匿名类_Lambda

@M了个J
李明杰

https://github.com/CoderMJLee
http://cnblogs.com/mjios

码拉松

小码哥教育 SEEMYGO

匿名类（Anonymous Class）

- 当接口、抽象类的实现类，在整个项目中只用过一次，可以考虑使用匿名类

```
public interface Eatable {
    String name();
    int energy();
}
```

```
public class Person {
    public void eat(Eatable e) {
        System.out.println(
                "eat - " + e.name()
                + " - " + e.energy());
    }
}
```

```
Person person = new Person();
person.eat(new Eatable() {
    @Override
    public String name() {
        return "Apple";
    }
    @Override
    public int energy() {
        return 100;
    }
}); // eat - Apple - 100
```

```
Eatable beef = new Eatable() {
    @Override
    public String name() {
        return "Beef";
    }
    @Override
    public int energy() {
        return 500;
    }
};
person.eat(beef); // eat - Beef - 500
person.eat(beef); // eat - Beef - 500
```

匿名类的使用注意

- 匿名类不能定义除编译时常量以外的任何 `static` 成员
- 匿名类只能访问 `final` 或者 有效 `final` 的局部变量
- 匿名类可以直接访问外部类中的所有成员（即使被声明为 `private`）
  - 匿名类只有在实例相关的代码块中使用，才能直接访问外部类中的实例成员（实例变量、实例方法）
- 匿名类不能自定义构造方法，但可以有初始化块
- 匿名类的常见用途
  - 代码传递
  - 过滤器
  - 回调

排序

■ 可以使用 JDK 自带的 java.util.Arrays 类对数组进行排序

```
Integer[] array = { 33, 22, 11, 77, 66, 99 };
Arrays.sort(array);
// [11, 22, 33, 66, 77, 99]
System.out.println(Arrays.toString(array));

Arrays.sort(array, new Comparator<Integer>() {
    @Override
    public int compare(Integer o1, Integer o2) {
        return o2 - o1;
    }
});
// [99, 77, 66, 33, 22, 11]
System.out.println(Arrays.toString(array));
```

■ Arrays.sort 默认是升序排列

- 把比较小的元素放左边
- 把比较大的元素放右边

■ compare 的返回值

- 等于 0
  - ✓ o1 == o2
- 大于 0
  - ✓ o1 > o2
- 小于 0
  - ✓ o1 < o2

小码哥教育 SEEMYGO

Comparable vs Comparator

- 如果数组元素本身具备可比较性（实现了 java.util.Comparable 接口）
  - 可以直接使用 Arrays.sort 方法进行排序

- java.util.Comparator 的存在意义？
  - 可以在不修改类源代码的前提下，修改默认的比较方式（比如官方类、第三方类）
  - 可以让一个类提供多种比较方式

Lambda Expression

- Lambda 表达式是 Java 8 开始才有的语法，发音：美 [ˈlæmdə]

- 函数式接口（Functional Interface）：只包含 1 个抽象方法的接口
  - 可以在接口上面加上 `@FunctionalInterface` 注解，表示它是一个函数式接口

- 当匿名类实现的是函数式接口时，可以使用 Lambda 表达式进行简化

```
@FunctionalInterface
public interface Testable {
    void test(int v);
}
```

Lambda 实例

```
@FunctionalInterface
public interface Calculator {
    int calculate(int v1, int v2);
}

static void execute(int v1, int v2, Calculator c) {
    System.out.println(c.calculate(v1, v2));
}

public static void main(String[] args) {
    execute(10, 20, (int v1, int v2) -> {
        return v1 + v2;
    }); // 30

    execute(11, 22, (v1, v2) -> v1 + v2); // 33
}
```

- 参数列表可以省略参数类型

- 当只有一条语句时
  - 可以省略大括号、分号、return

- 当只有一个参数时
  - 可以省略小括号

- 当没有参数时
  - 不能省略小括号

小码哥教育 SEEMYGO

Lambda 的使用注意

- Lambda 只能访问 final 或者 有效 final 的局部变量
- Lambda 没有引入新的作用域

```
public class OuterClass {
    private int age = 1;

    public class InnerClass {
        private int age = 2;
        void inner() {
            // int v = 4; // error
            Testable t = v -> {
                System.out.println(v); // 3
                System.out.println(age); // 2
                System.out.println(this.age); // 2
                System.out.println(InnerClass.this.age); // 2
                System.out.println(OuterClass.this.age); // 1
            };
            t.test(3);
        }
    }
}
```

```
@FunctionalInterface
public interface Testable {
    void test(int v);
}
```

