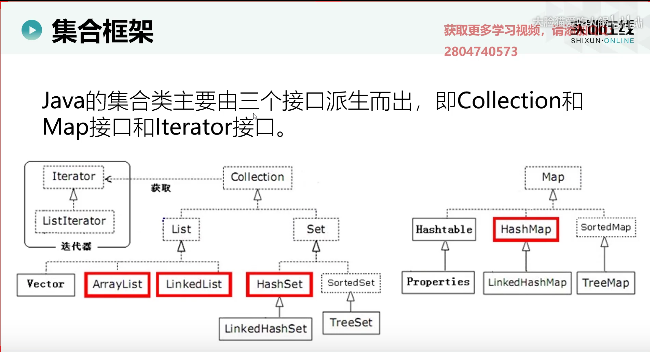
Java集合底层原理



关键字default能够使在接口中写方法体，

**public** **interface** Iterable<T> {

Iterator<T> iterator();

**default** **void** forEach(Consumer<? **super** T> action) {

Objects.*requireNonNull*(action);

**for** (T t : **this**) {

action.accept(t);

}

}

**default** Spliterator<T> spliterator() {

**return** Spliterators.*spliteratorUnknownSize*(iterator(), 0);

}

}

**public** **interface** Collection<E> **extends** Iterable<E> {

**int** size();

**boolean** isEmpty();

**boolean** contains(Object o);

Iterator<E> iterator();

Object[] toArray();

<T> T[] toArray(T[] a);

**boolean** add(E e);

**boolean** remove(Object o);

**boolean** containsAll(Collection<?> c);

**boolean** addAll(Collection<? **extends** E> c);

**boolean** removeAll(Collection<?> c);

**default** **boolean** removeIf(Predicate<? **super** E> filter) {

Objects.*requireNonNull*(filter);

**boolean** removed = **false**;

**final** Iterator<E> each = iterator();

**while** (each.hasNext()) {

**if** (filter.test(each.next())) {

each.remove();

removed = **true**;

}

}

**return** removed;

}

**boolean** retainAll(Collection<?> c);

**void** clear();

**boolean** equals(Object o);

**int** hashCode();

@Override

**default** Spliterator<E> spliterator() {

**return** Spliterators.*spliterator*(**this**, 0);

}

**default** Stream<E> stream() {

**return** StreamSupport.*stream*(spliterator(), **false**);

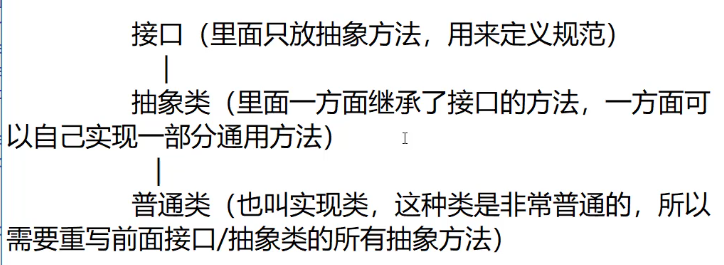
}

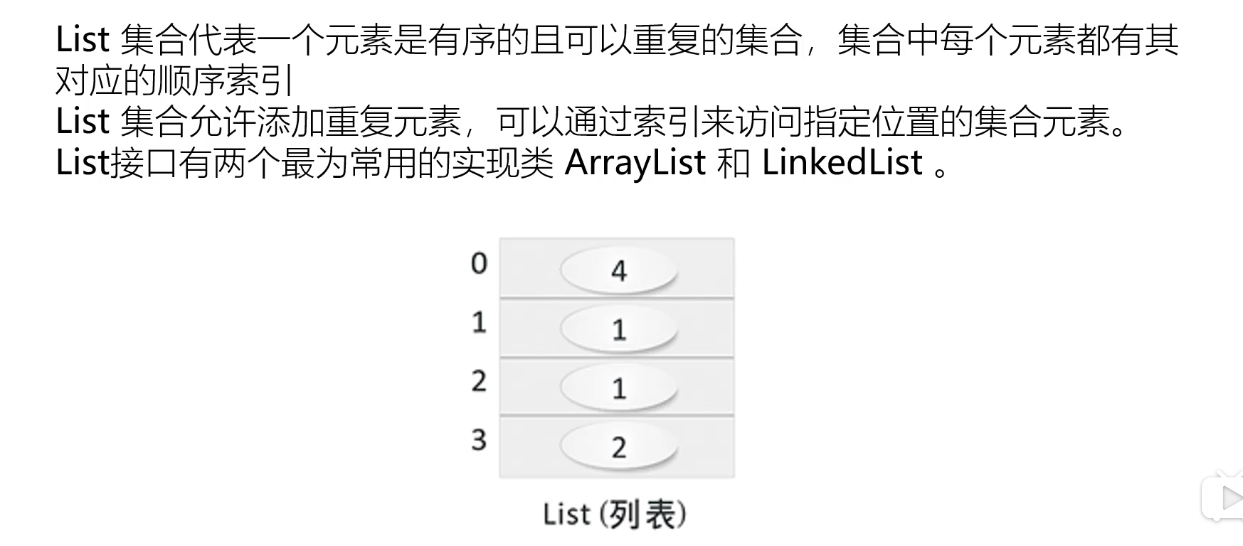
**default** Stream<E> parallelStream() {

