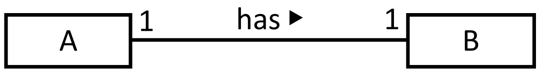
For each of the following diagrams, circle the two answers that correctly express the association relationship depicted.



a) Each A has one B

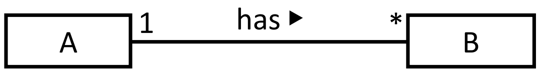
b) Each A has many Bs

c) Each A belongs to one B

d) Each B has one A

e) Each B has many As

f) Each B belongs to one A



a) Each A has one B

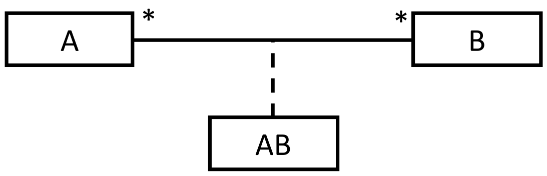
b) Each A has many Bs

c) Each A belongs to one B

d) Each B has one A

e) Each B has many As

f) Each B belongs to one A



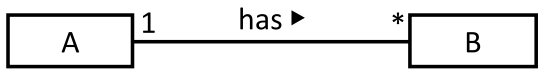
a) Each A has one B

b) Each A has many Bs

c) Each B has one A

d) Each B has many As

Circle the two answers that correctly express the following association relationship.



a) Each A has one B

b) Each A has many Bs

c) Each A belongs to one B

d) Each B has one A

e) Each B has many As

f) Each B belongs to one A

You have been asked to build a taxicab system similar to Uber. Create an object-oriented data model based on the following natural-language requirements. When deciding what to include, remember that the point here is that you are creating a design for your Rails MVC model. Your answer should take the form of a UML class diagram. Include only things that are specifically described.

- Include all relevant classes and attributes.
- Include all relevant associations and generalization relationships. Label all associations and association ends and include all multiplicities.

A driver can register one or more of their vehicles with the system. Vehicles have a make, model, and passenger capacity. A driver has some personal information including their SSN, name, date of birth, and address. Customers have a name and email. Customers can submit a request for a ride with a specific vehicle. In the request, the customer specifies the pick-up time and location (an address) and the destination location address.

Imagine that you have been hired to build a payroll system. Create an object-oriented data model based on the following natural-language requirements. When deciding what to include, remember that the point here is that you are creating a design for your Rails MVC model. Your answer should take the form of a UML class diagram. Include only things that are specifically described.

- Include all relevant classes and attributes.
- Include all relevant associations and generalization relationships. Label all associations and association ends and include all multiplicities.

A user has a first name, last name, SSN (social security number), and EID (employee identification number). Each user has a set of pay stubs. Each pay stub has pay-period start and end dates, a payment number, a payment amount (in cents), and a federal tax amount (in cents). Each user also has a set of direct-deposit bank accounts. Each direct-deposit bank account has the name of the bank, the bank's routing number, and the user's bank-account number. Each pay stub is deposited in one of the user's direct-deposit bank accounts.

Draw a UML class diagram that represents the three model classes given in Figure 1.

- Include all relevant classes and attributes. Don't include any "id" attributes (including foreign keys). You may also omit the "datetime" attributes that Rails provides by default.
- Include all relevant associations and generalization relationships. Label all associations and association ends and include all multiplicities.

```
# == Schema Information
#
# Table name: artists
#
#  id            :integer          not null, primary key
#  name          :string
#  year_founded  :integer
#  place_founded :string
#  about         :text
#  created_at    :datetime         not null
#  updated_at    :datetime         not null
#

class Artist < ApplicationRecord
    has_many :albums
    validates :year_founded, numericality: { less_than_or_equal_to: Date.today.year }
end
```

```
# == Schema Information
#
# Table name: albums
#
#  id            :integer          not null, primary key
#  title         :string
#  year_released :integer
#  genre         :string
#  artist_id     :integer
#  created_at    :datetime         not null
#  updated_at    :datetime         not null
#
# Indexes
#
#  index_albums_on_artist_id  (artist_id)
#

class Album < ApplicationRecord
  belongs_to :artist
  has_many :tracks
  validates :genre, inclusion: { in: ['Rock', 'R&B/HipHop', 'Pop', 'Country', 'Latin'] }
end
```

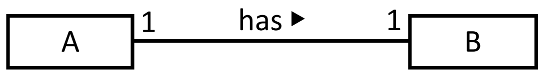
```
# == Schema Information
#
# Table name: tracks
#
#  id             :integer          not null, primary key
#  title          :string
#  track_number   :integer
#  length_seconds :integer
#  album_id       :integer
#  created_at     :datetime         not null
#  updated_at     :datetime         not null
#
# Indexes
#
#  index_tracks_on_album_id  (album_id)
#

class Track < ApplicationRecord
  belongs_to :album
end
```

Figure 1. Model classes for a music catalog application.

