Java 核心技术(高阶)

第七章 Lambda表达式

第一节 Lambda表达式的定义

华东师范大学 陈良育

例子(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("使用长度从小到大的比较器:");
  Arrays.sort(planets, new LengthAscComparator());
  System.out.println(Arrays.toString(planets));
```

```
public class LengthAscComparator
  implements Comparator<String> {
    public int compare(String first, String second) {
        //获取2个字符串长度
        int len1 = (first==null? 0 : first.length());
        int len2 = (first==null? 0 : second.length());

        //如果len1 == len2，返回0，表示两个字符串相等
        //如果len1 >  len2，返回正数，表示first>second
        //如果len1 <  len2，返回负数，表示first<second
        return len1 - len2;
    }
}
```

例子(2)

- 给定一个字符串数组，请按长度从大到小递减排序

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

  System.out.println("使用长度从大到小的比较器:");
  Arrays.sort(planets, new LengthDescComparator());
  System.out.println(Arrays.toString(planets));
```

```
public class LengthDescComparator
  implements Comparator<String> {
    public int compare(String first, String second) {
        //获取2个字符串长度
        int len1 = (first==null? 0 : first.length());
        int len2 = (first==null? 0 : second.length());

        //如果len1 == len2，返回0，表示两个字符串相等
        //如果len1 >  len2，返回正数，表示first>second
        //如果len1 <  len2，返回负数，表示first<second
        return (-1) * (len1 - len2);
    }
}
```

例子(3)

- 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));
```

例子(4)

- 给定一个字符串数组，请按长度从大到小递减排序

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

  System.out.println("使用Lambda，长度从大到小:");
  Arrays.sort(planets, (first, second) -> (-1)*(first.length()-second.length()));
  System.out.println(Arrays.toString(planets));
```

Lambda表达式(1)

- 面向过程程序语言：参数传递是基本类型的变量
- 面向对象语言
  - 传递基本类型的变量
  - 传递对象变量
- 传递方法/代码块(函数式程序语言设计)
  - 刚开始，Java为了简单性、一致性，拒绝此功能
  - 为了市场和技术的需要，Java 8开始，支持此项功能，提出Java的Lambda表达式实现

Lambda表达式(2)

- Lambda表达式
  - 数学家/逻辑学家Alonzo Church提出的λ演算
  - 表示可有效计算的数学函数，用字母λ表示
  - 在计算机编程中，通常用来表示一个匿名函数
  - Lambda表达式可以当作参数，传递给其他高阶函数

Java Lambda表达式(1)

- Lambda表达式
  - 参数，箭头，一个表达式

```
(String first, String second)
    -> first.length() - second.length()
```

  - 参数，箭头，{多个语句}

```
(first, second) ->
{
    //形参不写类型，可以从上下文推断出来
    int result = (-1) * (first.length() - second.length());
    return result;
}
```

Java Lambda表达式(2)

- Lambda表达式
  - 类似于匿名方法，一个没有名字的方法
  - 参数，箭头，表达式语句
  - 可以忽略写参数类型
  - 坚决不声明返回值类型
  - 没有public/protected/private/static/final等修饰符
  - 单句表达式，将直接返回值，不用大括号
  - 带return语句， 算多句，必须用大括号

Java Lambda表达式(3)

- Lambda表达式
  - 无参数，仅保留括号，箭头，表达式

```
new Thread(
    ()->
        {
            int sum=0;
            for(int i=1;i<=100;i++)
            {
                sum = sum + i;
            }
            System.out.println("总和:" + sum);
        }
    ).start();
```

Java Lambda表达式(4)

- Lambda表达式
  - 一个参数，可省略括号，箭头，表达式

```
public interface Adder {

    public int selfAdd(int x);

}
```

```
Adder   c1 =
          x ->
          {
              x++;
          };
```

Java Lambda表达式(5)

- Lambda表达式
  - 如果有返回值，返回值类型会在上下文推断出来的，无需声明
  - 只在某几个分支有返回值，这样是不合法的

```
Adder   c2 =
          x ->
          {
               if(x>0)
                    return x+1;
          };
```

总结

- 了解Lambda的作用
- 了解Lambda的定义语法

谢 谢！