匿名类 vs Lambda

```java
@FunctionalInterface
public interface Testable {
    void test(int v);
}
```

```java
public class OuterClass {
    private int age = 1;

    public class InnerClass {
        private int age = 2;
        void inner() {
            int v = 4;
            Testable t = new Testable() {
                @Override
                public void test(int v) {
                    System.out.println(v); // 3
                    System.out.println(age); // 2
                    // System.out.println(this.age); // error
                    System.out.println(InnerClass.this.age); // 2
                    System.out.println(OuterClass.this.age); // 1
                }
            };
            t.test(3);
        }
    }
}
```

小码哥教育 SEEMYGO

常用函数式接口

- java.util.function 包中提供了很多常用的函数式接口
  - Supplier（美 [səˈplaɪər]）
  - Consumer（美 [kənˈsuːmər]）
  - Predicate（美 [ˈpredɪkeɪt]）
  - Function（美 [ˈfʌŋkʃn]）
  - .....

小码哥教育 SEEMYGO

Supplier

```java
@FunctionalInterface
public interface Supplier<T> {
    T get();
}
```

小码哥教育 SEEMYGO

Supplier 应用

- 有时使用 Supplier 传参，可以避免代码的浪费执行（有必要的时候再执行）

```
String s1 = "A";
String s2 = "B";
String s3 = "C";
// A
getFirstNotEmptyString(s1, () -> s1 + s2 + s3)

String getFirstNotEmptyString(String s1, Supplier<String> s2) {
    if (s1 != null && s1.length() != 0) return s1;
    if (s2 == null) return null;
    String str = s2.get();
    return (str != null && str.length() != 0) ? str : null;
}
```

小码哥教育 SEEMYGO

Consumer

```
@FunctionalInterface
public interface Consumer<T> {
    void accept(T t);

    default Consumer<T> andThen(Consumer<? super T> after);
}
```

小码哥教育 SEEMYGO

Consumer 应用

```
void forEach(int[] nums, Consumer<Integer> c) {
    if (nums == null || c == null) return;
    for (int n : nums) {
        c.accept(n);
    }
}
```

```
int[] nums = { 11, 33, 44, 88, 77, 66 };
forEach(nums, (n) -> {
    String result = ((n & 1) == 0) ? "偶数" : "奇数";
    System.out.println(n + "是" + result);
});
```

小码哥教育 SEEMYGO

Consumer – andThen

```
void forEach(int[] nums, Consumer<Integer> c1, Consumer<Integer> c2) {
    if (nums == null || c1 == null || c2 == null) return;
    for (int n : nums) {
        c1.andThen(c2).accept(n);
    }
}
```

```
int[] nums = { 11, 33, 44, 88, 77, 66 };
forEach(nums, (n) -> {
    String result = ((n & 1) == 0) ? "偶数" : "奇数";
    System.out.println(n + "是" + result);
}, (n) -> {
    String result = ((n % 3) == 0) ? "能" : "不能";
    System.out.println(n + result + "被3整除");
});
```

小码哥教育 SEEMYGO

Predicate

```
@FunctionalInterface
public interface Predicate<T> {
    boolean test(T t);

    default Predicate<T> and(Predicate<? super T> other);
    default Predicate<T> negate();
    default Predicate<T> or(Predicate<? super T> other);

    static <T> Predicate<T> isEqual(Object targetRef);
}
```

小码哥教育 SEEMYGO

Predicate 应用

```
String join(int[] nums, Predicate<Integer> p) {
    if (nums == null || p == null) return null;
    StringBuilder sb = new StringBuilder();
    for (int n : nums) {
        if (p.test(n)) {
            sb.append(n).append("_");
        }
    }
    sb.deleteCharAt(sb.length() - 1);
    return sb.toString();
}
```

```
int[] nums = { 11, 33, 44, 88, 77, 66 };
String str = join(nums, (n) -> (n & 1) == 0);
// 44_88_66
System.out.println(str);
```

小码哥教育 SEEMYGO

Predicate – and

```
String join(int[] nums, Predicate<Integer> p1, Predicate<Integer> p2) {
    if (nums == null || p1 == null || p2 == null) return null;
    StringBuilder sb = new StringBuilder();
    for (int n : nums) {
        if (p1.and(p2).test(n)) {
            sb.append(n).append("_");
        }
    }
    sb.deleteCharAt(sb.length() - 1);
    return sb.toString();
}
```

```
int[] nums = { 11, 33, 44, 88, 77, 66 };
String str = join(nums, (n) -> (n & 1) == 0, (n) -> (n % 3) == 0);
// 66
System.out.println(str);
```

