CSCB07 — Software Design

Anya Tafliovich

CRC Cards

- A tool and method for systems analysis and design
- Part of the Object-Oriented development paradigm
- Highly interactive and human-intensive
- Final result: definition of classes and their relationships
- *What* rather than *How*
- Benefits:
  - Cheap and quick: all you need is index cards
  - Simple, easy methodology
  - Forces you to be concise and clear
  - Input from every team member

What is a CRC Card?

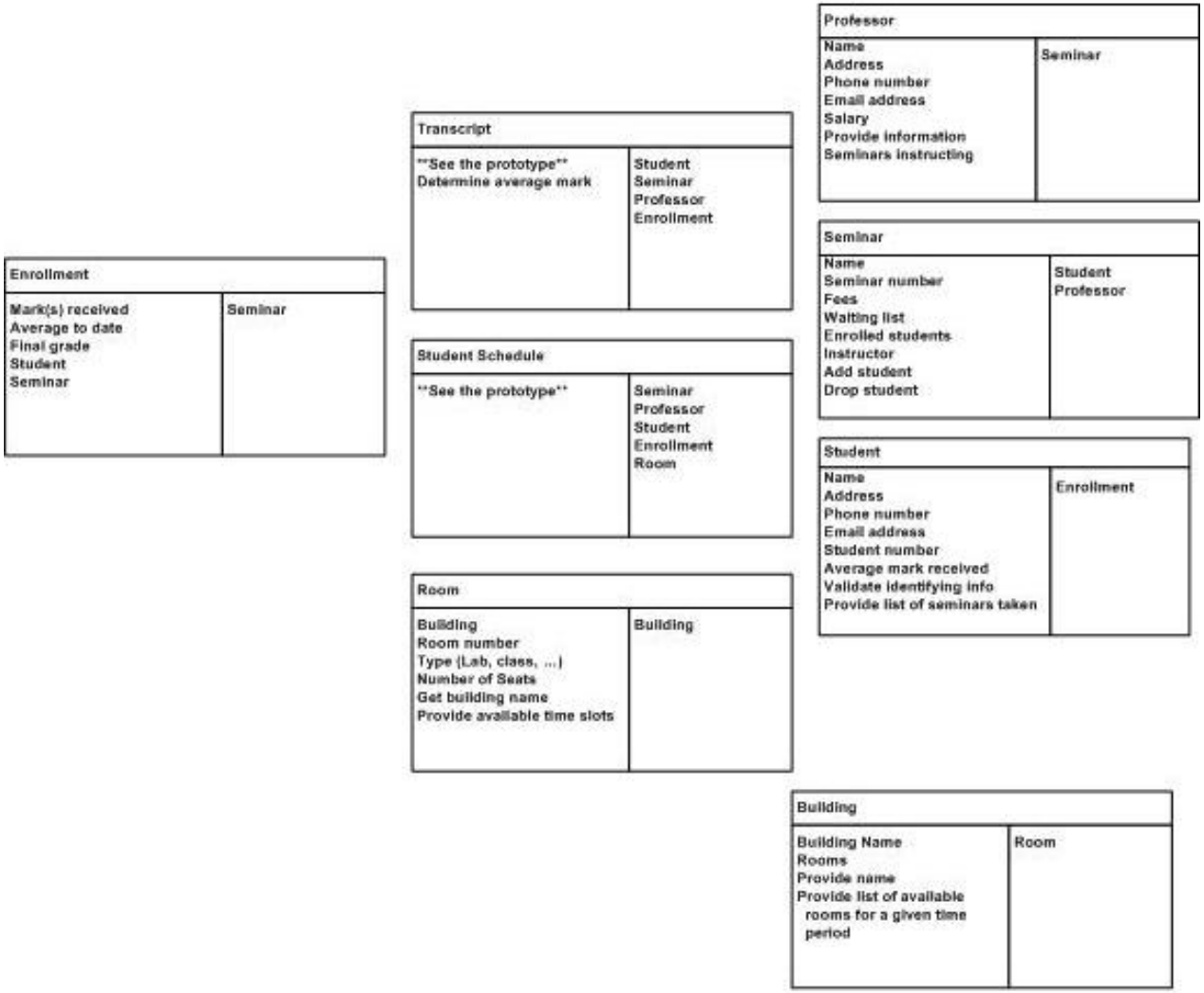
CRC stands for **Class**, **Responsibility**, and **Collaboration**.