Solutions

For each of the following diagrams, circle the two answers that correctly express the association relationship depicted.



a) Each A has one B

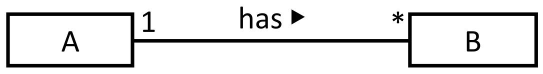
b) Each A has many Bs

c) Each A belongs to one B

d) Each B has one A

e) Each B has many As

f) Each B belongs to one A



a) Each A has one B

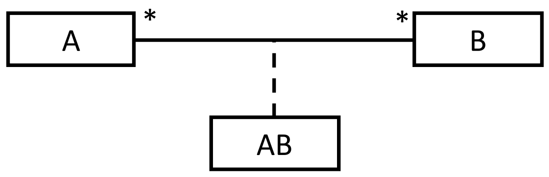
b) Each A has many Bs

c) Each A belongs to one B

d) Each B has one A

e) Each B has many As

f) Each B belongs to one A



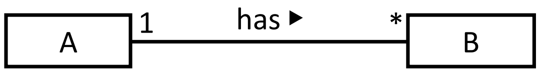
a) Each A has one B

b) Each A has many Bs

c) Each B has one A

d) Each B has many As

Circle the two answers that correctly express the following association relationship.



a) Each A has one B

b) Each A has many Bs

c) Each A belongs to one B

d) Each B has one A

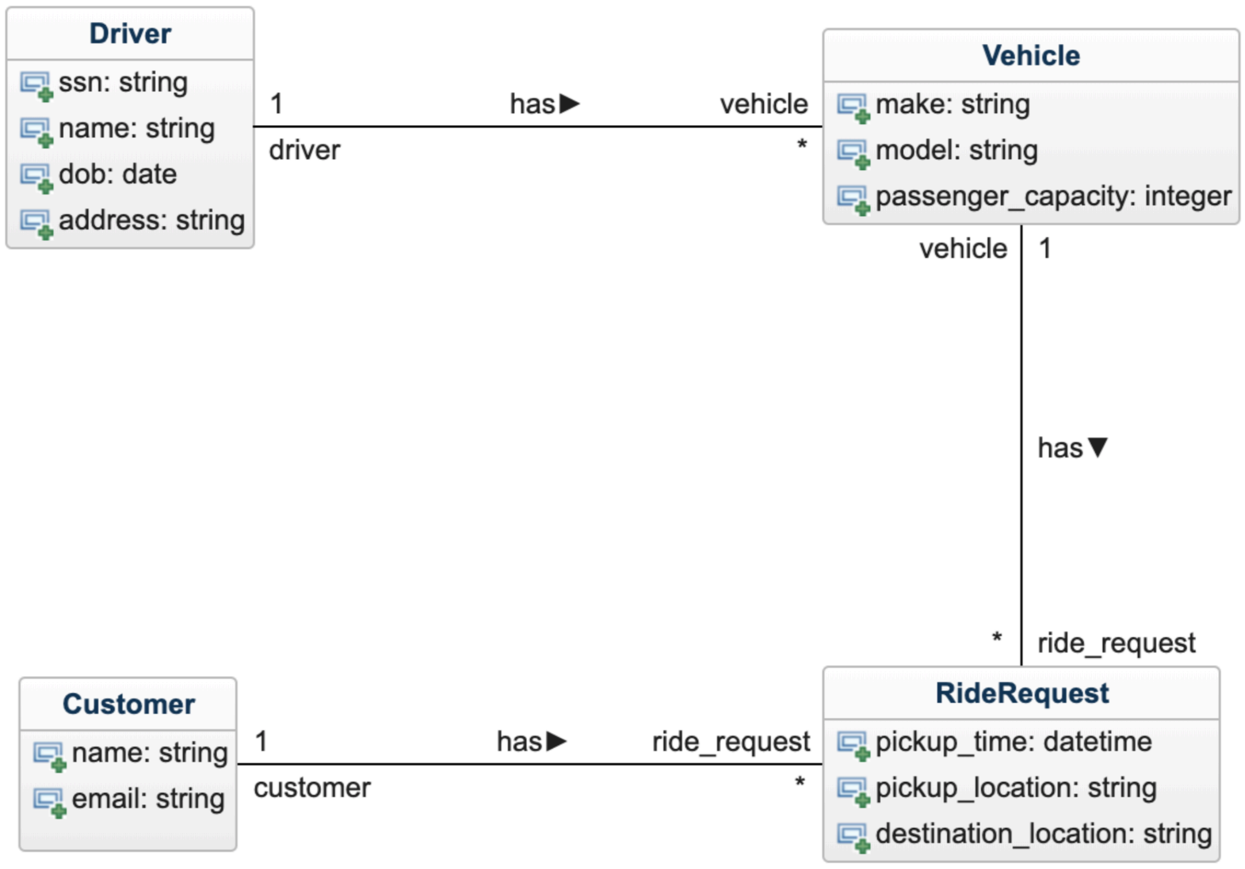
e) Each B has many As

f) Each B belongs to one A

