process ‘monolith’
- The OOAD process works for small-medium size problem domain. We will discuss other architectural styles in later lectures
- Ignore section 7.1.1

Reading

Reading	Optionality
Springer OOAD (chapter 7) (Ignore 7.1.1)	Required