- **Class**
  - An Object Oriented class (a name)
  - Also include information on superclass and subclasses here
- **Responsibility**
  - What information this class stores
  - What the class does
  - Some behaviour for which an object is held accountable / responsible
- **Collaboration**
  - Relationship to other classes
  - What other classes does this class use?

What does a CRC Card Looks Like?

Student	
Student number Name Address Phone number Enroll in a seminar Drop a seminar Request transcripts	Seminar

CRC Model



- A **CRC Model** is a collection of CRC cards.
- Specifies Object Oriented Design of the software system

How To Create a CRC Model?

- Typically, you are given a description (in English) of the requirements for a software system.
- You work in a team.
- Ideally, you all gather around a table.
- You need a set of index cards and some pens.
- Coffee / other beverages optional.

How To Create a CRC Model?

- Read the description. Again. And again.
- Identify core classes (simplistic advice: look for nouns).
- Create a card per class (begin with class names only).
- Add responsibilities (simplistic advice: look for verbs).
- Which other classes does this class need to talk to to fulfil its responsibilities? Add Collaborators.
- Add more classes as you discover them.
- Put classes away if they become unnecessary. (But don't tear them up yet!)
- Refine by identifying abstract classes, inheritance, etc.
- Keep adding/refining until everyone on the team is satisfied.

How Can We Tell It Works?

- A neat technique: a **Scenario Walk-through**.
- Choose a plausible set of inputs for each scenario.
- Manually "execute" each scenario.
  - Start with initial input for scenario and find a class that has responsibility for responding to that input.
  - Trace through the collaborations of each class that participates in satisfying that responsibility.
  - Make adjustments as necessary.
  - Repeat until scenario has "stabilized" (i.e. no further adjustments necessary).

Example

- Consider the following description of a software system:

  You are developing a software system to facilitate restaurant reviews. Each restaurant corresponds to a certain price range, neighbourhood, and cuisines it serves. Restaurants that serve alcohol must have a licence, which they need to renew every year. The system should also report how long, on average, customers wait for take out in restaurants that offer take-out service. When reviewers leave a review for a restaurant, they must specify a recommendation (Thumbs Up or Thumbs Down) and can also leave a comment. An owner of a restaurant can respond to a review with a comment. All users of the system log in with their username. Users can choose to be contacted by email ...

Example

- Let's begin!

Each restaurant corresponds to a certain price range, neighbourhood, and cuisines it serves. Restaurants that serve alcohol must have a licence, which they need to renew every year. The system should also report how long, on average, customers wait for take out in restaurants that offer take-out service. When reviewers leave a review for a restaurant, they must specify a recommendation (Thumbs Up or Thumbs Down) and can also leave a comment. An owner of a restaurant can respond to a review with a comment. All users of the system log in with their username. Users can choose to be contacted by email and ...

