1、Futures和Callables类

java.util.concurrent.Callable对象可以返回由线程完成的计算结果，而runnable接口只能运行线程。 Callable对象返回Future对象，该对象提供监视线程执行的任务进度的方法。 Future对象可用于检查Callable的状态，然后线程完成后从Callable中检索结果。 它还提供超时功能。

语法

```
//submit the callable using ThreadExecutor
//and get the result as a Future object
Future result10 = executor.submit(new FactorialService(10));

//get the result using get method of the Future object
//get method waits till the thread execution and then return the result of the execution.
Long factorial10 = result10.get();
```

2、fork-join框架

fork-join框架允许在几个工作进程中断某个任务，然后等待结果组合它们。 它在很大程度上利用了多处理器机器的生产能力。 以下是fork-join框架中使用的核心概念和对象。

Fork

Fork是一个进程，其中任务将其分成可以并发执行的较小且独立的子任务。

语法

```
Sum left  = new Sum(array, low, mid);
left.fork();
```

这里Sum是RecursiveTask的子类，left.fork()方法将任务分解为子任务。

Join

连接(Join)是子任务完成执行后任务加入子任务的所有结果的过程，否则它会持续等待。

语法

```
left.join();
```

这里剩下的是Sum类的一个对象。

ForkJoinPool

它是一个特殊的线程池，旨在使用fork-and-join任务拆分。

语法

```
ForkJoinPool forkJoinPool = new ForkJoinPool(4);
```

这里有一个新的ForkJoinPool，并行级别为4个CPU。

RecursiveAction

RecursiveAction表示不返回任何值的任务。

语法

```
class Writer extends RecursiveAction {
   @Override
```

```
   protected void compute() { }
}
```

递归任务

RecursiveTask表示返回值的任务。

语法

```
class Sum extends RecursiveTask {
   @Override
   protected Long compute() { return null; }
}
```

3、BlockingQueue接口

java.util.concurrent.BlockingQueue接口是Queue接口的子接口，另外还支持诸如在检索元素之前等待队列变为非空的操作，并在存储元素之前等待队列中的空间变得可用 。

BlockingQueue接口中的方法

序号	方法	描述
1	boolean add(E e)	将指定的元素插入到此队列中，如果可以立即执行此操作，而不会违反容量限制，在成功时返回true，并且如果当前没有空间可用，则抛出IllegalStateException。
2	boolean contains(Object o)	如果此队列包含指定的元素，则返回true。
3	int drainTo(Collection <? super E> c)	从该队列中删除所有可用的元素，并将它们添加到给定的集合中。
4	int drainTo(Collection <? super E> c, int maxElements)	最多从该队列中删除给定数量的可用元素，并将它们添加到给定的集合中。
5	boolean offer(E e)	将指定的元素插入到此队列中，如果可以立即执行此操作而不违反容量限制，则成功返回true，如果当前没有空间可用，则返回false。
6	boolean offer(E e, long timeout, TimeUnit unit)	将指定的元素插入到此队列中，等待指定的等待时间(如有必要)才能使空间变得可用。
7	E poll(long timeout, TimeUnit unit)	检索并删除此队列的头，等待指定的等待时间(如有必要)使元素变为可用。
8	void put(E e)	将指定的元素插入到此队列中，等待空间/容量可用。
9	int remainingCapacity ()	返回此队列可理想地(在没有内存或资源约束的情况下)接受而不阻止的附加元素数，如果没有内在限制则返回Integer.MAX_VALUE。
10	boolean remove(Object o)	从该队列中删除指定元素的单个实例(如果存在)。
11	E take()	检索并删除此队列的头，如有必要，等待元素可用。

4、ConcurrentMap接口

java.util.concurrent.ConcurrentMap接口是Map接口的子接口，支持底层Map变量上的原子操作。 它具有获取和设置方法，如在变量上的读取和写入。 也就是说，一个集合与同一变量上的任何后续读取相关联。 该接口确保线程安全性和原子性保证。

ConcurrentMap接口中的方法

序号	方法	描述
1	default V compute(K key, BiFunction<? super K,? super V,? extends V> remappingFunction)	尝试计算指定键及其当前映射值的映射(如果没有当前映射，则为null)。
2	default V computeIfAbsent(K key, Function<? super K,? extends V> mappingFunction)	如果指定的键尚未与值相关联(或映射到null)，则尝试使用给定的映射函数计算其值，并将其输入到此映射中，除非为null。
3	default V computeIfPresent(K key, BiFunction<? super K,? super V,? extends V> remappingFunction)	如果指定键的值存在且非空，则尝试计算给定键及其当前映射值的新映射。
4	default void forEach(BiConsumer<? super K,? super V> action)	对此映射中的每个条目执行给定的操作，直到所有条目都被处理或操作引发异常。
5	default V getOrDefault(Object key, V defaultValue)	返回指定键映射到的值，如果此映射不包含该键的映射，则返回defaultValue。
6	default V merge(K key, V value, BiFunction<? super V,? super V,? extends V> remappingFunction)	如果指定的键尚未与值相关联或与null相关联，则将其与给定的非空值相关联。
7	V putIfAbsent(K key, V value)	如果指定的键尚未与值相关联，请将其与给定值相关联。
8	boolean remove(Object key, Object value)	仅当当前映射到给定值时才删除键的条目。
9	V replace(K key, V value)	仅当当前映射到某个值时才替换该项的条目。
10	boolean replace(K key, V oldValue, V newValue)	仅当当前映射到给定值时才替换键的条目。
11	default void replaceAll(BiFunction<? super K,? super V,? extends V> function)	将每个条目的值替换为对该条目调用给定函数的结果，直到所有条目都被处理或该函数抛出异常。

5、ConcurrentNavigableMap接口

java.util.concurrent.ConcurrentNavigableMap接口是ConcurrentMap接口的子接口，并且支持NavigableMap操作，并且对其可导航子映射和近似匹配进行递归。

ConcurrentNavigableMap接口中的方法

序号	方法	描述
1	NavigableSet<K> descendingKeySet()	返回此映射中包含的键的相反顺序的NavigableSet视图。
2	ConcurrentNavigableMap<K, V> descendingMap()	返回此映射中包含的映射的反向排序视图。
3	ConcurrentNavigableMap<K,	

3	ConcurrentNavigableMap<K, V> headMap(K toKey)	返回该映射的部分键严格小于toKey的视图。
4	ConcurrentNavigableMap<K, V> headMap(K toKey, boolean inclusive)	返回该映射的部分视图，其键值小于(或等于，如果包含值为true)toKey。
5	NavigableSet<K> keySet()	返回此映射中包含的键的NavigableSet视图。
6	NavigableSet<K> navigableKeySet()	返回此映射中包含的键的NavigableSet视图。
7	ConcurrentNavigableMap<K, V> subMap(K fromKey, boolean fromInclusive, K toKey, boolean toInclusive)	返回此映射部分的视图，其键范围从fromKey到toKey。
8	ConcurrentNavigableMap<K, V> subMap(K fromKey, K toKey)	返回此映射，其键从fromKey(包括)到toKey所述部分的视图。
9	ConcurrentNavigableMap<K, V> tailMap(K fromKey)	返回此映射的部分视图，其键值大于或等于fromKey。
10	ConcurrentNavigableMap<K, V> tailMap(K fromKey, boolean inclusive)	从映射返回一个键大于(或等于，包含值为true)部分的视图。