Bài 64: Lớp generic

- ✓ Khái niệm
- ✓ Cú pháp tổng quát
- ✓ Giới hạn trên, dưới
- ✓ Minh họa, bài tập thực hành

Khái niệm

- Một kiểu generic là một lớp hoặc interface chứa các kiểu đã được tham số hóa
- Mục đích của generic:
  - Tái sử dụng lại cùng chức năng cho nhiều kiểu dữ liệu khác nhau
  - Cho phép kiểm soát các kiểu đang được sử dụng
  - Code dễ đọc
- Ví dụ lớp ArrayList<T> là một lớp điển hình ta đã sử dụng từ lâu

Ví dụ

➢Ví dụ sử dụng lớp generic ArrayList<T>:

```
// dùng kiểu generic ArrayList<T> để lưu các giá trị kiểu String
ArrayList<String> friends = new ArrayList<>();
friends.add("Nhung");
friends.add("Linh");
friends.add("Khánh");
friends.add("Loan");

// lưu giá trị số thực:
ArrayList<Double> doubles = new ArrayList<>();
doubles.add(1.25);
doubles.add(3.12456);
doubles.add(6.1454);
doubles.add(99.899);

// lưu các đối tượng của lớp Student
ArrayList<Student> students = new ArrayList<>();
students.add(new Student("B25DCCN100", "Ngô Thùy Linh"));
students.add(new Student("B25DCCN101", "Nguyễn Phương Hoa"));
students.add(new Student("B25DCCN102", "Trần Xuân Nhật"));
students.add(new Student("B25DCCN103", "Lê Hoài Nam"));
```

Cú pháp tổng quát

- Cú pháp tổng quát của kiểu generic:

```
access non-acess class name<T1, T2, T..., Tn> {
    // do something here...
}

access non-acess interface name<T1, T2, T..., Tn> {
    // do something here...
}
```

- Trong đó:
  - Các thành phần cơ bản của class, interface vẫn giữ nguyên
  - Bổ sung các kiểu được tham số hóa trong cặp <>
  - Cặp <> luôn nằm sau tên class/interface
  - Trong <> là các kí tự đại diện cho các kiểu

Cú pháp tổng quát

Tên các kiểu trong <> đặt theo quy ước:

- Dùng 1 chữ cái hoa làm tham số đại diện cho kiểu
- Các kí tự phổ biến và ý nghĩa:
  - E: element
  - K: key
  - V: value
  - N: number
  - T: type
  - S, U, … các kiểu thứ hai, thứ ba..
- Các kiểu sử dụng trong <> bắt buộc phải khác các kiểu nguyên thủy, có thể là bất kì class, interface nào

Cú pháp tổng quát

- Khi khai báo kiểu generic ta cung cấp đầy đủ tên kiểu cần truyền vào ở vế trái phép gán, vế phải chỉ cần <>.
- Ví dụ lớp generic:

```
public class MyMap<K, V> {
    private K key;
    private V value;

    public MyMap() {
    }

    public MyMap(K key, V value) {
        this.key = key;
        this.value = value;
    }
```

Cú pháp tổng quát

➢ Cú pháp sử dụng:

```
// tạo đối tượng thao tác với giá trị kiểu Integer, String
MyMap<Integer, String> map = new MyMap<>(1, "One");
System.out.println("Key: " + map.getKey()
        + " - Value: " + map.getValue());

// tạo đối tượng thao tác với giá trị kiểu String, String
MyMap<String, String> hello = new MyMap<>("Hello", "Xin chào");
System.out.println("Key: " + hello.getKey()
        + " - Value: " + hello.getValue());
```

➢ Kết quả:

```
Key: 1 - Value: One
Key: Hello - Value: Xin chào
```

Giới hạn trên

- Dấu ? Trong generic được gọi là kí tự đại diện
- Thường được sử dụng trong nhiều trường hợp khác nhau: làm kiểu của tham số, trường, biến cục bộ, đôi khi là kiểu trả về
- Để nới lỏng các hạn chế với 1 biến, tức là giới hạn các kiểu chưa biết lại trong phạm vi các kiểu con hoặc chính bản thân kiểu nào đó, ta sử dụng giới hạn trên
- Cú pháp: ? extends SuperType

Giới hạn trên

- Ví dụ sau sẽ nhận vào ArrayList<T> bất kì với T là con của Number hoặc chính Number:

```
public static double averageValue(ArrayList<? extends Number> list) {
    if (list == null || list.size() == 0) {
        return 0;
    }
    double sum = 0;
    for (var item : list) {
        sum += item.doubleValue();
    }
    return sum / list.size();
}
```

Giới hạn dưới

- Giới hạn dưới giúp thu hẹp các kiểu cần sử dụng trong phạm vi các kiểu cha hoặc chính bản thân 1 kiểu nào đó
- Để tạo giới hạn dưới ta dùng kí tự ? kết hợp với từ khóa super.
- Cú pháp: ? super SubType
- Giả định ta có các kiểu:
  - Grandchildren là con của Children
  - Children là con của Father
  - Father là con của Grandfather

Giới hạn dưới

- Ví dụ sau sẽ giới hạn kiểu được chấp nhận trong ArrayList<T> lại ở GrandChildren hoặc kiểu cha của nó:

```
public static void addNewObject(
                                ArrayList<? super Grandchildren> list) {
    list.add(new Grandchildren("Anh"));
    list.add(new Grandchildren("Trang"));
    list.add(new Grandchildren("Thanh"));
}

public static void main(String[] args) {
    ArrayList<Father> fathers = new ArrayList<>();
    ArrayList<Child> children = new ArrayList<>();
    ArrayList<Grandchildren> grandchildren = new ArrayList<>();
    addNewObject(fathers); // ok
    addNewObject(children); // ok
    addNewObject(grandchildren); // ok
}
```

Minh họa

- Thực hiện trong công cụ lập trình

Nội dung tiếp theo

Giới thiệu về Collection