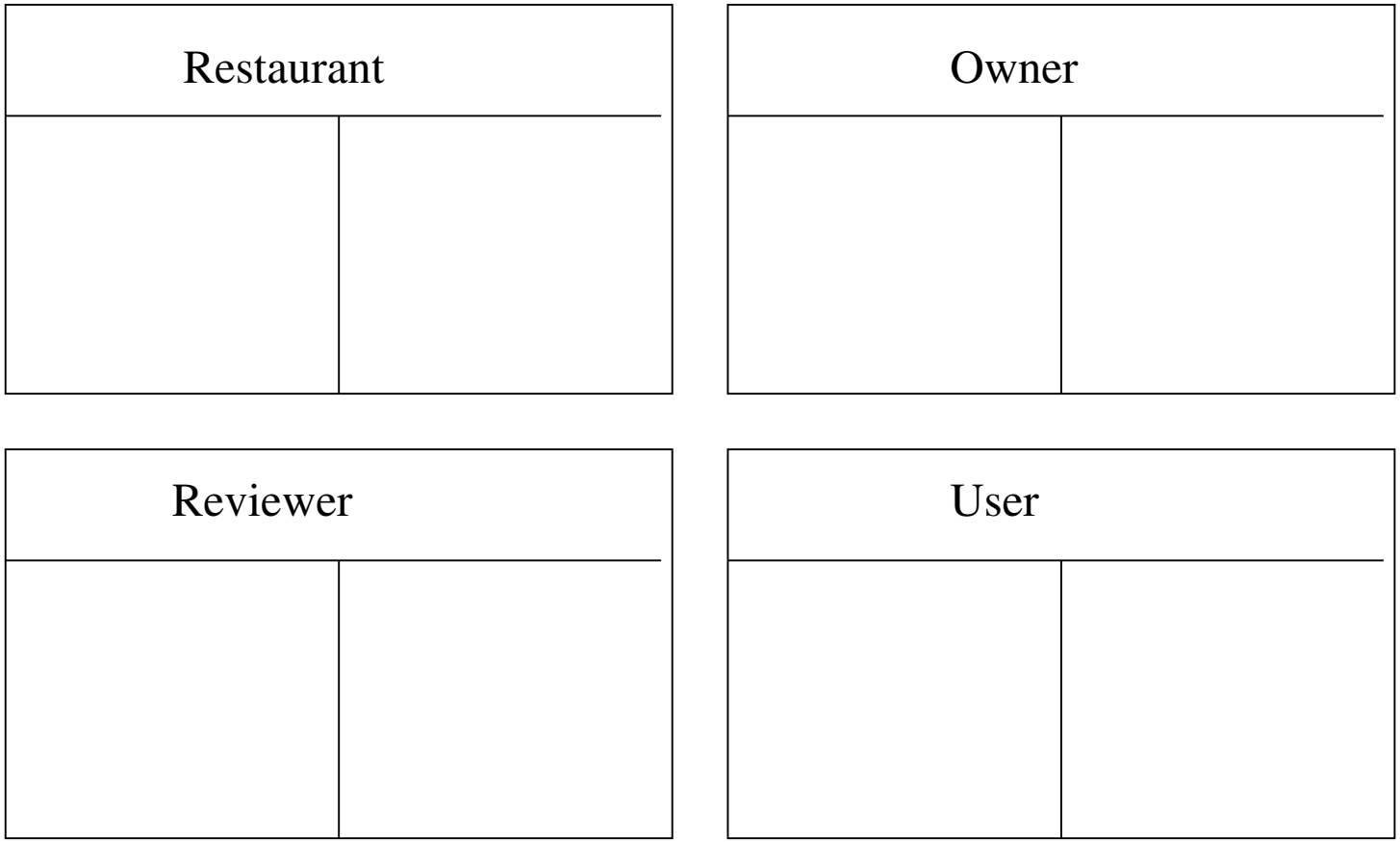
Example

- But which ones are the main players?

Each **restaurant** *corresponds to a certain price range, neighbourhood, and cuisines it serves. Restaurants that serve alcohol must have a licence, which they need to renew every year. The system should also report how long, on average, customers wait for take out in restaurants that offer take-out service. When* **reviewers** *leave a review for a restaurant, they must specify a recommendation (Thumbs Up or Thumbs Down) and can also leave a comment. An* **owner** *of a restaurant can respond to a review with a comment. All* **user**s *of the system log in with their username. Users can choose to be contacted by email ...*

