# JDK8

作者：wanmisy
原文链接：https://ld246.com/article/1532247617125
来源网站：链滴

# 1 函数式接口

## 1.1 定义

- 函数式接口是只包含一个抽象方法声明的接口，如java.lang.Runnable

## 1.2 声明方式

- @FunctionalInterface

## 1.3 案例

```
@FunctionalInterface
public interface WorkerInterface {
    public void dosomething();
}


public class WorkInterfaceTest {
    public static void excute(WorkerInterface workerInterface) {
        workerInterface.dosomething();
    }

    public static void main(String[] args) {
        // 运用同名类实现
        excute(new WorkerInterface() {

            @Override
```

```
            public void dosomething() {
                System.out.println("hello world");
            }
        });

        // 用lambda表达式实现
        excute(() -> System.out.println("hello world"));
    }
}
```

运行结果:

```
hello world
hello world
```

# 2 Lambda表达式和函数式接口

## 2.1 定义

- 为匿名函数，是没有声明的方法，即无访问修饰符、返回值和名称

## 2.2 语法

- (arg1, arg2...) -> { body }
- (type1 arg1, type2 arg2...) -> { body }

## 2.3 结构

- 一个Lambda表达式可以有0个或多个参数;
- 参数类型可明确声明，或根据上下文推断。如(int a)与(a)效果相同;
- 所有参数需在()内,参数之间用逗号隔开。如(a,b)或(int a, int b);
- 空圆括号代表参数为空；当只有一个参数，且其类型可推导时，圆括号（）可省略。例如：a -> ret rn a*a;
- Lambda 表达式的主体可包含零条或多条语句
- 如果Lambda表达式的主体只有一条语句，花括号{}可省略。匿名函数的返回类型与该主体表达式致
- 如果Lambda表达式的主体包含一条以上语句，则表达式必须包含在花括号{}中（形成代码块）。名函数的返回类型与代码块的返回类型一致，若没有返回则为空

## 2.4 案例

### 2.4.1 实现Runnable接口

```
public class RunnableLambda {
    public static void main(String[] args) {
```

```
        // 1.1使用匿名内部类
        new Thread(new Runnable() {

            @Override
            public void run() {
                System.out.println("hello world1");
            }
        }).start();

        // 1.2使用匿名内部类
        Runnable r1 = new Runnable() {

            @Override
            public void run() {
                System.out.println("hello world2");
            }
        };

        r1.run();

        // 2.1 lambda表达式
        new Thread(() -> System.out.println("hello world3")).start();;

        // 2.2 lambda表达式
        Runnable r2 = ()-> System.out.println("hello world4");
        r2.run();
    }
}
```

结果

```
hello world1
hello world2
hello world3
hello world4
```

### 2.4.2 排序数组

- 匿名内部类的排序

```
public class ArraySortLambda {
public static void main(String[] args) {
String[] players = { "Jack", "Tom", "John", "Andy", "Role"};

    // 1.1 使用匿名内部类根据 name 排序 players
    Arrays.sort(players, new Comparator() {
        @Override
        public int compare(String s1, String s2) {
            return (s1.compareTo(s2));
        }
    });
```

```
    // java8新特性
    Arrays.asList(players).stream().forEach(System.out::println);


}
}
```

结果

```
Andy
Jack
John
Role
Tom
```

● lambda表达式的排序

```
import java.util.Arrays;
import java.util.Comparator;
public class ArraySortLambda {
public static void main(String[] args) {
String[] players = { "Jack", "Tom", "John", "Andy", "Role"};
// 排序1
Comparator sortByName = (String s1, String s2) -> (s1.compareTo(s2));
Arrays.sort(players, sortByName);
// 或者排序二
Arrays.sort(players, (String s1, String s2) -> (s1.compareTo(s2)));

    Arrays.asList(players).stream().forEach(System.out::println);

}
}
```