Bài 63: Phương thức generic

- ✓ Khái niệm và đặc điểm
- ✓ Cú pháp tổng quát
- ✓ Giới hạn các kiểu trong generic
- ✓ Minh họa, bài tập thực hành

Khái niệm và đặc điểm

- Phương thức generic là phương thức có kiểu được tham số hóa
- Ý tưởng: cho phép sử dụng các kiểu(class, interface) làm tham số của phương thức
- Mục đích: tái sử dụng cùng một chức năng cho nhiều kiểu dữ liệu khác nhau
- Đối số của phương thức thông thường là các giá trị còn đối số của phương thức generic là các kiểu dữ liệu

Lợi ích của generic

- Kiểm soát được kiểu đang sử dụng tại thời điểm biên dịch(compile) chương trình: đảm bảo tính đúng đắn. Lỗi được phát hiện và sửa tại thời điểm biên dịch dễ hơn nhiều so với khi chương trình đang chạy
- Giảm thao tác ép kiểu các thành phần trong generic: chỉ những kiểu hợp lệ được chỉ rõ mới được phép đưa vào phương thức generic
- Cho phép thực hiện các thuật toán chung áp dụng cho tập hợp của nhiều kiểu dữ liệu khác nhau. Code dễ đọc hiểu và an toàn về kiểu

Ví dụ

- Phương thức sau hiển thị các phần tử của kiểu bất kì trong tham số kiểu ArrayList<T> mà nó nhận được:

```
/**
 * phương thức dùng để hiển thị danh sách các phần tử
 * của một kiểu bất kì.
 *
 * @param list danh sách cần hiển thị
 * @param <T>  kiểu cần thực hiện thao tác
 */
public static <T> void showList(ArrayList<T> list) {
    for (var e : list) {
        System.out.print(e + " ");
    }
    System.out.println();
}
```

Cú pháp

- Cú pháp tổng quát tạo phương thức generic:

```
access none-access <Types> methodType methodName(params) {
    // method body
    // do something here
}
```

- Trong đó:
  - Các thành phần đặc trưng của phương thức được giữ nguyên và bổ sung them thành phần generic
  - Phần Types trong cặp <> là danh sách các kiểu có thể được sử dụng làm kiểu trả về của phương thức, kiểu của các tham số
  - Phần Types có thể khai báo nhiều kiểu, mỗi kiểu biểu thị bằng 1 chữ cái hoa phân tách nhau bởi dấu phẩy

Cú pháp

- Cú pháp tổng quát tạo phương thức generic:

```
access none-access <Types> methodType methodName(params) {
    // method body
    // do something here
}
```

- Trong đó:
  - Ví dụ: <T>, <T, V>, <T, K, V>
  - Cặp <> là bắt buộc và là đặc trưng của generic. Nó luôn đứng sau access modifier và trước kiểu của phương thức
  - Danh sách tham số của phương thức có thể dùng các kiểu đã khai báo trong phần Types

Ví dụ

- Một ví dụ lấy giá trị phần tử chính giữa của danh sách bất kì:

```
/**
 * Lấy phần tử chính giữa của một List
 *
 * @param list danh sách các phần tử
 * @param <T>  kiểu của danh sách đang thao tác
 * @return giá trị chính giữa tìm được
 */
public <T> T getMid(ArrayList<T> list) {
    int mid = list.size() / 2;
    return list.get(mid);
}
```

Giới hạn các kiểu

- Giới hạn các kiểu được tham số hóa trong <> của phương thức generic là cách giúp phương thức generic triển khai được các thuật toán phổ biến dùng chung như đếm, sắp xếp, tìm kiếm cho nhiều kiểu dữ liệu khác nhau
- Cú pháp: *SubType extends SuperType*
- Trong đó:
  - SubType là một kí tự đại diện cho kiểu nào đó cần dùng
  - Từ khóa extends ngụ ý sử dụng chung cho mối quan hệ kế thừa với các lớp và implements với interface
  - SuperType có thể là các lớp hoặc inteface

Ví dụ

- Phương thức sau chỉ nhận vào danh sách các phần tử của kiểu có implements interface Comparable:

```
public <T extends Comparable> T findMax(ArrayList<T> list) {
    if (list == null || list.size() == 0) {
        return null;
    }
    var max = list.get(0);
    for (var e : list) {
        if (e.compareTo(max) > 0) {
            max = e;
        }
    }
    return max;
}
```

Minh họa

- Thực hiện trong công cụ lập trình

Nội dung tiếp theo

Lớp generic