GIAO DIỆN
INTERFACE

ĐẠI HỌC SÀI GÒN

09/2023

Giao diện - Interface

Khái niệm

- Là một khai báo của một tập hợp các phương thức mà không có phần thân.
- Không thể tạo một đối tượng từ interface.

Mục đích sử dụng

- Tính đa hình: Cho phép một đối tượng có nhiều hình thức.
- Lập trình dựa trên hợp đồng: Định nghĩa phương thức mà lớp thực thi phải triển khai.

Đặc điểm

- Không thể khởi tạo.
- Có thể kế thừa nhiều interfaces.
- Phạm vi truy cập mặc định là public.

Khai báo

- public **interface** InterfaceName {

}

```
public interface HinhHoc {
    double tinhDienTich();
    double tinhChuVi();
}
```

Triển khai

- public class tênClass implements tênInterface {

  thực thi tất cả các phương thức khai báo trong Interface;

}

```
public class HinhVuong implements HinhHoc {

    private double canh;

    public HinhVuong(double canh) {...3 lines }

    @Override
    public double tinhDienTich() {...3 lines }

    @Override
    public double tinhChuVi() {...3 lines }
}
```

Một số Interface của Java

vì sao đã có abtract lại cần có khái niệm Interface

- Đa kế thừa: Trong Java, một lớp không thể kế thừa từ nhiều lớp. Tuy nhiên, một lớp có thể triển khai nhiều interface. Điều này cho phép một lớp có hành vi (hoặc tính năng) từ nhiều nguồn mà không cần lo lắng về các vấn đề liên quan đến đa kế thừa.
- Tính linh hoạt: Interface cung cấp một hình thức hợp đồng mà lớp triển khai phải tuân theo, nhưng không ép buộc cấu trúc hoặc phương pháp triển khai cụ thể. Điều này giúp mã nguồn trở nên linh hoạt và dễ dàng thay đổi hơn.

vì sao đã có abtract lại cần có khái niệm Interface

- Phân tách trách nhiệm: Interface cho phép phân tách rõ ràng giữa hành vi và triển khai.
- Tính năng mở rộng trong Java 8+: Từ Java 8 trở đi, interface có thể chứa phương thức với triển khai mặc định và phương thức tĩnh. Điều này giúp thêm các phương thức mới vào interface mà không làm vỡ các lớp hiện có triển khai interface đó.

vì sao đã có abtract lại cần có khái niệm Interface

- Tính trừu tượng: Trong khi cả lớp trừu tượng và interface đều cho phép trừu tượng hóa, interface thường được sử dụng để định nghĩa một "hợp đồng" hoặc một bộ hành vi mà lớp triển khai cần tuân theo. Lớp trừu tượng có thể chứa trạng thái (các trường dữ liệu), trong khi interface không thể.

Interface trong Java

- Trong Java, một interface là một kiểu tham chiếu, giống như một lớp, nhưng chỉ chứa phương thức trừu tượng, biến hằng số, phương thức mặc định và phương thức tĩnh. Một interface không thể được khởi tạo và không thể chứa bất kỳ phương pháp triển khai nào (ngoại trừ các phương pháp mặc định và tĩnh).

Interface trong Java

- java.util.List: đại diện cho một danh sách các phần tử (các thực thể như ArrayList, LinkedList thực hiện nó).
- java.util.Iterator: Cho phép duyệt qua các phần tử của một bộ sưu tập.
- java.io.Serializable: Lớp thực hiện Serializable có thể được tuần tự hóa, tức là trạng thái của đối tượng có thể được chuyển đổi thành một dạng có thể lưu trữ hoặc truyền.
- java.util.Comparator: Được sử dụng để so sánh hai đối tượng.

Iterator

- Sửa đổi Bộ sưu tập trong Quá trình Lặp: Đây là lợi ích chính của Iterator. Khi duyệt qua một bộ sưu tập, muốn xóa một phần tử mà không gặp lỗi ConcurrentModificationException

```
List<String> list = new ArrayList<>(Arrays.asList("A", "B", "C"));
Iterator<String> iterator = list.iterator();
while (iterator.hasNext()) {
    String item = iterator.next();
    if ("B".equals(item)) {
        iterator.remove();
    }
}
```

Iterator

- Đa dạng trong việc Lặp: Một số cấu trúc dữ liệu không hỗ trợ vòng lặp for-each mà chỉ hỗ trợ Iterator.
- Truy cập và Kiểm soát Chi tiết hơn: Với Iterator, có khả năng kiểm tra xem có phần tử tiếp theo hay không thông qua phương thức hasNext(). Điều này có thể hữu ích trong một số tình huống cần kiểm tra trước khi thực hiện một hành động.