**return** StreamSupport.*stream*(spliterator(), **true**);

}

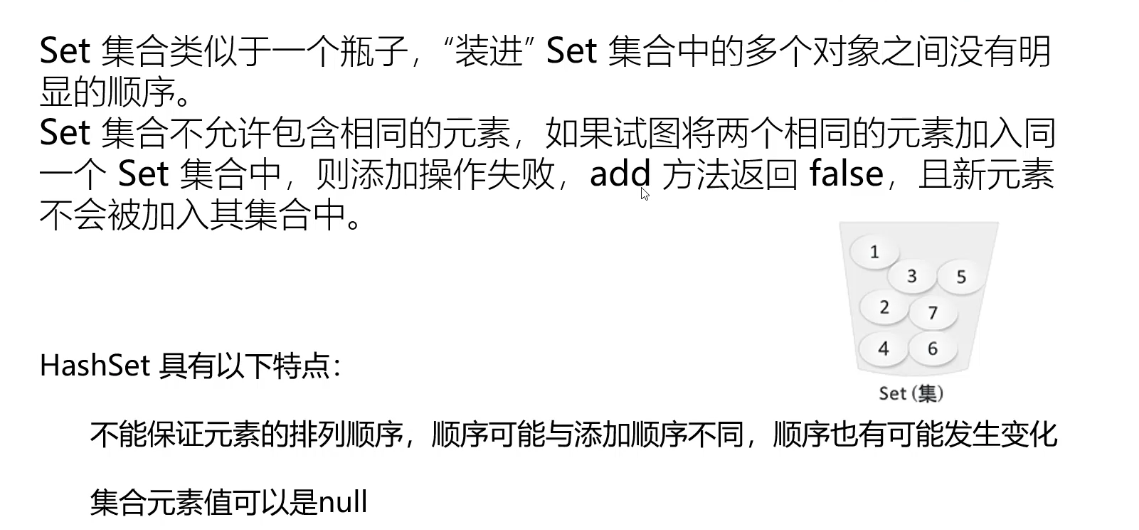
}

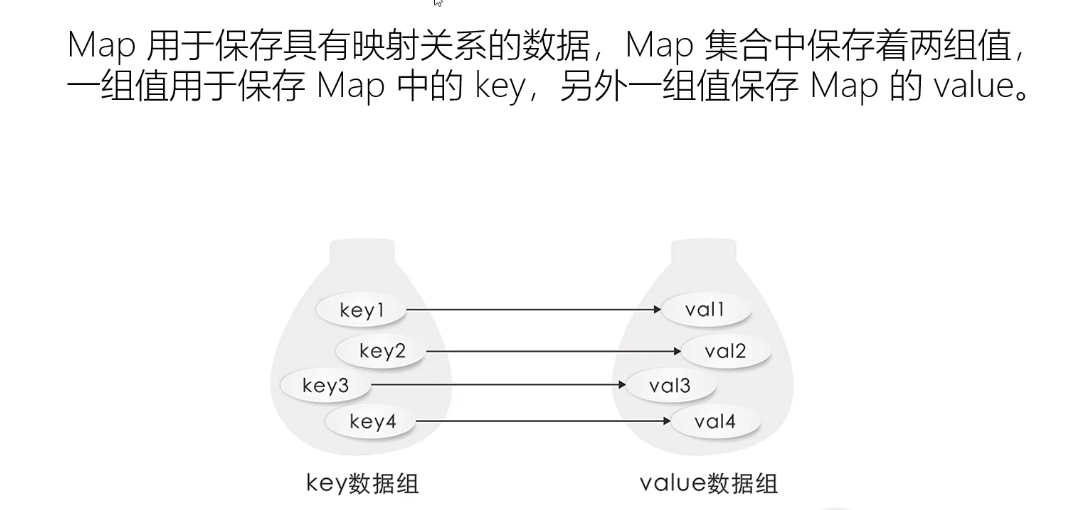




List他是对等，的先被加载的先执行

Set是不对等的





深入理解java并发之synchronized实现底层原理

JVM底层ClassLoader源码分析