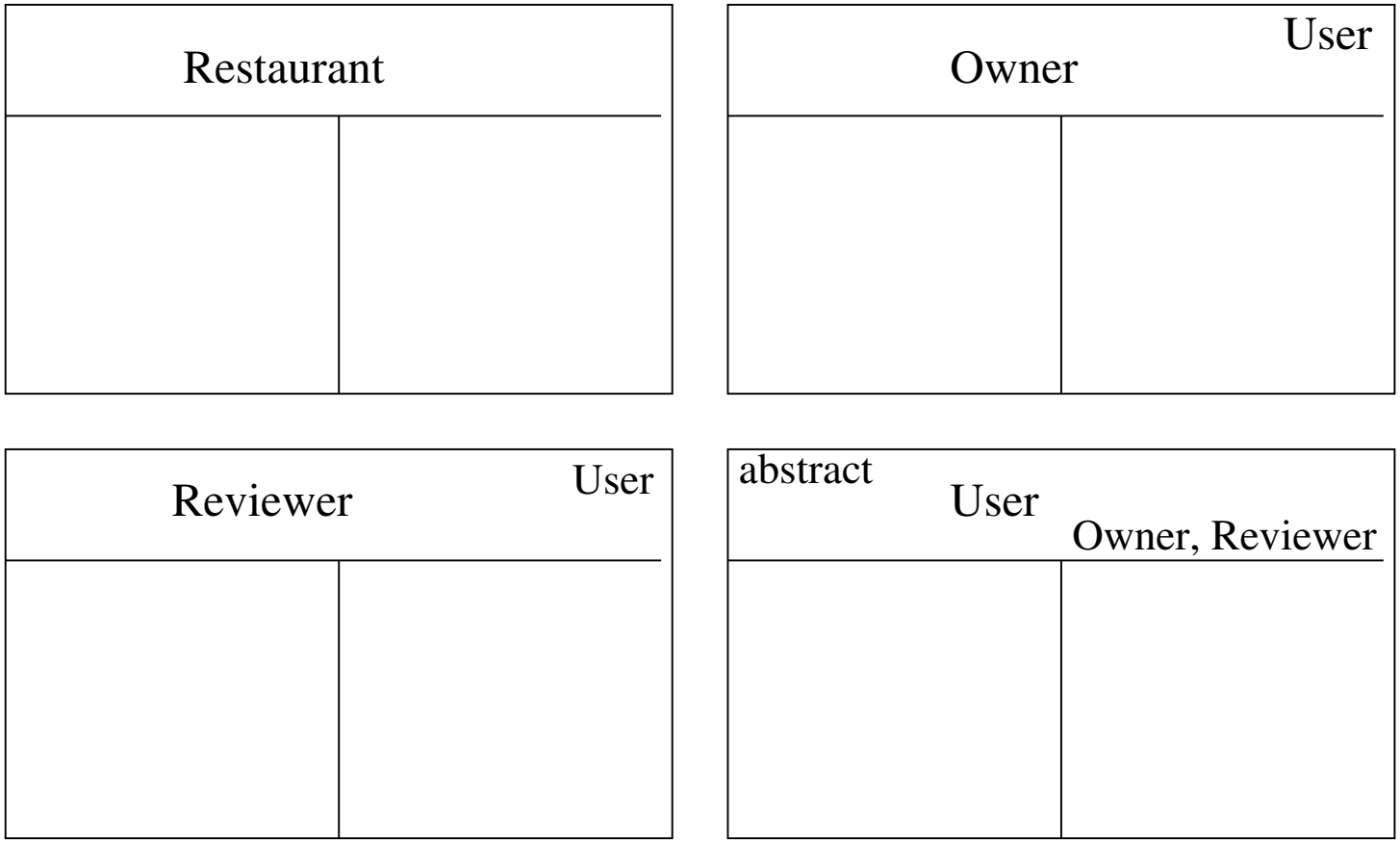
Example

OK, so we begin...



Example

As experienced designers, we immediately notice...



Example

- And what do they do?

Each **restaurant** *corresponds to a certain price range, neighbourhood, and cuisines it serves. Restaurants that serve alcohol must have a licence, which they need to renew every year. The system should also report how long, on average, customers wait for take out in restaurants that offer take-out service. When* **reviewers** *leave a review for a restaurant, they must specify a recommendation (Thumbs Up or Thumbs Down) and can also leave a comment. An* **owner** *of a restaurant can respond to a review with a comment. All* **user**s *of the system log in with their username. Users can choose to be contacted by email ...*

Example

Adding these responsibilities...

