有且仅有一个抽象方法的接口.

@FunctionalInterface

修饰符 interface 接口名{

public [abstract] 返回值类型 方法名称(参数列表);

//其他非抽象方法内容(默认、静态、私有)

}

@Override 注解检测方法是否为重写

@FunctionalInterface 注解检测接口是否为函数式接口

函数式接口的使用:通常作为方法的参数和返回类型

```
//调用show方法,方法的参数是一个接口,所以我们可以传递接口的匿名内部类
show(new MyFunctionalInterface() {
    @Override
    public void method() {
        System.out.println("使用匿名内部类重写接口中的抽象方法");
    }
});

//调用show方法,方法的参数是一个函数式接口,所以我们可以Lambda表达式
show(()->{
    System.out.println("使用Lambda表达式重写接口中的抽象方法");
});

//简化Lambda表达式
show(()-> System.out.println("使用Lambda表达式重写接口中的抽象方法"));
```

常用函数式接口:java.util.function

① Supplier接口:java.util.function.Supplier<T>,T get()获取泛型参数指定类型的对象数据

(生产型接口)

```
/*
Supplier<T>接口被称之为生产型接口,指定接口的泛型是什么类型,那么接口中的get方法就会生产什么类型的数据
*/
public class Demo01Supplier {
    //定义一个方法,方法的参数传递Supplier<T>接口,泛型执行String,get方法就会返回一个String
    public static String getString(Supplier<String> sup){
        return sup.get();
    }

    public static void main(String[] args) {
        //调用getString方法,方法的参数Supplier是一个函数式接口,所以可以传递Lambda表达式
        String s = getString(()->{
            //生产一个字符串,并返回
            return "胡歌";
        });
        System.out.println(s);

        //优化Lambda表达式
        String s2 = getString(()->"胡歌");
        System.out.println(s2);
    }
```

② Consumer接口:java.util.function.Consumer<T>