Lambda表达式：

1、lambda的演变

Lambda表达式是由匿名内部类演变而来的，其目的就是为了简化代码。

有以下员工接口:

public interface Employee {

public void getSalary();

}

如果想使用接口中的方法原先的做法是

方式一：写一个类实现该接口，创建类的实例调用方法。

方式二：使用匿名内部类完成上述功能的实现:

public class EmployeeTest {

public static void main(String[] args) {

Employee employee = new Employee(){

@Override

public void getSalary(){

System.out.println("员工Tim的工资为12元");

}

};

employee.getSalary();

}

}

将上述匿名内部类转为使用lambda表达式:

employee = ()-> System.***out***.println("Salary是5000元");  
employee.getSalary();

总结：

lambda表达式可以看作是一个接口的实例，使用lambda是有前提条件的，就是该接口必须是函数式接口。那么什么是函数式接口呢？用一句话总结就是有且仅有一个抽象方法的接口(在java提供的类库中，函数式接口上面都有@FunctionalInterface注解)，注意是抽象方法，这个接口中可以包含多个默认方法，类方法。如Employee接口中只有一个getSalary()抽象方法。

接口本质上是抽象类abstract，默认可以省略该关键字，但是其内的方法是不允许有方法体的，在jdk8以后，接口可以有默认的方法，默认的方法必须写方法体，相当于是接口的方法，如下代码：

public interface Employee {  
 public void getSalary(int a);  
  
 default void grade(String str){  
 System.***out***.println(str);  
 }  
}

//测试类中调用

employee = ()-> System.***out***.println("Salary是5000元");  
employee.getSalary();

//调用会报错，因为，在Employee接口中，找不到没有参数的抽象方法，所以因此可以直接通过这个表达式，找到对应接口中唯一的那个抽象方法，从而调用方法，扩展业务逻辑，相当于是对该抽象方法，进行实现，所以->右边的实现，是可以随便写的。

lambda表达式可以总结为：一个(), 一个 ->, 一段代码{}

一个()的()中传递的是需要的参数，一段代码{}中的代码是具体的业务代码。在这里只是简单讲一下，lambda表达式是由匿名内部类演化而来的，目的是简化代码。其他的像什么情况下()可以省略，{}可以省略，以及讲到的函数式接口大家可以自行查一下，重点在后面的如何使用。

“：：”的使用：

Java 8 中我们可以通过 `::` 关键字来访问类的构造方法，对象方法，静态方法。

IConvert<String, String> convert = Something::startsWith;  
String converted = convert.convert("123");

获取IConvert接口中的方法，将Something::startsWith(

将something中的startswith方法作为该接口中唯一的一个抽象方法的实现)，当调用convert.convert("123");实际运行的是something中的startswith()方法。相当于还是一个扩展。