小码哥教育 SEEMYGO

Predicate – or

```
String join(int[] nums, Predicate<Integer> p1, Predicate<Integer> p2) {
    if (nums == null || p1 == null || p2 == null) return null;
    StringBuilder sb = new StringBuilder();
    for (int n : nums) {
        if (p1.or(p2).test(n)) {
            sb.append(n).append("_");
        }
    }
    sb.deleteCharAt(sb.length() - 1);
    return sb.toString();
}
```

```
int[] nums = { 11, 33, 44, 88, 77, 66 };
String str = join(nums, (n) -> (n & 1) == 0, (n) -> (n % 3) == 0);
// 33_44_88_66
System.out.println(str);
```

小码哥教育 SEEMYGO

Predicate – negate

```
String join(int[] nums, Predicate<Integer> p) {
    if (nums == null || p == null) return null;
    StringBuilder sb = new StringBuilder();
    for (int n : nums) {
        if (p.negate().test(n)) {
            sb.append(n).append("_");
        }
    }
    sb.deleteCharAt(sb.length() - 1);
    return sb.toString();
}
```

```
int[] nums = { 11, 33, 44, 88, 77, 66 };
String str = join(nums, (n) -> (n & 1) == 0);
// 11_33_77
System.out.println(str);
```

小码哥教育 SEEMYGO

Function

```
@FunctionalInterface
public interface Function<T, R> {
    R apply(T t);

    default <V> Function<V, R> compose(Function<? super V, ? extends T> before);
    default <V> Function<T, V> andThen(Function<? super R, ? extends V> after);

    static <T> Function<T, T> identity();
}
```

小码哥教育 SEEMYGO

Function 应用

```
int sum(String[] strs, Function<String, Integer> f) {
    if (strs == null || f == null) return 0;
    int result = 0;
    for (String str : strs) {
        result += f.apply(str);
    }
    return result;
}
```

```
String[] strs = { "12", "567", "666" };
int result = sum(strs, Integer::valueOf);
System.out.println(result);
```

小码哥教育 SEEMYGO

Function – andThen

```
// 将所有数字的个位数加起来
int sum(String[] strs,
        Function<String, Integer> f1,
        Function<Integer, Integer> f2) {
    if (strs == null || f1 == null || f2 == null) return 0;
    int result = 0;
    for (String str : strs) {
        result += f1.andThen(f2).apply(str);
    }
    return result;
}
```

```
String[] strs = { "12", "567", "666" };
int result = sum(strs, Integer::valueOf, (i) -> i % 10);
// 15
System.out.println(result);
```

小码哥教育 SEEMYGO

Function – compose

```
// 将所有数字的个位数加起来
int sum(String[] strs,
        Function<String, Integer> f1,
        Function<Integer, Integer> f2) {
    if (strs == null || f1 == null || f2 == null) return 0;
    int result = 0;
    for (String str : strs) {
        result += f2.compose(f1).apply(str);
    }
    return result;
}
```

```
String[] strs = { "12", "567", "666" };
int result = sum(strs, Integer::valueOf, (i) -> i % 10);
// 15
System.out.println(result);
```

方法引用（Method Reference）

- 如果 Lambda 中的内容仅仅是调用某个方法，可以使用方法引用（Method Reference）来简化

种类	用法
引用静态方法	ClassName::staticMethodName
引用特定对象的实例方法	ObjectName::instanceMethodName
引用特定类型的任意对象的实例方法	ClassName::methodName
引用构造方法	ClassName::new
引用当前类中定义的实例方法	this::instanceMethodName
引用父类中定义的实例方法	super::instanceMethodName

小码哥教育 SEEMYGO

引用类方法

```
@FunctionalInterface
public interface Testable {
    int test(int v1, int v2);
}
```

```
Testable t1 = (v1, v2) -> Math.max(v1, v2);
// 20
System.out.println(t1.test(10, 20));

Testable t2 = Math::max;
// 20
System.out.println(t2.test(10, 20));
```

小码哥教育 SEEMYGO

引用特定对象的实例方法

```
@FunctionalInterface
public interface Testable {
    void test(int v);
}

public class Person {
    public void setAge(int age) {
        System.out.println(
        "Person - setAge - " + age);
    }
}
```

```
static void execute(Testable t, int v) {
    t.test(v);
}

// 10
execute(v -> System.out.println(v), 10);
// Person - setAge - 10
execute(v -> new Person().setAge(v), 10);

// 20
execute(System.out::println, 20);
// Person - setAge - 20
execute(new Person()::setAge, 20);
```

