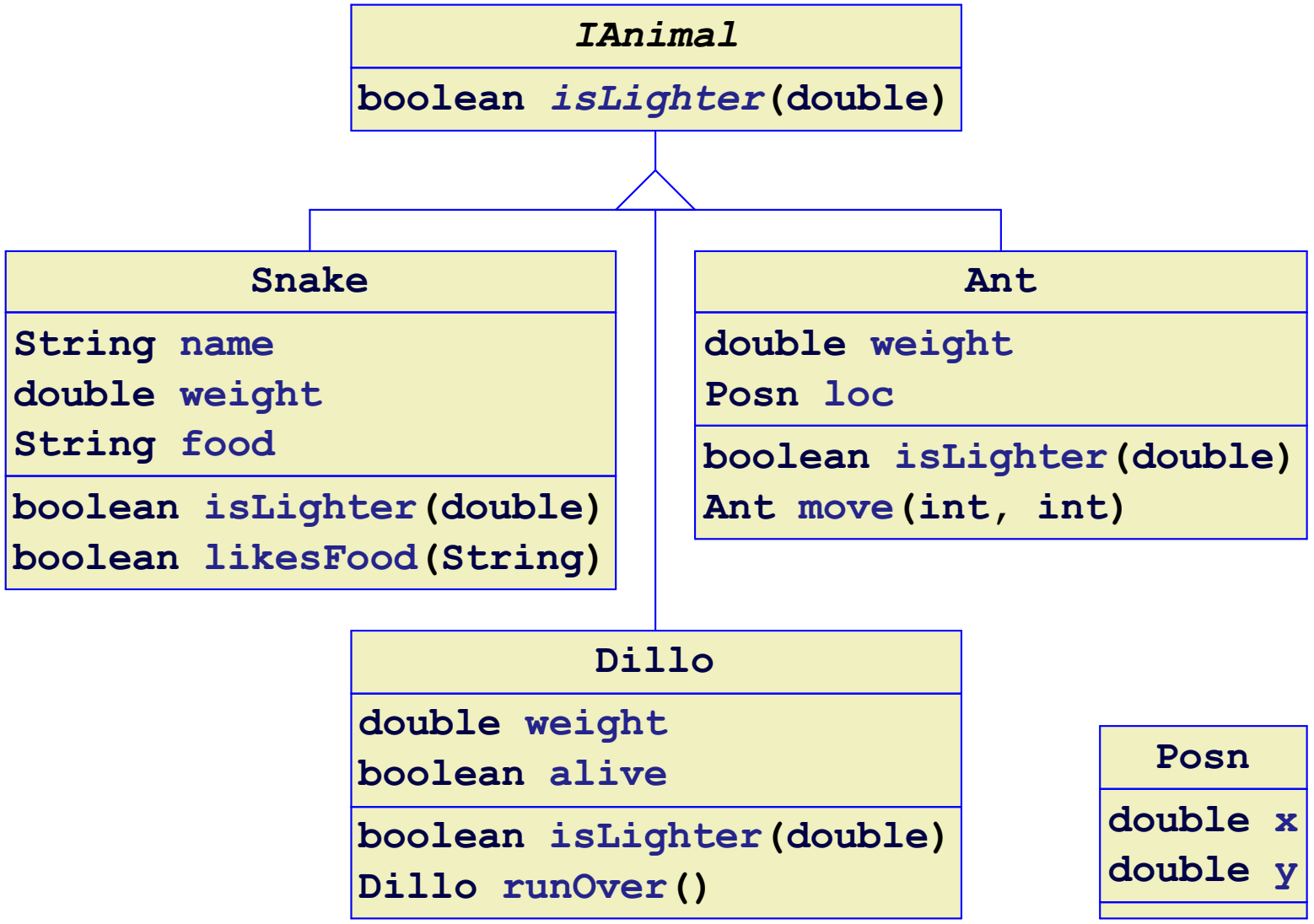
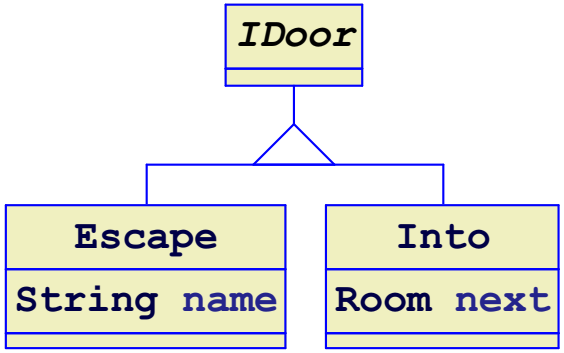
Class Diagrams



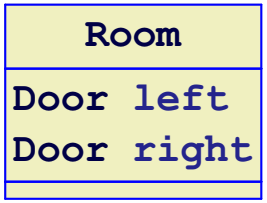
Maze

A maze consists of rooms and doors:

