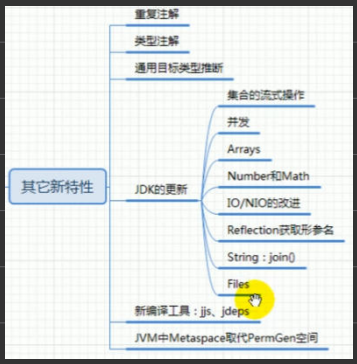
日期: /

大处着眼，小处着手
逆向思维，反证法
透过问题看本质
小不忍则乱大谋
识时务者为俊杰

Java 8 其它新特性

这不就是…？



年轻不经事

1、Lambda表达式

(1)举例：Comparator<Integer> com =

(o1,o2) -> Integer.compare(o1,o2)

(2)格式：（ o ） ——→

Lambda形参列表　Lambda操作符　Lambda体（方法体）

(3) 6种情况

① 无参无返　( ) -> { };

② 1参无返　(String s) -> { };

③ 类型推断　(s) -> { };　左右注型

④ 1参省括号

⑤ 多参　(s1, s2)

⑥ 只有一条语句，{ }和return可省

日期: /

2、函数式接口 Lambda表达式的本质

(1) 概念：一个接口中只有一个抽象方法

注解：@FunctionalInterface

(2) 内置的4个核心函数式接口

消费型接口	Consumer<T>	void accept(T t)
供给型接口	Supplier<T>	T get()
函数型接口	Function<T,R>	R apply(T t)
断定型接口	Predicate<T>	boolean test(T t)

3、引用

(1) 方法引用

1. 使用情境：当要传递给Lambda体的操作，已经有实现的方法了，可以使用方法引用！

2. 方法引用，本质上就是Lambda表达式，而Lambda表达式作为函数式接口的实例。所以方法引用，也是函数式接口的实例。

3. 使用格式： 类(或对象) :: 方法名

4. 具体分为如下的三种情况：

   情况1 对象 :: 非静态方法

   情况2 类 :: 静态方法

   情况3 类 :: 非静态方法

5. 方法引用使用的要求：要求接口中的抽象方法的形参列表和返回值类型与方法引用的方法的形参列表和返回值类型相同！（针对于情况1和情况2）

```
// 情况三：类 :: 实例方法 (有难度)
// Comparator中的int comapre(T t1,T t2)
// String中的int t1.compareTo(t2)
```

第一个形参作为调用者

日期: /

(2) 构造器引用和方法引用

一、构造器引用

和方法引用类似，函数式接口的抽象方法的形参列表和构造器的形参列表一致。抽象方法的返回值类型即为构造器所属的类的类型

二、数组引用

大家可以把数组看做是一个特殊的类，则写法与构造器引用一致。

```
Function<Integer,Employee> func2 = Employee :: new;
Employee employee1 = func2.apply( t: 1002);
System.out.println(employee1);
```

```
Function<Integer,String[]> func2 = String[] :: new;
String[] arr2 = func2.apply( t: 10);
System.out.println(Arrays.toString(arr2));
```

$$\int_1^{y-x} \frac{1-\cos\left(\frac{\pi}{2}t\right)}{2}\,dt$$

$$= \frac{x}{2} - \frac{\sin\left(\frac{\pi}{2}t\right)}{\pi}\Bigg]_1^{y-x}$$

$$= \frac{y-x}{2} - \frac{\sin(\pi)}{\pi}$$

日期: /

日期: /