小码哥教育 SEEMYGO

引用特定类型的任意对象的实例方法

```
String[] strings = { "Jack", "james", "Apple", "abort" };

Arrays.sort(strings, (s1, s2) -> s1.compareToIgnoreCase(s2));
// [abort, Apple, Jack, james]
System.out.println(Arrays.toString(strings));

Arrays.sort(strings, String::compareToIgnoreCase);
// [abort, Apple, Jack, james]
System.out.println(Arrays.toString(strings));
```

小码哥教育 SEEMYGO

引用构造方法

```
@FunctionalInterface
public interface Testable {
    Object test(int v);
}

public class Person {
    public Person(int age) {
        System.out.println("Person - " + age);
    }
}
```

```
Testable t1 = v -> new Person(v);
// Person - 18
// Person@816f27d
System.out.println(t1.test(18));

Testable t2 = Person::new;
// Person - 18
// Person@6ce253f1
System.out.println(t2.test(18));
```

小码哥教育 SEEMYGO

引用数组的构造方法

```
Testable t1 = v -> new int[v];
// 3
System.out.println(((int[]) t1.test(3)).length);

Testable t2 = int[]::new;
// 3
System.out.println(((int[]) t2.test(3)).length);
```

小码哥教育 SEEMYGO

引用当前类中定义的实例方法

```
@FunctionalInterface
public interface Testable {
    void test(int v);
}
```

```
public class Person {
    public void setAge(int age) {
        System.out.println("setAge - " + age);
    }

    public void show() {
        Testable t1 = v -> setAge(v);
        // setAge - 10
        t1.test(10);

        Testable t2 = this::setAge;
        // setAge - 10
        t2.test(10);
    }
}
```

小码哥教育 SEEMYGO

引用父类中定义的实例方法

```java
@FunctionalInterface
public interface Testable {
    void test(int v);
}
```

```java
public class Person {
    public void setAge(int age) {
        System.out.println(
        "Person - setAge - " + age);
    }
}
```

```java
public class Student extends Person {
    public void setAge(int age) {
        System.out.println("Student - setAge - " + age);
    }
    public void show() {
        Testable t1 = v -> super.setAge(v);
        // Person - setAge - 10
        t1.test(10);

        Testable t2 = super::setAge;
        // Person - setAge - 10
        t2.test(10);
    }
}
```

小码哥教育 SEEMYGO

综合实例

```
<X, Y> void process(
          Iterable<X> eles,
          Predicate<X> tester,
          Function<X, Y> mapper,
          Consumer<Y> block) {
    if (eles == null || tester == null) return;
    if (mapper == null || block == null) return;
    for (X ele : eles) {
        if (!tester.test(ele)) continue;
        Y data = mapper.apply(ele);
        block.accept(data);
    }
}
```

小码哥教育 SEEMYGO

综合实例

```
public class Person {
    private String name;
    private int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public int getAge() {
        return age;
    }
    @Override
    public String toString() {
        return "Person [name=" + name + ", age=" + age + "]";
    }
}
```

小码哥教育 SEEMYGO

综合实例

```
List<Person> ps = new ArrayList<>();
ps.add(new Person("Jack", 20));
ps.add(new Person("Rose", 10));
ps.add(new Person("Kate", 15));
ps.add(new Person("Larry", 40));
process(ps, (p) -> p.getAge() >= 15 && p.getAge() <= 25,
          Person::toString, System.out::println);
// Person [name=Jack, age=20]
// Person [name=Kate, age=15]
```

```
List<Integer> is = new ArrayList<>();
is.add(11);
is.add(22);
is.add(33);
is.add(44);
process(is, (i) -> (i & 1) == 0, (i) -> "test_" + i, System.out::println);
// test_22
// test_44
```

小码哥教育 SEEMYGO

Stream 初体验

```
ps.stream()
.filter((p) -> p.getAge() >= 15 && p.getAge() <= 25)
.map(Person::toString)
.forEach(System.out::println);
// Person [name=Jack, age=20]
// Person [name=Kate, age=15]
```

```
is.stream()
.filter((i) -> (i & 1) == 0)
.map((i) -> "test_" + i)
.forEach(System.out::println);
// test_22
// test_44
```