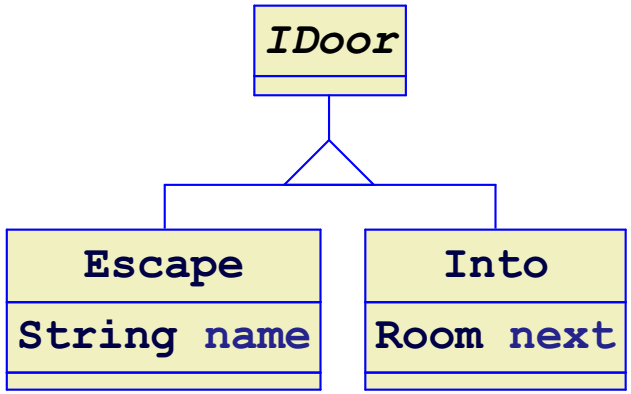
- A door is either
  - a door into a room
  - an escape to a particular place



- A room has two doors, left and right



Door Data Definition



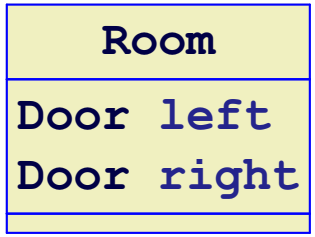

```
interface IDoor {
}

class Into implements IDoor {
  Room next;
  Into(Room next) {
    this.next = next;
  }
}

class Escape implements IDoor {
  String name;
  Escape(String name) {
    this.name = name;
  }
}
```

Copy

Room Data Definition




```
class Room {
  IDoor left;
  IDoor right;
  Room(IDoor left, IDoor right) {
   this.left = left;
   this.right = right;
  }
}
```

Copy

Examples

```
class Examples {
  IDoor meadow = new Escape("meadow");
  IDoor street = new Escape("street");
  Room ms = new Room(meadow, street);
  Room planets = new Room(new Escape("mars"),
                          new Escape("venus"));
  Room maze = new Room(new Into(ms),
                       new Into(planets));
}
```

Copy

Finding Paths

Implement the **`IDoor`** method **`canEscape`** that takes a string and returns a boolean indicating whether an escape with the given name is available

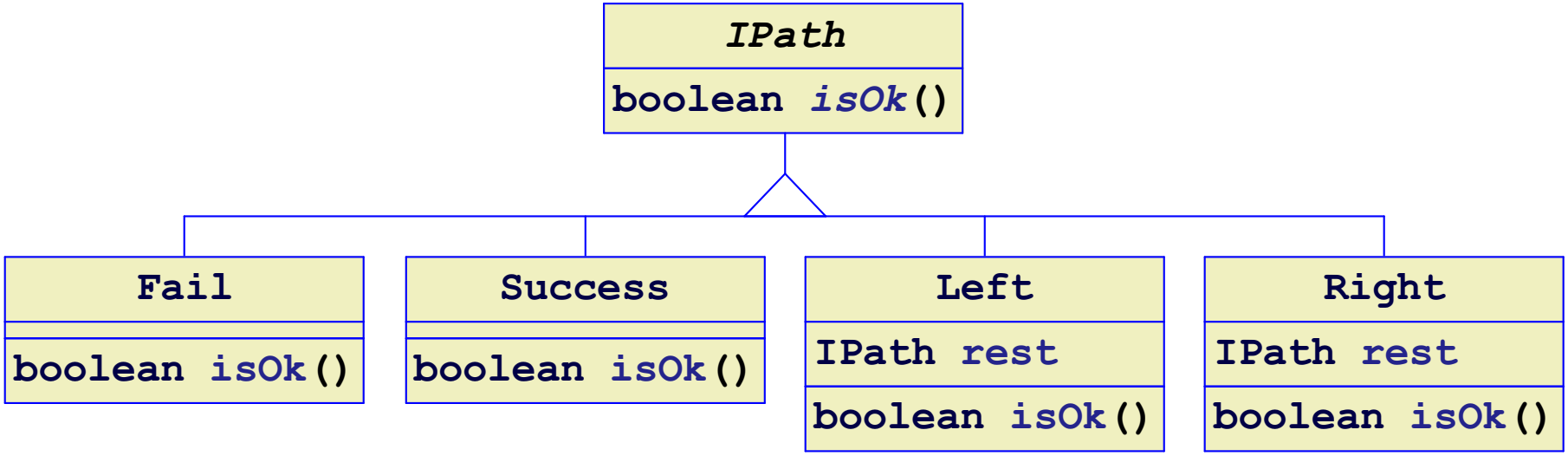
Replace the **`canEscape`** method with a **`escapePath`** method that takes a string and returns either a path of "left" and "right" leading to the escape, or a failure value

```
Path escapePath(String dest)
```

Paths

A path result is either

- failure
- immediate success
- left followed by a (succesful) path
- right followed by a (successful) path



Paths

```
interface IPath {
  boolean isOk();
}

class Fail implements IPath {
  Fail() { }
  public boolean isOk() { return false; }
}

class Success implements IPath {
  Success() { }
  public boolean isOk() { return true; }
}

class Right implements IPath {
  IPath rest;
  Right(IPath rest) { this.rest = rest; }
  public boolean isOk() { return true; }
}

class Left implements IPath {
  IPath rest;
  Left(IPath rest) { this.rest = rest; }
  public boolean isOk() { return true; }
}
```

Copy