You have been asked to build a taxicab system similar to Uber. Create an object-oriented data model based on the following natural-language requirements. When deciding what to include, remember that the point here is that you are creating a design for your Rails MVC model. Your answer should take the form of a UML class diagram. Include only things that are specifically described.

- Include all relevant classes and attributes.
- Include all relevant associations and generalization relationships. Label all associations and association ends and include all multiplicities.

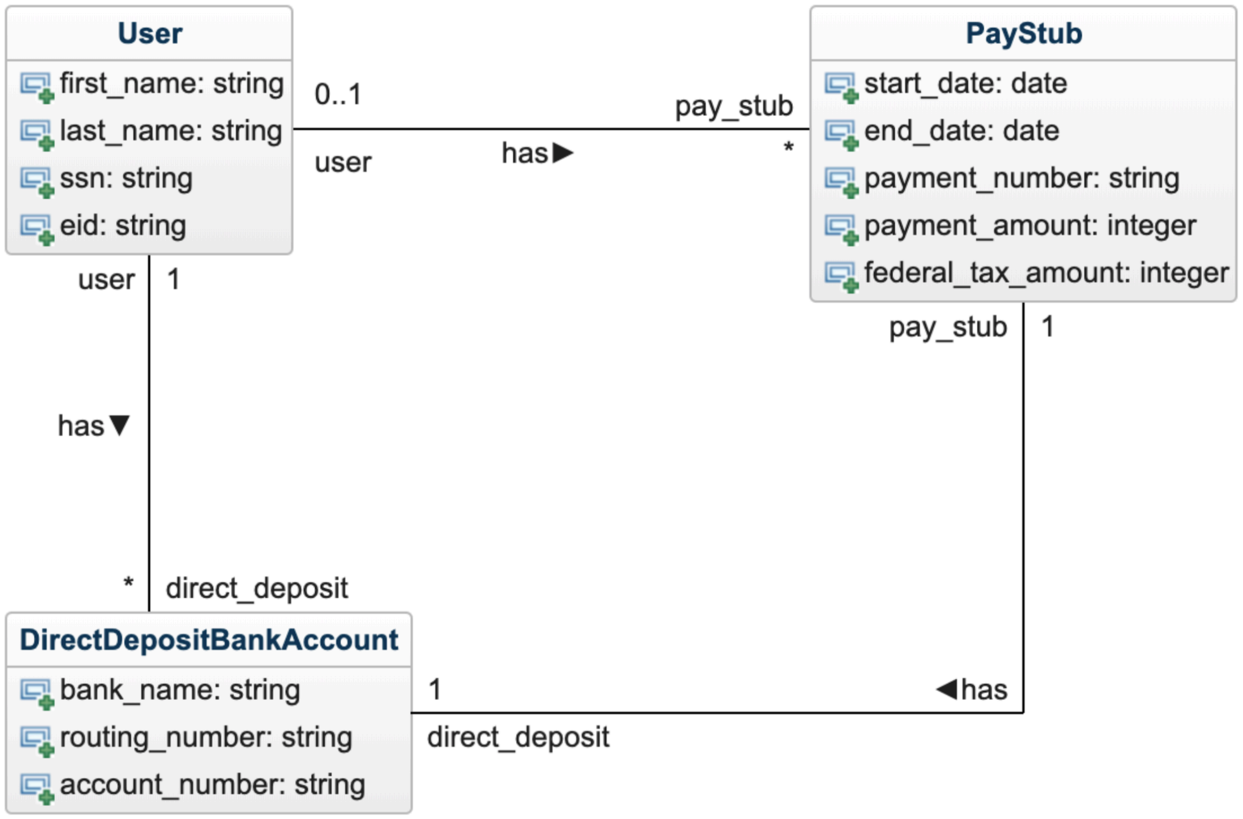
A driver can register one or more of their vehicles with the system. Vehicles have a make, model, and passenger capacity. A driver has some personal information including their SSN, name, date of birth, and address. Customers have a name and email. Customers can submit a request for a ride with a specific vehicle. In the request, the customer specifies the pick-up time and location (an address) and the destination location address.



