抽象クラス

応用編4日目

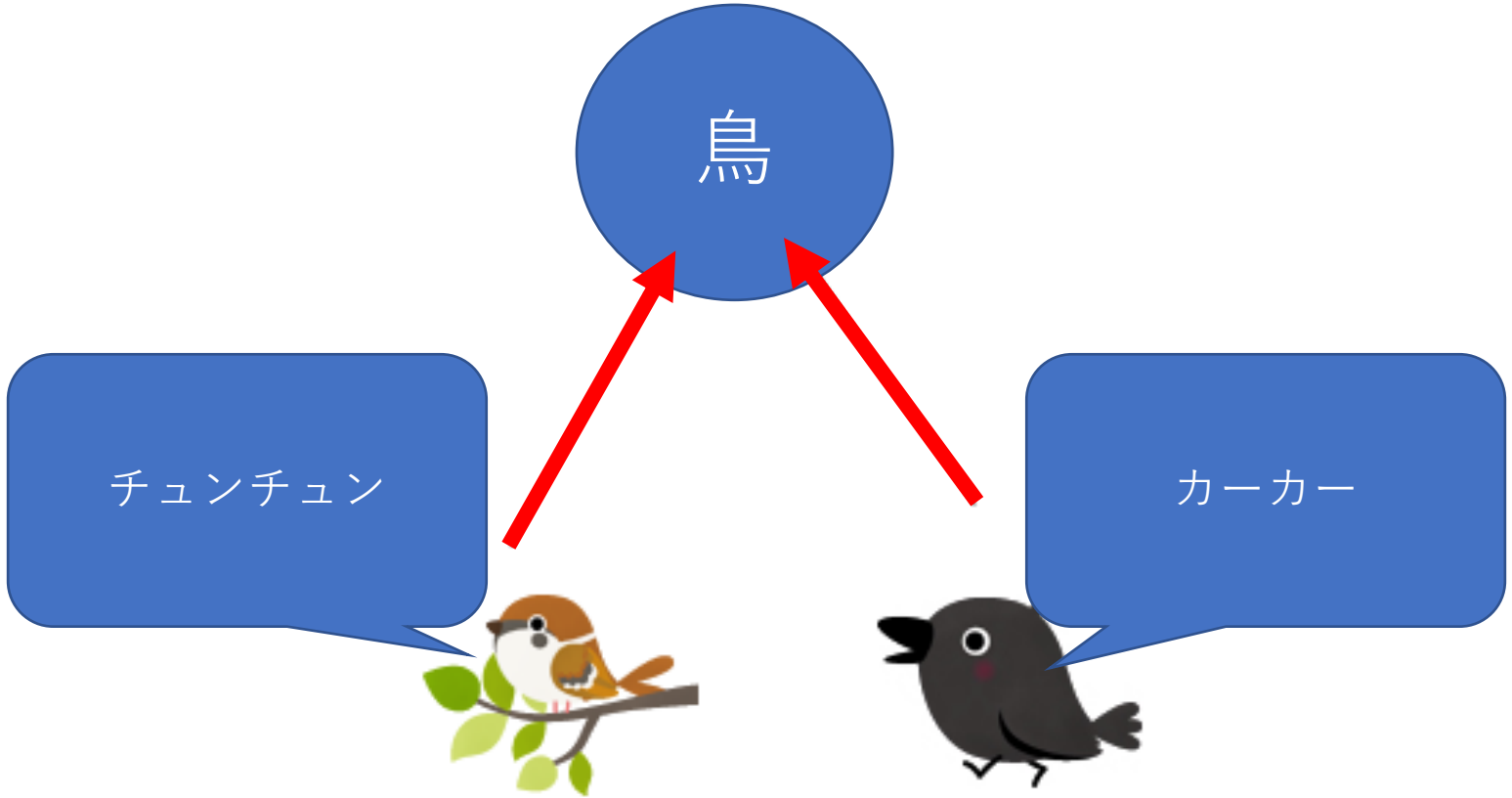
抽象クラス

- クラスの定義の前に**abstract**修飾子　→　**抽象クラス**
- 抽象クラス自身ではインスタンスを生成しないクラス
- 例えば、「カラス」や「すずめ」という鳥は存在するが、「鳥」という名前の鳥は存在しない
- つまり、「鳥」というのは抽象的な概念であり、実在しない

