Lambda表达式:允许应用表达式来代替接口,提供了一个正常的参数列表,和一个使用这些参数的主体(表达式或者代码块);

还增强了集合库,添加了2个对集合数据进行批量操作的包,java.util.function,java.util.stream.

stream如同迭代器,附加额外功能.

基本语法:

表达式:(参数列表 parameters) -> 表达式 expression

代码块: (参数列表 parameters) -> 代码块 statements;

简单例子:

1.() -> 5 不需要参数,返回值为5

2.x -> 2 \* x 返回参数的2倍

3. (x, y) -> x – y 接收两个参数,返回差值

4. int x, int y) -> x + y 接收两个int 返回和

5.(String s) -> sys.p(s); 接收String ,打印s

==============================================================================s实际例子:

1.适用于只有一个抽象方法的接口,创建的时候只会识别这一个接口.

没有参数

(1).直接创建类实现接口,

(2).直接创建接口,写一个内部类实现

(3.) APP app=()->{

具体执行的方法

};

简洁的()->方法;

lambda表达式,不会单独生成class文件

有参数:

APP app=()->{

具体执行的方法

};

简洁的()->方法;

参数的参数类型是可省略的

代码块中,只有一行代码,大括号可以省略

带返回值的方法

只有一个return的 一种只写返回值,另一种需要大括号

参数加修饰符:需要加参数的类型

===============================================================================

1.8中添加了许多对集合操作的方法.

并行执行的

list.parallelStream().返回一个stream,将stream转为int

(a->a) 将a赋值给a,返回给stream.再将这些求和.