**Lambda表达式-**

## Lambda表达式介绍

Lambda表达式是jdk8引进的新特性。Lambda表达式实际上是带参数的表达式，是一个可传递的代码块，可以把它作为函数的参数，或者赋值给变量。

在继续往下介绍Lambda表达式之前，要先介绍另一个概念，函数式接口。简单来说，函数式接口就是有且只有一个抽象方法（不包括默认方法静态方法等）的接口，常见的函数式接口有Comparable接口和Runnable接口等。

Lambda表达式的使用和函数式接口有很大关系。

## Lambda表达式使用

### 使用前提

Lambda表达式可以作为其它函数的参数，也可以赋值给变量，但是这两种做法都有一个共同的前提，那就是其它函数的参数类型或者变量的类型为**函数式接口**，表达式可以起到替代实现函数式接口的对象的作用。

### 使用语法

Lambda表达式的语法如下（符号全是英文符号）：

(arg1, arg2, arg3…) -> { do something }

其中参数的类型不需要写，参数类型会自动推断，参数的数量由函数式接口中的抽象方法参数个数确定

就拿Runnable接口来进行Lambda表达式作参数的示范，如果我们想创建一个线程，有四种方式可以做到，实现Runnable接口就是其中的一种方式。在实现Runnable接口时我们有两种方式选择，一种是编写一个类实现该接口，一种是使用匿名内部类，下面是这两种方式的代码。

1. 编写类实现Runnable接口

**public class RunnableTest {**

**public static void main(String[] args) {**

**// 将实现了Runnable接口的对象作参数，创建线程**

**Thread mt = new Thread(new MyThread());**

**mt.start();**

**}**

**}**

**class MyThread implements Runnable {**

**@Override**

**public void run() {**

**System.out.println("我是通过实现Runnable接口创建的新线程！");**

**}**

**}**

1. 使用匿名内部类

**public class RunnableTest {**

**public static void main(String[] args) {**

**// 使用匿名内部类对象作参数，创建线程**

**Thread mt = new Thread(new Runnable() {**

**@Override**

**public void run() {**

**System.out.println("我是通过匿名内部类创建的新线程！");**

**}**

**});**

**mt.start();**

**}**

**}**

1. 使用Lambda表达式

**public class RunnableTest {**

**public static void main(String[] args) {**

**// 使用Lambda表达式作参数，创建线程**

**Thread mt = new Thread( () -> {**

**System.out.println("我是通过Lambda表达式创建出来的线程");**

**} );**

**mt.start();**

**}**

**}**

可见使用Lambda表达式是可以作为函数的参数的，但是函数的参数需要是一个函数式接口，另一种是将表达式赋值给变量，但变量类型也需要是函数式接口，如Comparator接口，下面是实现整数逆序的比较器。

**import java.util.Comparator;**

**import java.util.TreeSet;**

**public class ComparatorTest {**

**public static void main(String[] args) {**

**// 使用Lambda表达式创建使整数逆序的比较器**

**Comparator<Integer> ctor = (i1, i2) -> {return i2 - i1;};**

**TreeSet<Integer> tree = new TreeSet<>(ctor);**

**tree.add(1);**

**tree.add(5);**

**tree.add(3);**

**tree.add(4);**

**tree.add(2);**

**// 使用Lambda表达式输出set的每个值**

**tree.forEach(x -> System.out.println(x));**

**}**

**}**

在需要函数式接口时，使用Lambda表达式可以简化代码的书写。你可能注意到，上面forEach()里面的Lambda表达式没有()也没有{}，这是为什么呢？Lambda表达式根据参数列表()和花括号体{}里面参数和语句数量的不同，可以省略一些代码，具体如下：

1. 参数列表：参数只有一个时，可以省略()，其余情况不能省略
2. 花括号体：语句只有一条时，{}可以省略。当函数式接口的抽象方法需要返回值时，如果省略了{}，语句前的return可以不用写，如果不省略{}，语句前必须要写return。语句有两条及以上，必须要写{}，要返回值的语句前要写return

除此之外，当Lambda表达式的花括号体只有一条语句时，还可以使用::符号来进行类方法或指定对象的实例方法的引用，大致是 类名::类方法 这样子，具体的用法我就不在这写了

## 小结

在变量需要或者函数的参数是实现了函数式接口的对象时，可以使用Lambda表达式来替代，从而简化代码的书写。

表达式的参数为一个时，()可以省略。

表达式的花括号体只有一条语句时，{}可以省略，省略时如果要return不用写，不省略时要return则要写

表达式的花括号体只有一条语句时，可以进行 类名::类方法 或 对象::实例方法 的引用，引进来的方法作为函数式接口的抽象方法的实现