抽象クラスの定義

```
abstract class (クラス名){
    …
}
```

「鳥」は、あくまでも抽象的な概念



抽象メソッド

- 抽象クラスは**抽象メソッド**を持つ
- **abstract**という修飾子がついたメソッド。実装はない。
- **実装は抽象クラスを継承したサブクラス**で行われる

抽象メソッドの定義（抽象クラス内）

```
abstract (戻り値)(メソッド名)(引数);
```

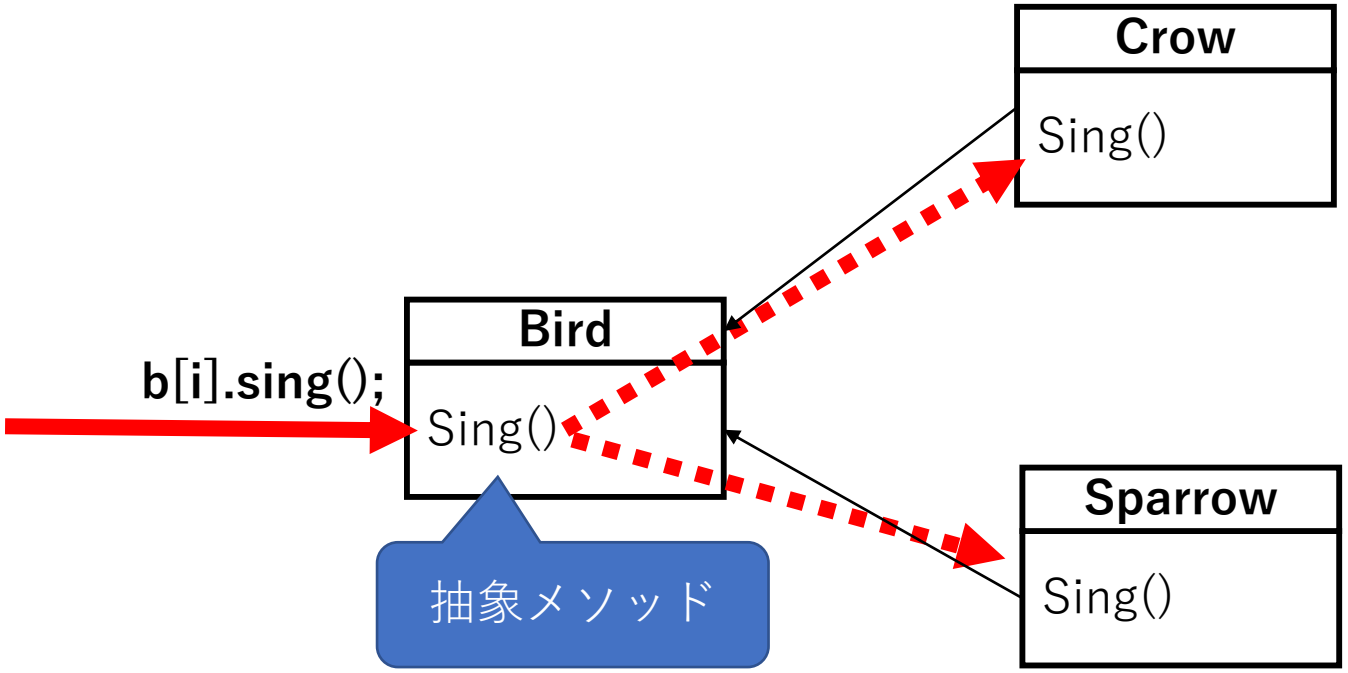
同一名・同一引数
戻り値の型も同一のメソッド

抽象メソッドの実装（サブクラス内）

```
override (戻り値)(メソッド名)(引数)
{
   …
}
```

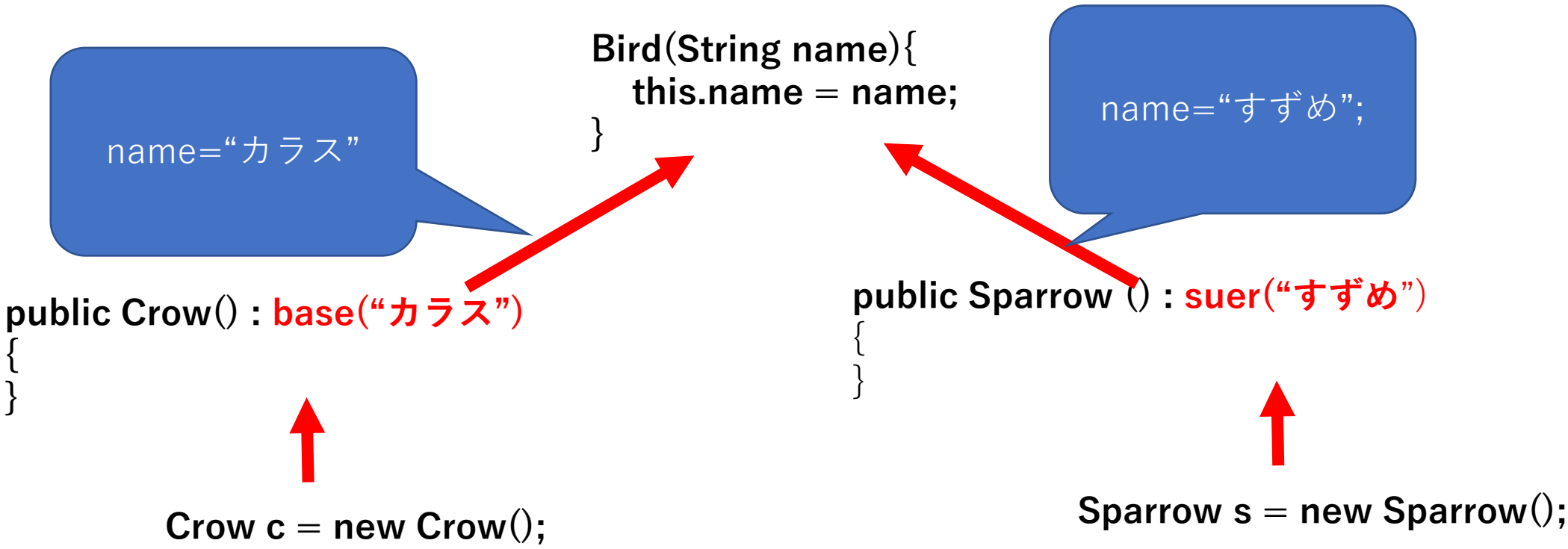
抽象クラスとしてインスタンスを持つ

- **SampleEx402**参照



親クラスのコンストラクタの呼び出し

- **base**を用いると、親クラスのコンストラクタを呼び出すことが出来る



抽象プロパティ

- 抽象クラスで定義する抽象的なプロパティ
- 実装はサブクラスで**override**して記述する

```
// スーパークラス（抽象プロパティ持つ）
abstract class VectorBase
{
    // 抽象プロパティ
    public abstract double X
    {
        set;
        get;
    }
    public abstract double Y
    {
        set;
        get;
    }
}
```

```
class Vector : VectorBase
{
    …
    public override double X
    {
        set { x = value; }
        get { return x; }
    }
    public override double Y
    {
        set { y = value; }
        get { return y; }
    }
}
```

抽象クラスを用いることのメリット

- 共通の性質を持つ新しいクラスを追加しやすい
  - カラス・すずめ　→　ハト・つばめなど
- 複数の似た性質を持つクラスを一元管理しやすい