Programming II

第11回 抽象クラス（第11章前半）

塩澤秀和

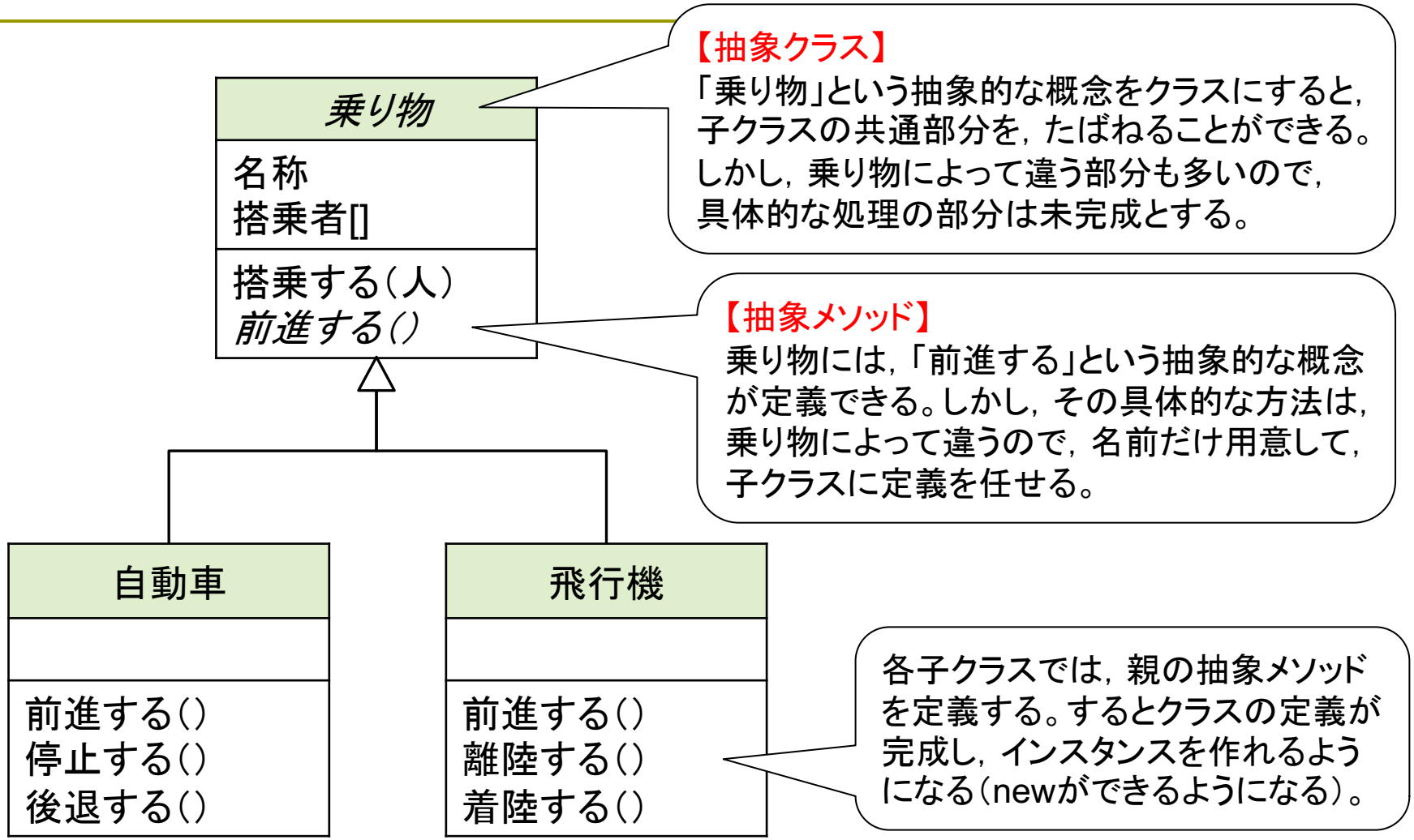
11.1「継承の材料」としてのクラス

- 「継承の材料」としてのクラス（p.408）
  - 他のクラスの「親」としてだけ働くクラスを定義したい
  - 親クラスとしての役割しかない
    ⇒ 自分のクラスのインスタンスは作らない（作らせない）

- なんのために？
  - 同種のクラスの共通部分をまとめて，実装の無駄を省く
  - 子クラスの追加に備えて，共通する機能を実装しておく
  - 同種のクラスの使用方法を，親クラスによって共通化する

