Java 核心技术(高阶)

第七章 Lambda表达式

第二节 函数式接口

华东师范大学 陈良育

Lambda应用过程(1)

- 给定一个字符串数组，请按长度从小到大递增排序

```
String[] planets = new String[] {
        "Mercury", "Venus", "Earth", "Mars",
        "Jupiter", "Saturn", "Uranus",
        "Neptune" };

System.out.println("使用Lambda, 长度从小到大:");
Arrays.sort(planets,
        (String first, String second)
           -> first.length() - second.length());
System.out.println(Arrays.toString(planets));
```

Lambda应用过程(2)

- Arrays.sort(T[] a, Comparator<? super T> c)

```
Arrays.sort(planets,
        (String first, String second)
           -> first.length() - second.length());
```

```
Comparator<String> c = (String first, String second) ->
    first.length() - second.length();

Arrays.sort(planets, c);
```

Lambda应用过程(3)

- Lambda表达式

```
Comparator<String> c = (String first, String second) ->
    first.length() - second.length();
```

- 匿名内部类

```
Comparator<String> c = new Comparator<String>() {
            public int compare(String first, String second)
            {
                return first.length() - second.length();
            }
        };
```

- Lambda表达式(匿名函数)自动成为接口方法的实现

函数式接口(1)

- 函数式接口(Functional Interface)
  - 是一个接口，符合Java接口的定义
  - 只包含一个抽象方法的接口
  - 可以包括其他的default方法、static方法、private方法
  - 由于只有一个未实现的方法，所以Lambda表达式可以自动填上这个尚未实现的方法
  - 采用Lambda表达式，可以自动创建出一个(伪)嵌套类的对象(没有实际的嵌套类class文件产生)，然后使用，比真正嵌套类更加轻量，更加简洁高效

函数式接口(2)

- Lambda表达式

```
Comparator<String> c = (String first, String second) ->
    first.length() - second.length();
```

- 匿名内部类

```
Comparator<String> c = new Comparator<String>() {
            public int compare(String first, String second)
            {
                return first.length() - second.length();
            }
        };
```

- Lambda表达式(匿名函数)自动成为接口方法的实现

函数式接口(3)

- Comparator接口：有2个未实现的方法

All Methods	Static Methods	Instance Methods	Abstract Methods	Default Methods

Modifier and Type	Method and Description
int	compare(T o1, T o2) Compares its two arguments for order.
boolean	equals(Object obj) Indicates whether some other object is "equal to" this comparator.

```
Comparator<String> c = (String first, String second) ->
    first.length() - second.length();
```

- 任何实现Comparator接口的类，肯定继承了Object，也就有equals实现。
- https://docs.oracle.com/javase/8/docs/api/java/lang/FunctionalInterface.html

自定义函数式接口(1)


```
public interface StringChecker {
    public boolean test(String s);
}
```

```
String[] planets = new String[] {
        "Mercury", "Venus", "Earth", "Mars",
        "Jupiter", "Saturn", "Uranus", "Neptune" };

StringChecker evenLength = s ->
    {
        if(s.length()%2 == 0)
            return true;
        return false;
    };

for(String p : planets) {
    if(evenLength.test(p)) {
        System.out.println(p);
    }
}
```

自定义函数式接口(2)

```
public interface StringChecker {
    public boolean test(String s);
}
```

```
@FunctionalInterface
//系统自带的函数式接口注解，用于编译器检查

public interface StringChecker {
    public boolean test(String s);
    //public boolean test2(String s);
}
```

系统自带的函数式接口(1)

- 函数式接口
  - 只带有一个未实现的方法，内容简单
  - 大量重复性的函数式接口，使得源码膨胀
- 系统自带的函数式接口
  - 涵盖大部分常用的功能，可以重复使用
  - 位于java.util.function包中

系统自带的函数式接口(2)

- 系统自带的函数式接口(部分常用)

接口	参数	返回值	示例
Predicate<T>	T	Boolean	接收一个参数，返回一个布尔值
Consumer<T>	T	void	接受一个参数，无返回
Function<T, R>	T	R	接受一个参数，返回一个值
Supplier<T>	None	T	数据工厂

系统自带的函数式接口(3)

- Predicate<T>: 接收一个参数，返回一个布尔值

All Methods	Static Methods	Instance Methods	Abstract Methods	Default Methods

Modifier and Type	Method and Description
boolean	test(T t) Evaluates this predicate on the given argument.

```
Predicate<String> oddLength = s ->
        s.length()%2 == 0 ? false:true;

for(String p : planets) {
        if(oddLength.test(p)) {
                System.out.println(p);
        }
}
```

系统自带的函数式接口(4)

- Consumer<T>：接收一个参数，做操作，无返回

All Methods	Instance Methods	Abstract Methods	Default Methods

Modifier and Type	Method and Description
void	accept(T t) Performs this operation on the given argument.

```
Consumer<String> printer = s ->
    System.out.println("Planet :" + s);

for(String p : planets) {
    printer.accept(p);
}
```

系统自带的函数式接口(5)

- Supplier<T>：无输入参数，返回一个数据

All Methods	Instance Methods	Abstract Methods
Modifier and Type	Method and Description	
T	get() Gets a result.	

```
Supplier<String> planetFactory = () ->
    planets[(int) floor(random() * 8)];

for (int i = 0; i < 5; i++) {
    System.out.println(planetFactory.get());
}
```

系统自带的函数式接口(6)

- Function<T>：接收一个参数，返回一个参数

All Methods	Static Methods	Instance Methods	Abstract Methods	Default Methods

Modifier and Type	Method and Description
R	apply(T t) Applies this function to the given argument.

```
Function<String, String> upper = s ->
        {
            //可以做更复杂的操作
            return s.toUpperCase();
        };

for(String p : planets) {
        System.out.println(upper.apply(p));
}
```

总结

- 了解Lamdba表达式和函数式接口的关系
- 掌握自定义函数式接口
- 学习JDK自带的标准函数式接口

谢 谢！