1. lambda表达式
2. 为什么需要lambda表达式

lambda表达式是一个可传递的代码块，可以在以后执行一次或者多次。这种传递代码的方式比较简洁，可读性强。

1. lambda表达式的语法规范

Java中Lambda表达式的形式是：参数，箭头（->）以及一个表达式。Java是一种强类型语言，变量的类型是一定要指定的，因此参数的类型需要指定。但是，如果参数的类型可以通过上下文推导出，也可以省略类型。如：

(String first, String second) -> {

if(first.length() < second.length()) return -1;

else if(first.length() > second.length()) return 1;

else return 0;

}

即是lambda表达式中没有参数，也要提供空括号。但是，若只有一个参数，且参数类型可以推导出，那么可以省略参数的括号。

1. lambda表达式的变量作用域

我们可以在lambda表达式中使用外围方法或类中的变量，这种变量称为自由变量，并非lambda表达式的参数也不是定义的变量。它能够被lambda表达式捕获到并且使用，但是注意，这种变量一定是值不会改变的变量，因为若是可以改变的变量，那么并发执行多个动作时就会造成危险。这种自由变量称为闭包。

1. lambda表达式的this关键字

在lambda表达式中，使用this关键字时代表的并不是当前的lambda表达式，而是指创建这个lambda表达式的方法的this参数。如：

public class Application{

public void init(){

ActionListener listener = event->{

System.out.println(this.toString());

}

}

}

这里的this指的是Application对象，调用的是它的toString方法。

1. lambda表达式的使用场景

lambda表达式在什么情况下使用呢？

1. 在算法的适当位置运行代码（如，排序中的比较器）
2. 发生某种情况时运行代码（如，点击按钮）
3. Comparator的优化

我们可以继续优化Comparator，除了使用lambda表达式重写外，还可以使用Comparator接口中的静态方法来创建比较器，如静态的comparing方法取一个“键提取器”函数，它将类型T映射为一个可比较的类型。对要比较的对象应用这个函数，然后对返回的键进行比较。如，假设有一个Person对象数组，可以按照名字对这些对象进行排序：

Arrays.sort(people, Comparator.comparing(Person::getName));

如果有多次比较，就将比较器和thenComparing方法连接起来，处理比较结果相同的情况：

Arrays.sort(people, Comparator.comparing(Person::getLastName).thenComparing(Person::getFirstName));

1. 函数式接口

只有一个抽象方法的接口，称为函数式接口。当需要这种接口的对象时，我们可以提供一个lambda表达式。当我们在需要传递一个函数式接口Comparator的地方传递一个lambda表达式，如下所示：

Arrays.sort(words, (first,second)->first.length()-second.length());

在底层实现中，Arrays.sort方法会接收实现了Comparator<String>的某个类的对象，在这个对象上调用compare方法会执行这个lambda表达式的体。

1. 方法引用

当lambda表达式只涉及一个方法时，我们可以通过一个方法引用来传递方法到该lambda表达式的位置，替代表达式。如：

本应该调用new Timer(1000, event->System.out.println(event));，这里使用方法引用，new Timer(1000, System.out::println);

方法引用指示编译器生成一个函数式接口的实例，覆盖这个接口的抽象方法来调用给定的方法。方法引用有三种情况，第一种是等价与向方法传递参数的lambda表达式，如x->System.out.println(x)；第二种是第一个参数会成为方法的隐式参数，如(x,y)->x.compareToIgnoreCase(y)；第三种是所有参数都传递到静态方法中，如(x,y)->Math.pow(x,y);

注意，只有当lambda表达式的体只调用一个方法时，没有其他操作，才能够将lambda表达式重写为方法引用。如：

s->s.length() == 0;

是不行的，因为不仅有方法调用，还有一个比较操作，所以不能使用方法引用。

包含对象的方法引用与等价的lambda表达式有一个细微的差别，构造方法引用时如果有异常会在构造时抛出，而lambda表达式要在调用时才会抛出异常。

1. 构造器引用

构造器引用与方法引用很类似，只不过方法名为new，如Person::new是Person构造器的一个引用，若有多个构造器，引用哪个是根据上下文来决定的。