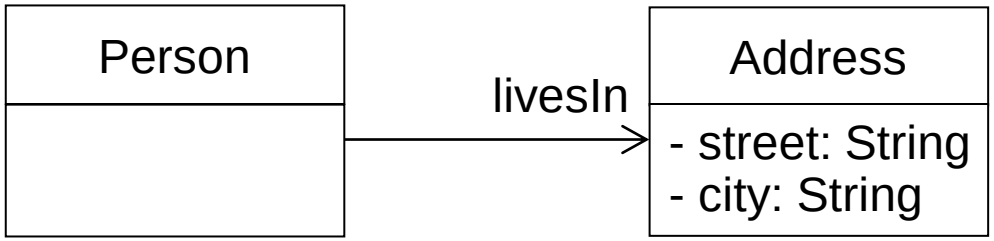
Navegación

- Indica si desde una clase podemos navegar a la otra a través de la asociación
- Se muestra con una flecha

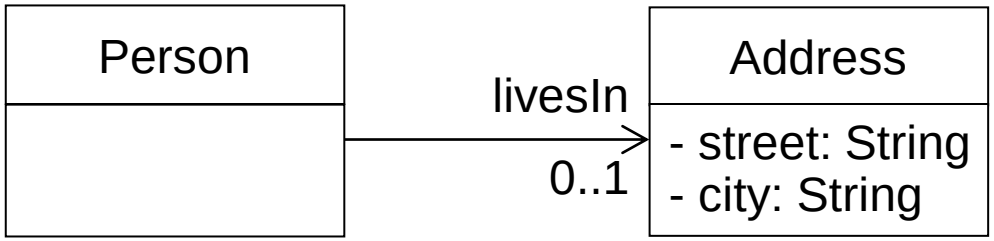



```
class Person {
  private Address livesIn;
}
```

```
class Address {
  private String street;
  private String city;
}
```

Navegación

- Indica si desde una clase podemos navegar a la otra a través de la asociación
- Se muestra con una flecha



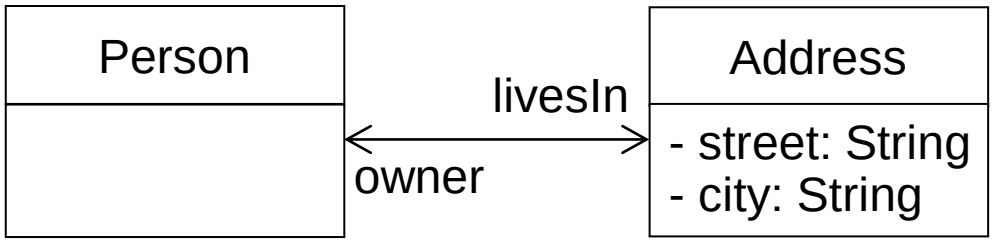

```
class Person {
  private Address livesIn;
}
```

```
class Address {
  private String street;
  private String city;
}
```

Ahora livesIn puede ser null

Navegación bidireccional

- Navegación entre ambas clases
- Equivale a omitir ambas flechas
  - Pero esto también puede significar que esa decisión de diseño aún no se ha tomado

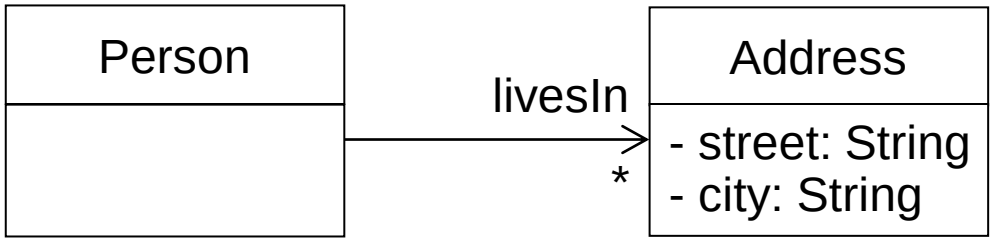



```
class Person {
  private Address livesIn;
}
```

```
class Address {
  private String street;
  private String city;
  private Person owner;
}
```

Navegación 1-a-muchos

- Una colección en un extremo
- La navegación bidireccional de muchos a muchos crearía colecciones en ambos extremos



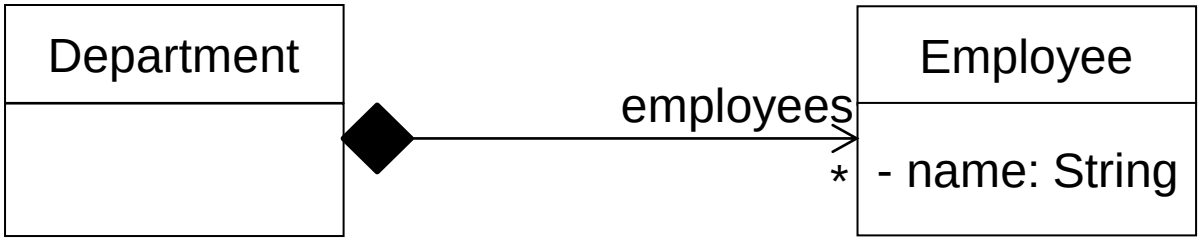

```
class Person {
  private Address[] livesIn;
}
```

```
class Address {
  private String street;
  private String city;
}
```

O mejor aún: alguno de los tipos de Java Collection (List, Set, etc)

Composición+navegación uni-dir

- Una colección en un extremo
- El código de las clases debe encargarse de la semántica de la composición:
  - El contenido no pertenece a más de un contenedor
  - Al eliminar el contenedor se elimina el contenido

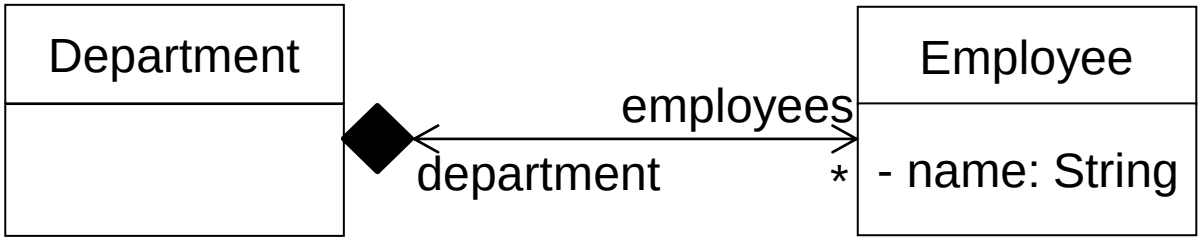



```
class Department {
  private Employee[] employees;
}
```

```
class Employee {
  private String name;
}
```

Composición + navegación bi-dir

- Una colección en un extremo y una referencia en el otro




```
class Department {
  private Employee[] employees;
}
```

```
class Employee {
  private String name;
  private Department department;
}
```