# Method Reference

使用方式：

Object Type：：instanceMethod

Numbers::isMoreThanFifty

Anonymous Class (没有明显定义类型名称，所以称为匿名)

|  |
| --- |
| List<Integer> newList = *findNumbers*(list,  **new** BiPredicate<Integer, Integer>() {  **public** **boolean** test(Integer i1, Integer i2) {  **return** Numbers.*isMoreThanFifty*(i1, i2);  }  }); |

Lambda Expresion

|  |
| --- |
| newList = *findNumbers*(list, (i1, i2) -> Numbers.*isMoreThanFifty*(i1, i2)); |

Method Reference

|  |
| --- |
| newList = *findNumbers*(list, Numbers::*isMoreThanFifty*); |

三者比较起来，以Method Reference最为简介明了。就是直接调用所需功能块。

## 直接调用功能块（Method References）

说白了，method reference就是直接调用功能块（method）

这个软件包中，MethodReferenceTest.java是一个很好的Method reference的实例。作为Java Application运行（Run AS🡺Java Application），结果如下

|  |
| --- |
| John::65  Ailian::59  Charles::36  Printing rosterSet from lambda stream:  Ailian::59  Charles::36  John::65  Printing rosterSet:  Ailian::59  Charles::36  John::65 |

例如, 如果你定义了一个类型（class）Person，其中有一个通用（static）的功能块

|  |
| --- |
| **public** **class** Person {  **public** **enum** Sex {  ***MALE***, ***FEMALE***  }  String name;  Date birthday;  Sex gender;  String emailAddress;  **int** age;  **public** Person(String name, Date birthday) {  **this**.name = name;  **this**.birthday = birthday;  getAge();  }  … …  **public** **static** **int** compareByAge(Person a, Person b) {  **return** a.birthday.compareTo(b.birthday);  }  … …  } |

是用来提供对Person这类的个体作比较的功能的。过去通常的作法是，定义一个比较类型

|  |
| --- |
| **class** PersonAgeComparator  **implements** Comparator<Person> {  **public** **int** compare(Person a, Person b) {  **return** a.getBirthday().compareTo(b.getBirthday());  }  } |

然后在列表排序使用new语句

|  |
| --- |
| // Without method reference  Arrays.*sort*(rosterAsArray, **new** PersonAgeComparator()); |

使用了Method Reference之后，程序变得干净了很多，也更容易理解了。

|  |
| --- |
| Arrays.*sort*(rosterAsArray, Person::*compareByAge*); |

这里直接调用Person类型下的通用功能块compareByAge（）就可以了。实际上就是我们要做的事。