- 「多態性」と組み合わせることで効果を発揮する

11.2 抽象クラスの考え方



※ 抽象クラスや抽象メソッドの名前は，斜体(イタリック体)で表す

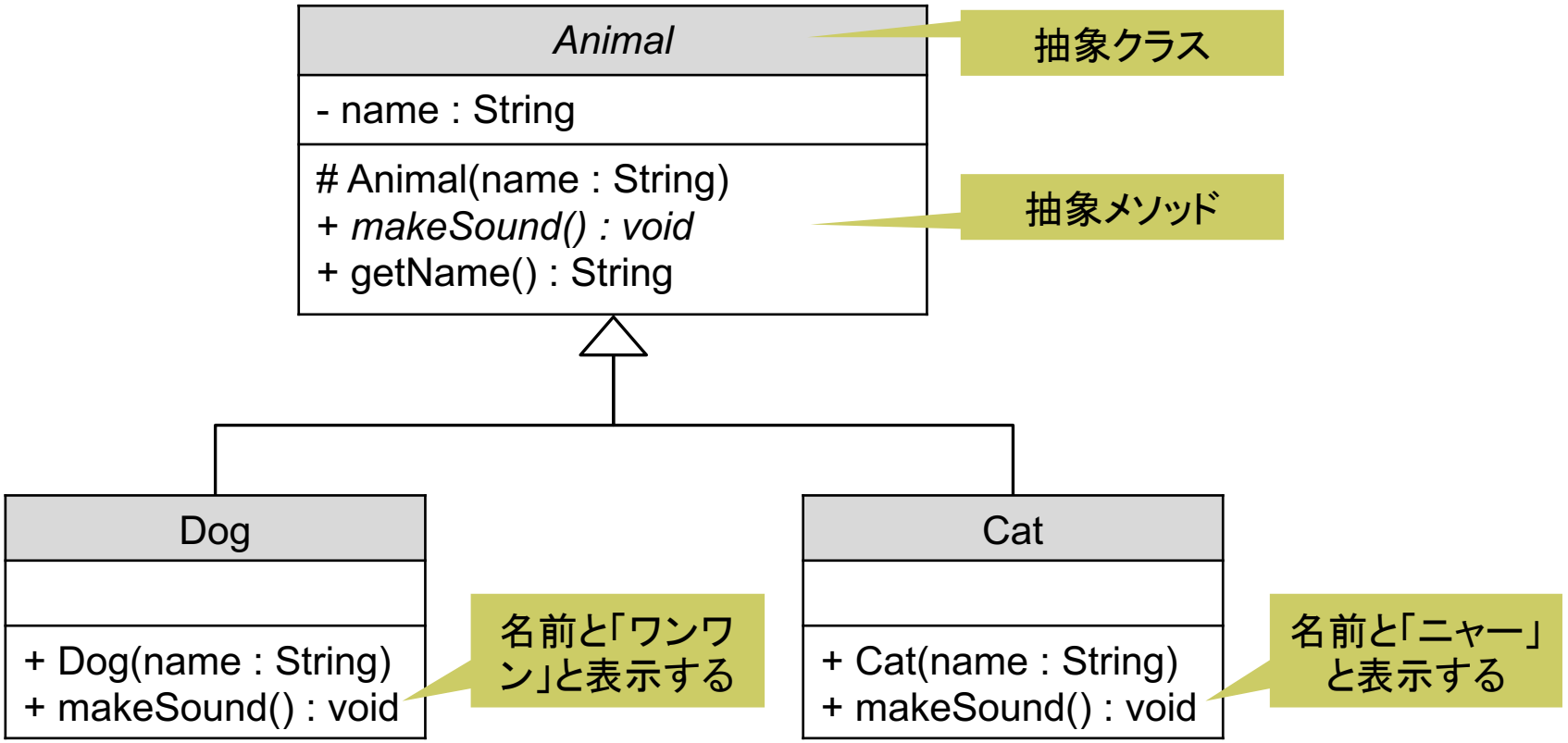
11.3 抽象クラスと抽象メソッド

- 抽象クラス（p.426）
  - 親クラスとしてだけ働き，newできない（させない）クラス
  - 修飾子abstractをつけて定義する（意味は「抽象的」）
  - 例： public abstract class Vehicle { … }

- 抽象メソッド（p.425）
  - 子クラスでオーバーライドされることを前提として，名前と引数しか決めずに，具体的な処理は未定のメソッド
  - 修飾子abstractをつけ，処理はなしでセミコロンだけ書く
  - 例： public abstract void moveForward() ;
  - 文法事項：抽象メソッドが1つでもあるクラスは抽象クラスにしなければならない ← 未完成なのだから当たり前

11.4 抽象クラスの演習

- 第9回の例題の改良



さらに，Dogを継承したShibaInuクラスと，Animalを継承したHamsterクラスを適当に（鳴き声などを適当に）定義してみよう

11.5 抽象クラスの演習（続き）

```
public class Main {
    public static void main(String [] args) {
        Animal hana = new ShibaInu("ハナ");
        hana.makeSound();

        Animal [] animals = { new Dog("ポチ"), new Cat("タマ"),
                              hana, new Hamster("シロ") };
        for (Animal a : animals) { a.makeSound(); }

        System.out.println("犬は, " + howManyDogs(animals) + "匹です");
    }

    public static int howManyDogs(Animal[] animals) {
        int n = 0;
        // Dogクラスのインスタンスの個数を数える
        return n;
    }
}
```

演習

- 11.4～11.5
  - プログラムを完成させよ
  - 最終的なソースコードと実行結果を提出

第11回 まとめ

- 抽象クラス
- 抽象メソッド
- abstract