Ví dụ:

- Xóa các phân số trong danh sách các phân số

```
public void xoaPhanso (int tuso, int mauso){

    for (PhanSo1 ps : dsPhanso) {
        if(ps.getTuSo()*mauso==ps.getMauSo()*tuso){
            dsPhanso.remove(ps);
        }
    }
}
```

```
run:
Exception in thread "main" java.util.ConcurrentModificationException
```

Lỗi : ConcurrentModificationException

- ConcurrentModificationException là một ngoại lệ được ném ra bởi các phương thức trong Java Collections Framework khi nó phát hiện ra rằng một bộ sưu tập đang được sửa đổi khi đồng thời nó đang được duyệt qua

Ví dụ: Sử dụng Iterator

- Xóa các phân số trong danh sách các phân số

```
public void xoaPhanso (int tuso, int mauso){

    for (Iterator<PhanSo1> it = dsPhanso.iterator(); it.hasNext();) {
        PhanSo1 ps = it.next();
        if(ps.getTuSo()*mauso==ps.getMauSo()*tuso){
            it.remove();
        }
    }
}
```

Comparable

- là một interface trong Java và nó được sử dụng để xác định thứ tự tự nhiên của các đối tượng của một lớp. Một lớp triển khai Comparable đồng ý với việc tất cả các thực thể của nó đều có thể được so sánh lẫn nhau. Điều này làm cho các đối tượng của lớp có thể sắp xếp tự nhiên, ví dụ, trong một danh sách hoặc mảng.

Comparable

```
public interface Comparable<T> {
    public int compareTo(T o);
}
```

- Để triển khai Comparable, một lớp phải ghi đè phương thức compareTo(T o). Phương thức này trả về giá trị:
- <0 nếu đối tượng hiện tại nhỏ hơn đối tượng được so sánh (o).
- =0 nếu đối tượng hiện tại bằng với đối tượng được so sánh.
- >0 nếu đối tượng hiện tại lớn hơn đối tượng được so sánh.

Ví dụ:

- Khai báo lớp Phân số thực thi Comparable

```
public class PhanSo implements Comparable<PhanSo> {
    private int tuSo;
    private int mauSo;
```

Ví dụ:

- Ghi đè phương thức compareTo

```
@Override
public int compareTo(PhanSo o) {
    double giaTri1 = this.tuSo * 1.0 / this.mauSo;
    double giaTri2 = o.tuSo * 1.0 / o.mauSo;
    return Double.compare(giaTri1, giaTri2);
}
```

```
public void sapXep (){
    Collections.sort(dsPhanso);
}
```

Comparator

- là một interface trong Java và được sử dụng để so sánh hai đối tượng. Trong một số trường hợp, khi muốn sắp xếp hoặc so sánh các đối tượng dựa trên một tiêu chí cụ thể mà không tuân theo thứ tự tự nhiên của chúng.

java.util.Comparator

- Interface này chủ yếu chứa hai phương thức:
- compare(T o1, T o2): So sánh hai đối tượng o1 và o2. Trả về một số âm, số dương hoặc 0 tương ứng với việc o1 nhỏ hơn, lớn hơn hoặc bằng o2.
- equals(Object obj): Kiểm tra xem Comparator này có bằng obj không. Thường thì bạn không cần triển khai phương thức này, vì lớp Object đã có một phiên bản mặc định.

Sử dụng Comparator

- Ví dụ sắp xếp danh sách tên

```
List<String> names = Arrays.asList("Minh", "Nguyen", "Van");
Collections.sort(names, new Comparator<String>() {
    public int compare(String s1, String s2) {
        return s1.compareTo(s2);
    }
});
```

Phương thức tiện ích trong Java 8+

- comparing(): Tạo một Comparator từ một hàm trích xuất khóa.
- thenComparing(): Tạo một Comparator phụ sau một Comparator khác.
- reversed(): Đảo ngược thứ tự của một Comparator.

Ví dụ

- Lớp PhanSo1 thực thi interface Comparator

```
public class PhanSo1 implements Comparator<PhanSo1>{
    private int tuSo;
    private int mauSo;

    public PhanSo1() {
    }
```

Ví dụ

- Ghi đè phương thức compare

```
@Override
public int compare(PhanSo1 t, PhanSo1 t1) {
    double giaTri1 = t.tuSo * 1.0 / t.mauSo;
    double giaTri2 = t1.tuSo * 1.0 / t1.mauSo;
    return Double.compare(giaTri2, giaTri1);
}
```

Ví dụ

- Trong class QlPhanso1, thiết kế phương thức sắp xếp

```
public class QlPhanso1 {
    ArrayList<PhanSo1> dsPhanso ;

    public QlPhanso1() {
        this.dsPhanso = new ArrayList<>();
    }
```

```
public void sapxep (){
    Collections.sort(dsPhanso, new PhanSo1());
}
```

Ví dụ: sử dụng một lớp ẩn danh (anonymous class) để tạo Comparator

```
public void sapxep2(){
    Comparator<PhanSo1> cpm = new Comparator<PhanSo1>() {
        @Override
        public int compare(PhanSo1 t, PhanSo1 t1) {
            double giaTri1 = t.getTuSo() * 1.0 / t.getMauSo();
            double giaTri2 = t1.getTuSo() * 1.0 / t1.getMauSo();
            return Double.compare(giaTri2, giaTri1);
        }
    };
}
```

TRAO ĐỔI


SGU
TRƯỜNG ĐẠI HỌC SÀI GÒN
SAIGON UNIVERSITY
ĐẠI HỌC SÀI GÒN
CẤM ĐẬU XE