Restaurant	
renew license	

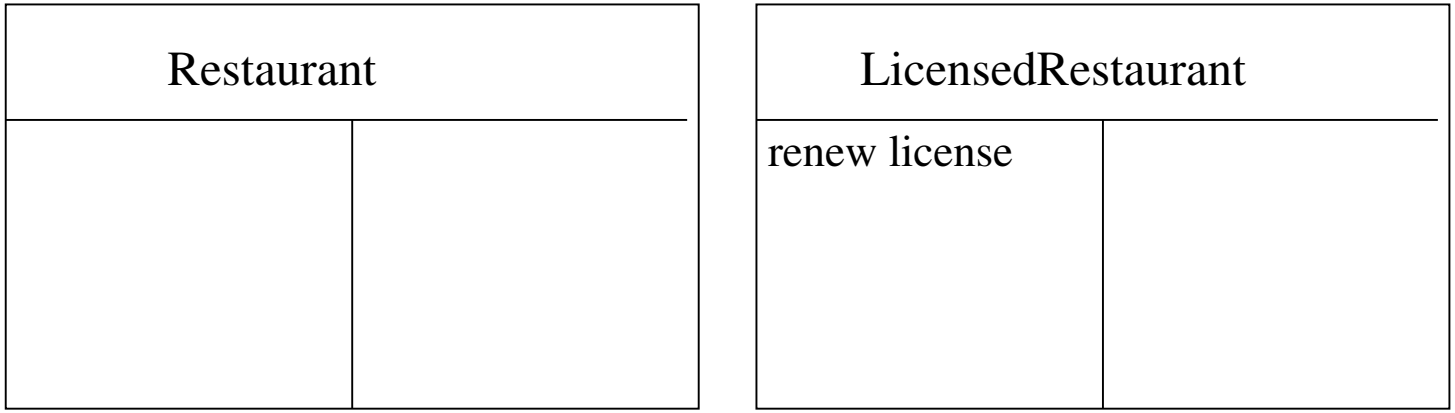
Owner	User
respond to review	

Reviewer	User
write review	

abstract User	Owner, Reviewer
log in	

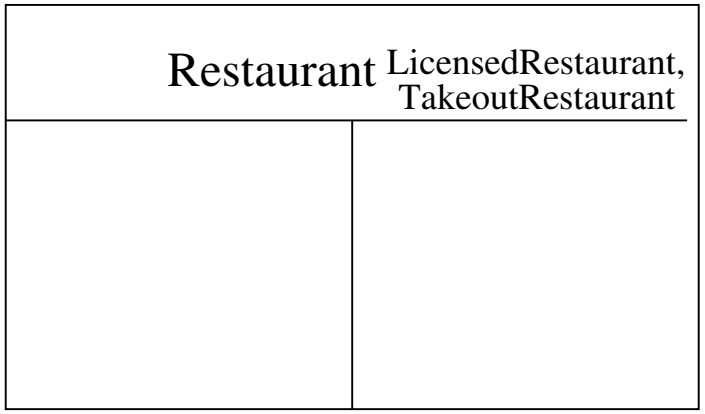
Example

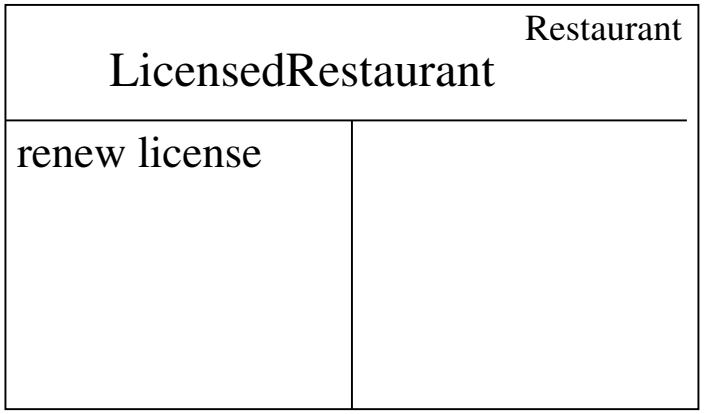
But wait! Not ALL restaurants have licences! We need a new type of Restaurant.

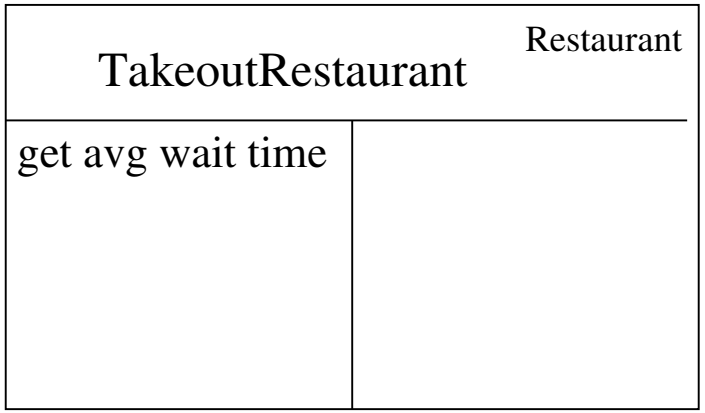


Example

But there's also take-out. OK, we need some hierarchy.

