# Java 泛型通配符 ? super 和 ? extends的區別

泛型通配符在增強代碼靈活性和類型安全性上的應用

# 泛型通配符簡介

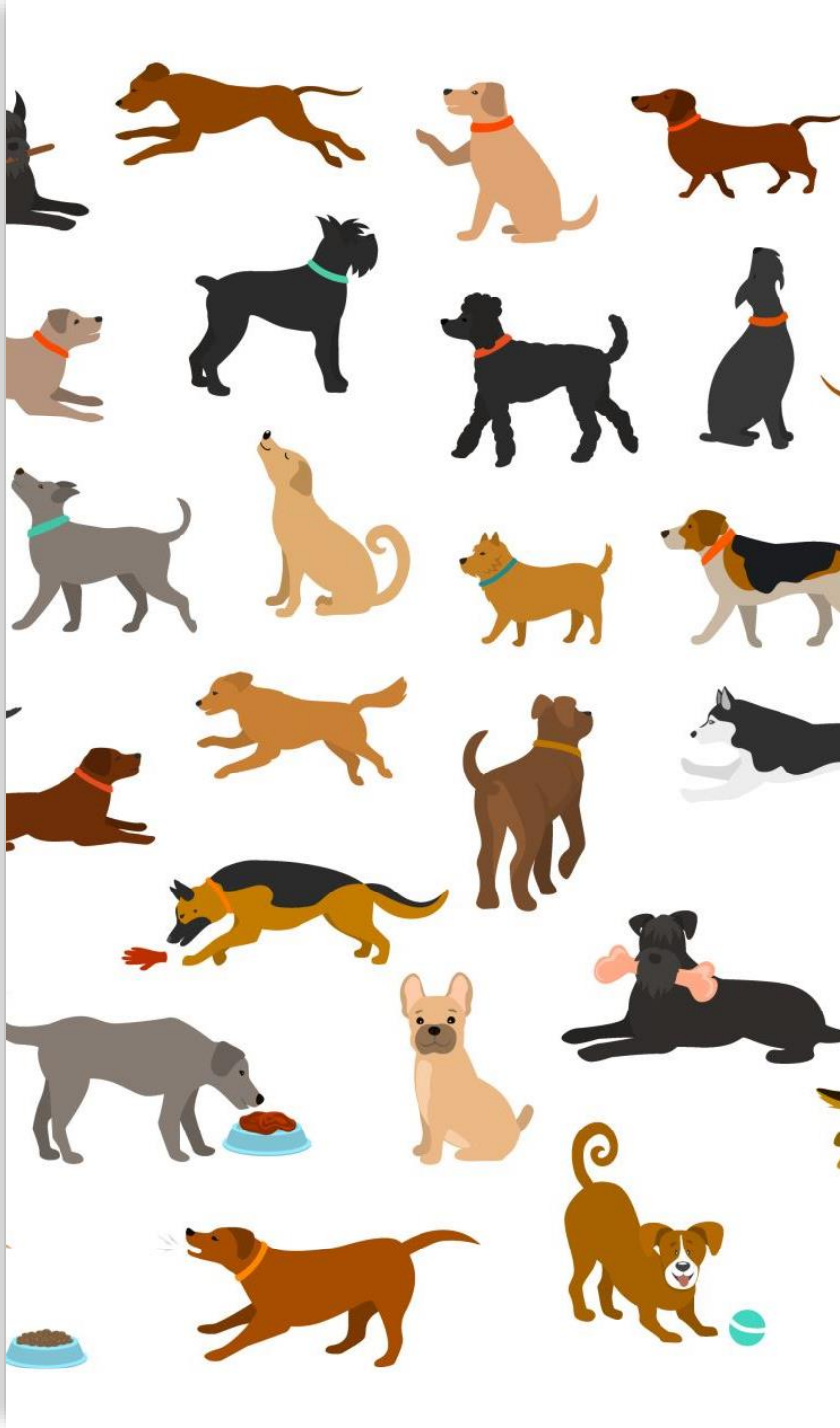
- ? extends：用於上界通配，表示某類的子類或本身
- ? super：用於下界通配，表示某類的父類或本身

# List<? super Dog> 的特性

- 含義：列表可以是 Dog 或其父類（如 Animal、Object）

- 插入元素：可以添加 Dog 或其子類型對象
  - 例如：animals.add(new Dog()) 或 animals.add(new Bulldog())

- 讀取元素：只能讀取為 Object 類型
  - 例如：Object obj = animals.get(0)



# List<? extends Dog> 的特性

- 含義：列表可以是 Dog 或其子類類型（如 Bulldog ）
- 插入元素：不能添加任何元素（除了 null ）
- 讀取元素：可以讀取為 Dog 類型
  - 例如：Dog dog = dogs.get(0)

# ? super Dog  vs ? extends Dog

| 特性 | List<? super Dog> | List<? extends Dog> |
| --- | --- | --- |
| 允許的列表類型 | List<Dog>，<br>List<Animal>，<br>List<Object> | List<Dog>，<br>List<Bulldog> 等子類 |
| 插入元素 | 可以插入 Dog 或其子類型 | 不能插入（只能插入 null） |
| 讀取元素 | 只能讀取為 Object | 可以讀取為 Dog |
| 使用場景 | 適合插入元素的情況 | 適合讀取元素的情況 |

# 何時使用 ? super 與 ? extends

- ? super：適合需要添加元素到集合中的場景
  - 例如：將新對象添加到集合中時

- ? extends：適合需要從集合中讀取元素的場景
  - 例如：只需要讀取元素並對其進行操作時

# 小結

- ? super T：適合「寫入」操作，可以插入 T 或 T 的子類，但讀取時只能得到 Object。

- ? extends T：適合「讀取」操作，確保能安全讀取 T 或其子類型，但不允許插入新元素。

- 根據具體操作選擇合適的通配符，以提高代碼的靈活性和安全性。

# 問題與討論

- 什麼場景下需要使用泛型通配符？

- 如何選擇？SUPER 或？EXTENDS 以適應不同的應用需求？