ThoughtWorks®

OO BootCamp

Java 8 新特性

Function 接口、Lambda 表达式、Stream API

ThoughtWorks

Lambda 表达式

```
  public interface Action {
      void doSomething();
  }
```

```
 void use(Action action)
```

```
first.use(
        new Action() {
            public void doSomething() {
                System.out.println(" do some thing");
            }
        }
);
```

```
first.use(
        () -> System.out.println(" do some thing")
);
```

```
public interface Calculator {
    double calculate(double a,double b);
}
```

```
double ca(Calculator caculator)
```

```
first.ca(new Calculator() {
    public double calculate(double a, double b) {
        return a+b;
    }
});
```

```
first.ca((x,y)->{return x+y;});


first.ca((x,y)->x+y);
```

注意：1. 参数类型和个数 2. 返回值 3.大括号

ThoughtWorks

函数接口

练习

- 创建一个线程，随便做点什么
  Thread(Runnable action)

练习

- 使用 List::sort(Comparator c) 按巧克力个数排序，反序

接口的 default 方法

java.util.function 包

Consumer<T>
- void accept(T t)

BiConsumer<T,U>
- void accept(T t, U u)

Supplier<T>
- T get()

Function<T,R>
- R apply(T t)

BiFunction<T,U,R>
- R apply(T t, U u)

Predicate<T>
- boolean test(T t)

Iterable<T>

```
void  forEach (Consumer<? super T> action)
```

Map<K, V>

```
void forEach (BiConsumer<? super K, ? super V> action)
```

Map<K, V>

```
V computeIfAbsent (K key,
     Function<? super K, ? extends V> mappingFunction)
```

Collection<E>

```
boolean removeIf (Predicate<? super E> filter)
```

- 注意：我们也可以直接使用这个接口，不限于核心库内部

ThoughtWorks®

函数引用

函数引用

```
public void doSome(Consumer<String> consumer){
    String msg = "PI 的值等于"+Math.PI;
    consumer.accept(msg);
}

doSome(x - > System.out.println(x));
doSome(System.out::println);
```

```
public void output(Function<String,String> function){
    String apply = function.apply("this is some message");
    System.out.println(apply);
}

output(String::toUpperCase);
```

注意：第二个示例是一个实例方法

构造函数引用

```
public class Employee {
    private int id;
    private String name;

    public Employee(int id) {
        this.id = id;
    }
  … …
}

public void checkEmployeeMap(Map<Integer, Employee> map){
    map.computeIfAbsent(10086, Employee::new);
}
```

ClassName::methodName
- 如果不是静态方法，则第一个参数做目标对象

instanceName::methodName

Stream API

简单示例

```
public void countHuawei(List<String> paper){
    paper.stream().filter(w->w.equals("huawei")).count();
}
```

Steam API 的结构

`paper.stream()`	`.filter(w->w.equals("huawei"))`	`.count();`
1. 获得Stream	2. Map - 转化成其他Stream	3. Reduce - 流的终结

获得Stream

- Arrays.stream(xx[] array)
- Collection.stream( ) / Collection.parallelStream()
- Stream.of(T …) / stream.parallel()

转化

- Stream<T> filter (Predicate<? super T> predicate)
- <R> Stream<R> map (Function<? super T, ? extends R> mapper)
- Stream<T>. distinct( )
- Stream<T> limit (long maxSize)
- Stream<T> skip (long n)

终结

- void forEach (Consumer<? super T> action)
- Optional<T> min (Comparator<? super T> comparator)
- Optional<T> max (Comparator<? super T> comparator)