Bringing a UML to Life

In this task, your goal is to **translate a UML class diagram into working Java code exactly as shown**. No extra creativity, no missing pieces. Just clean, correct object-oriented programming.

Think of this as **connecting the dots** between theory (UML) and practice (Java).

Your Mission

You are given a UML diagram that describes:

- Classes
- Interfaces
- Inheritance
- Implementation
- Composition & Aggregation
- Attributes and Methods
- Relationships between objects

Your task is to **implement this diagram in Java exactly as it is**:

1. **Keep everything in one Java file (i.e. in App.java)**
2. Follow the UML **exactly**
3. Do **not** add extra fields or methods
4. A `main` method (App.java) has been provided to you**, use that**

How to Read the UML (Quick Guide)

Before coding, understand what the diagram is telling you:

1 Classes vs Interfaces

- Boxes marked with **C** → `class`
- Boxes marked with **I** → `interface`

2 Inheritance (`extends`)

- A **solid arrow** pointing upward means:
    - The child class **extends** the parent class
    - Example: `Human extends Mammal`

3 Interface Implementation (`implements`)

- A **dashed arrow** pointing to an interface means:
  - The class **implements** that interface
  - Example: `Dog implements Domesticated`

4 Methods

- Methods shown in the UML **must exist**
- Return types and method names must match
- Method bodies can be simple (e.g., `System.out.println(...)`)

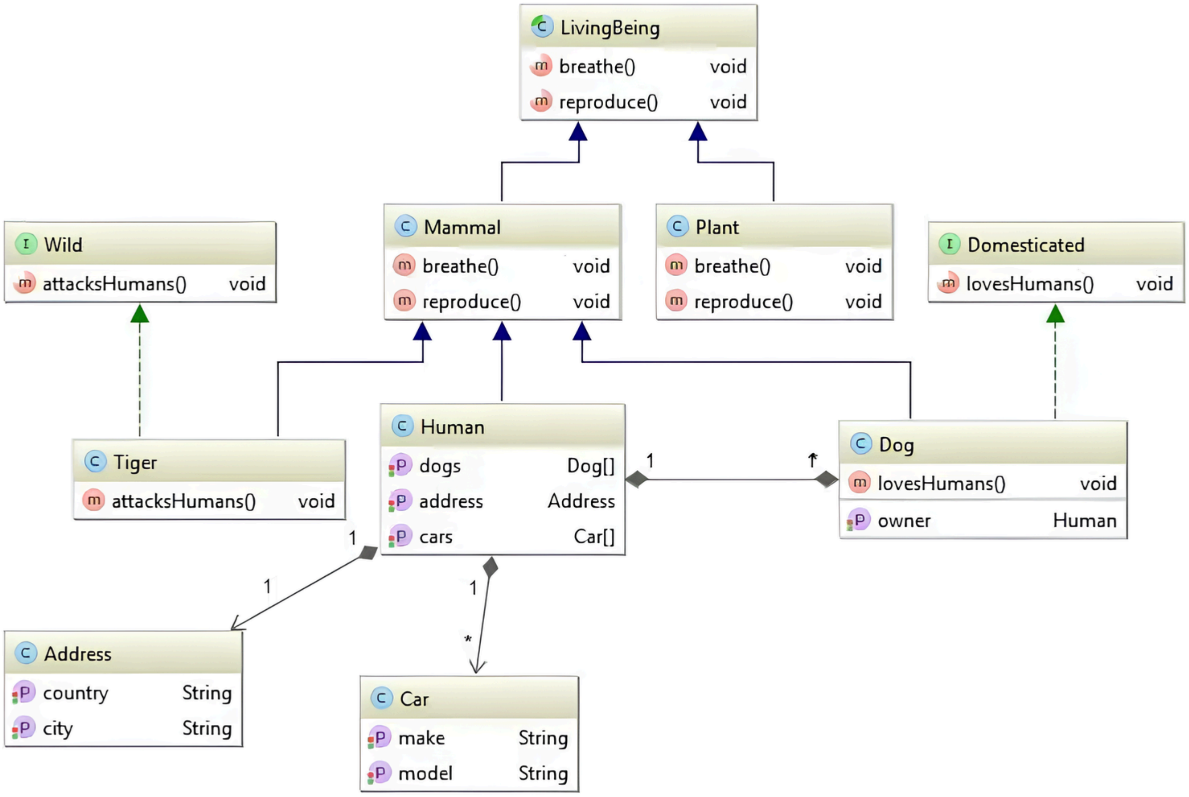
5 Attributes

- Attributes must:
  - Have the **correct data type**
  - Be placed in the **correct class**
- Example:
  - `Human` has `Dog[] dogs`
  - `Dog` has `Human owner`

6 Relationships (Very Important!)

- **Human → Address**: one-to-one; one human can only have one address
- **Human → Car**: one-to-many; one human can own multiple cars
- **Human ↔ Dog**: bidirectional association
- These relationships must exist **in code**, not just in comments.

UML Diagram to Follow



App.java

```
public class App
{
    public static void main(String[] args) {
        Address address = new Address("Finland", "Helsinki");

        Car car1 = new Car("Toyota", "Corolla");
        Car car2 = new Car("Tesla", "Model 3");
        Car[] cars = {car1, car2};

        Dog dog1 = new Dog();
        Dog dog2 = new Dog();
        Dog[] dogs = {dog1, dog2};

        Human human = new Human(dogs, address, cars);

        dog1.owner = human;
        dog2.owner = human;

        human.breathe();
        human.reproduce();

        dog1.lovesHumans();

        Tiger tiger = new Tiger();
        tiger.attacksHumans();

        System.out.println("\nHuman lives in: "
            + human.address.city + ", " + human.address.country);
        System.out.println("Human owns " + human.cars.length + " cars:");
        for (Car car : human.cars) {
            System.out.println("- " + car.make + " " + car.model);
        }

        System.out.println("Human owns " + human.dogs.length + " dogs.");
        System.out.println("Dog's owner is a Human: " + (dog1.owner ==
human));
    }
}
```