基本定义

Lambda表达式的基本结构是：

f -> { }

f 参数变量

-> 语法符号

{ } 语句块

（作为参数的变量名，是可以自定义的，不是固定的）

Lambda表达式在功能上等同于一个匿名方法：

**public** **void** **unknown**(f) {

System.out.println(f.getName());

}

（相当于方法名和public void 修饰符都省略了。）

类型识别

F 变量的类型是系统根据上下文 **自动** 识别的。

回顾上节课的代码：

List<Fruit> fruits = Arrays.asList(......);

fruits.forEach(f -> {

System.out.println(f.getName());

});

ForEach( ) 方法表示循环遍历fruits 集合，每次循环时 f 变量指代当前遍历到的元素。

由于 fruits 变量的类型使用泛型语句定义为 List<Fruit>，表示集合中元素的类型是 Fruit。

所以，f 变量的类型自然是 Fruit，就可以调用 Fruit 类的方法了。

多参数

箭头（->）前表示参数变量，有 **多个参数** 的时候，必须使用小括号（） 包裹：

(student1, student2) -> {}

无参数

**没有参数** 的时候，必须使用小括号（）：

() -> {}

单条执行语句

箭头后的执行语句 **只有一条** 时，可以不加大括号{} 包裹

s -> System.out.println(s);

（但是不建议这么写）

有类型参数

fruits.forEach((Fruit f) -> {

System.out.println(f.getName());

});

如果代码比较复杂，而为了易阅读、易维护，也可以为参数变量指定类型。

（即使只有一个参数，也必须使用小括号（）包裹，否则会出错）

引用外部变量

Lambda表达式{ } 内的执行语句，除了能引用参数变量以外，还可以引用外部的变量。

List<Fruit> fruits = Arrays.asList(......);

String message = "水果名称：";

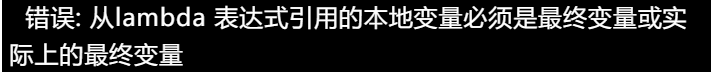
fruits.forEach(f -> {

System.out.println(message + f.getName());

});

message = "";

可以看到报错了：



规范一：

因为使用Lambda表达式有规范：引用的 **局部变量** 不允许被修改。

（即使是写在表达式后面也不行）

下面代码修改了局部变量也是错误的：

String message;

fruits.forEach(f -> {

message = "水果名称：";

});

（我们在《Java面向对象》课程中写过final关键字，Lambda表达式引用的 局部变量 即使不声明为 final，也要具备 final 的特性：变量值初始化后不允许被修改。）

规范二

使用Lambda表达式还有一点要注意：参数不能与 **局部变量** 同名。下面写法是错误的：

String f = "水果名称：";

fruits.forEach(f -> {});

双冒号（::）操作符

前面例子中，只有一条执行语句的 Lambda 表达式：

List<String> names = Arrays.asList("zhangSan", "LiSi", "WangWu");

names.forEach(n -> {

System.out.println(n);

});

可以进一步简化为：

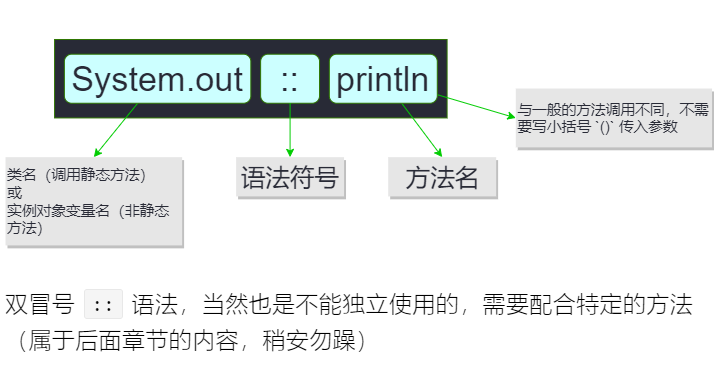
names.forEach(System.out::println);

这段代码的重点是使用 ：：时，系统每次遍历取得的元素（n），会**自动**作为参数传递给System.out.println() 方法打印输出：

System.out::println 等同于 n ->{System.out.println(n);}

:: 语法干脆省略了参数变量，所以读这段代码需要相互对照、并且自行脑补

语法含义



不同用法

方法一：静态方法调用

使用LambdaTest : : print 代替 f->LambdaTest.print(f).

方法二：调用非静态方法

Print( ) 方法不再表示为 static，于是需要实例对象来调用。

fruits.forEach(**new** LambdaTest()::print);

只是简写了：

LambdaTest lt = **new** LambdaTest();

fruits.forEach(lt::print);

用法三：多参数

本章第二节课的案例：

Collections.sort(students, (student1, student2) -> {

*// 第一个参数的学号 vs 第二个参数的学号*

**return** student1.getRollNo() - student2.getRollNo();

});

就碰到了多参数的情况。如果把比较的过程定义成一个方法：

**private** **static** **int** **compute**(Student s1, Student s2) {

... ...

... ...

}

那么排序过程就可以简写为：

Collections.sort(students, SortTest::compute);

用法四：父类方法

Super关键字。