1. 字符串

String提供了一个与split相反的**join**方法，将字符串通过分隔符组合起来.字符串可以来自于Array或者Iterable<? extends CharSequence>:

```
1. String joined = String.join("/", "usr", "local", "bin"); // "usr/local/bin"
2. System.out.println(joined);
3. String ids = String.join(", ", ZoneId.getAvailableZoneIds());
4. System.out.println(ids);
```

2. 数字类

Short, Integer, Long, Float, 和 Double提供了**sum**、**max**和**min**，用来在流操作中作为聚合函数使用。Boolean包含了**logicalAnd, logicalOr, 和 logicalXor**

7种数字类型的包装类提供了**BYTES**字段，用来表示该类型的长度
System.out.println(Integer.BYTES);

8种原始类型的包装类型，提供了静态的**hashCode**方法，用来返回与实例方法相同的hash码，省去装箱/拆箱。

3. 集合

Class/Interface	New Methods
Iterable	forEach
Collection	removeIf
List	replaceAll, sort
Map	forEach, replace, replaceAll, remove(key, value) (removes only if key mapped to value), putIfAbsent, compute, computeIf(Absent|Present), merge
Iterator	forEachRemaining
BitSet	stream

4 比较器

comparing：

```
1. Arrays.sort(people,
Comparator.comparing(Person::getLastName).thenComparing(Person::g
etFirstName));
```

reverseOrder： 可以得到相反的排序

5 使用文件

1.读取文件行的流

为了延迟读取一个文件中的行，可以使用**Files.line**s方法，它会产生一个包含字符串的流，每个字符串就是文件的一行

```
1. Stream<String> lines = Files.lines(path);
2. Optional<String> passwordEntry = lines.filter(s ->
s.contains("password")).findFirst();
```

一旦包含password的第一行被找到，剩下的行就不会再被读取