Imagine that you have been hired to build a payroll system. Create an object-oriented data model based on the following natural-language requirements. When deciding what to include, remember that the point here is that you are creating a design for your Rails MVC model. Your answer should take the form of a UML class diagram. Include only things that are specifically described.

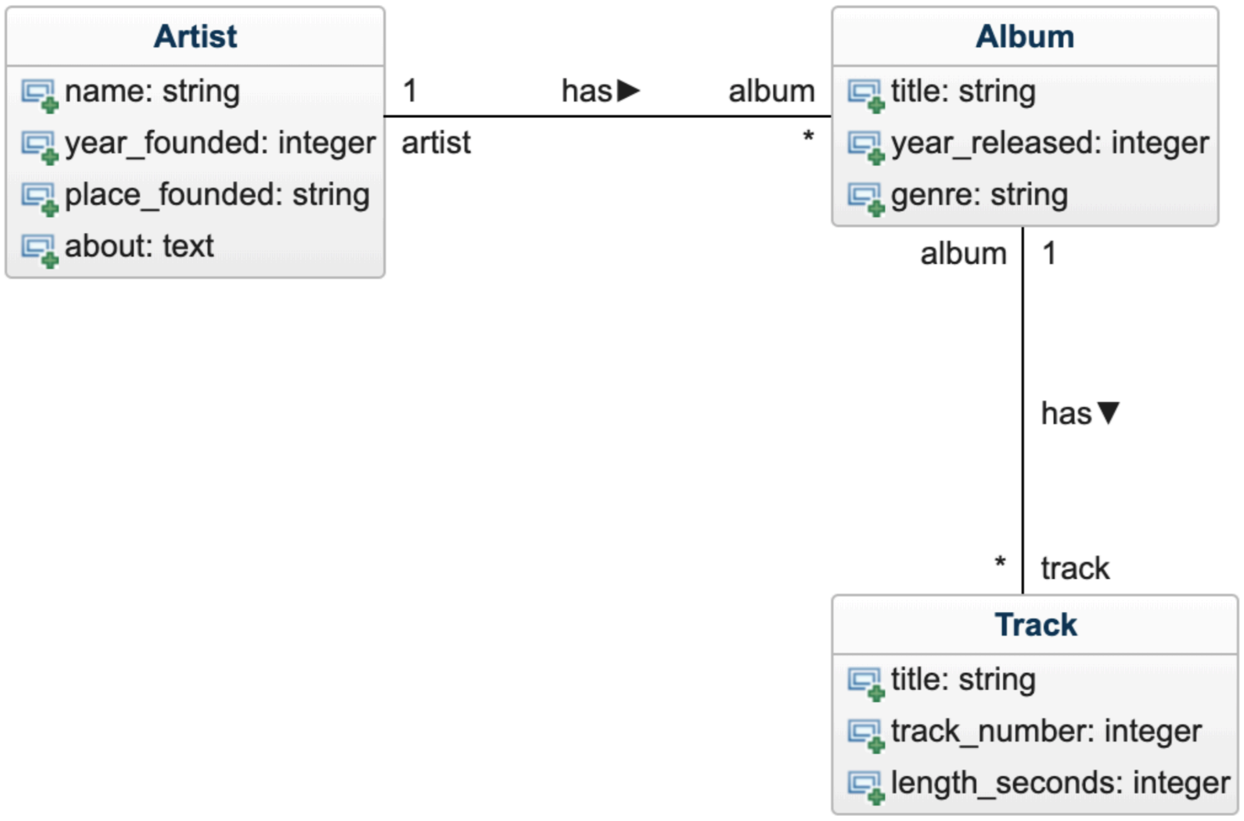
- Include all relevant classes and attributes.
- Include all relevant associations and generalization relationships. Label all associations and association ends and include all multiplicities.

A user has a first name, last name, SSN (social security number), and EID (employee identification number). Each user has a set of pay stubs. Each pay stub has pay-period start and end dates, a payment number, a payment amount (in cents), and a federal tax amount (in cents). Each user also has a set of direct-deposit bank accounts. Each direct-deposit bank account has the name of the bank, the bank's routing number, and the user's bank-account number. Each pay stub is deposited in one of the user's direct-deposit bank accounts.



Draw a UML class diagram that represents the three model classes given in Figure 1.

- Include all relevant classes and attributes. Don't include any "id" attributes (including foreign keys). You may also omit the "datetime" attributes that Rails provides by default.
- Include all relevant associations and generalization relationships. Label all associations and association ends and include all multiplicities.


Artist
name: string
year_founded: integer
place_founded: string
about: text
1
artist
has▶
album
*
Album
title: string
year_released: integer
genre: string
album
1
has▼
*
track
Track
title: string
track_number: integer
length_seconds: integer