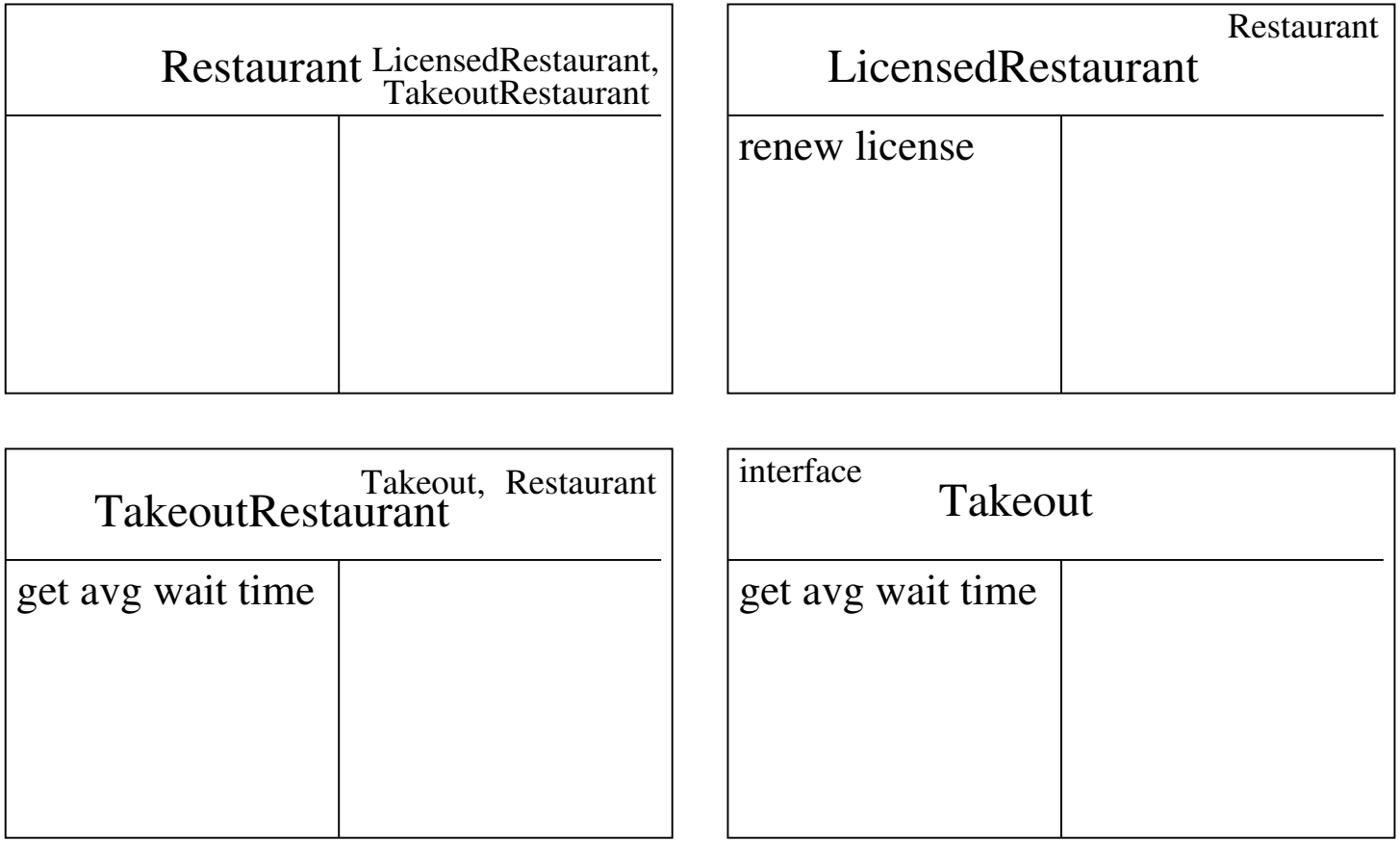






Example

But what if a restaurant is both a TakeoutRestaurant and a LicensedRestaurant?



Example

- So, what do they need to fulfil their responsibilities?

Each **restaurant** *corresponds to a certain price range, neighbourhood, and cuisines it serves. Restaurants that serve alcohol must have a licence, which they need to renew every year. The system should also report how long, on average, customers wait for take out in restaurants that offer take-out service. When* **reviewers** *leave a review for a restaurant, they must specify a recommendation (Thumbs Up or Thumbs Down) and can also leave a comment. An* **owner** *of a restaurant can respond to a review with a comment. All* **user**s *of the system log in with their username. Users can choose to be contacted by email ...*

Example

To write a review...

Looks like we need a Review object!

Review	
thumbsUp comment	

Owner	User
respond to review	

Reviewer	User
write review	Restaurant Review

Restaurant	