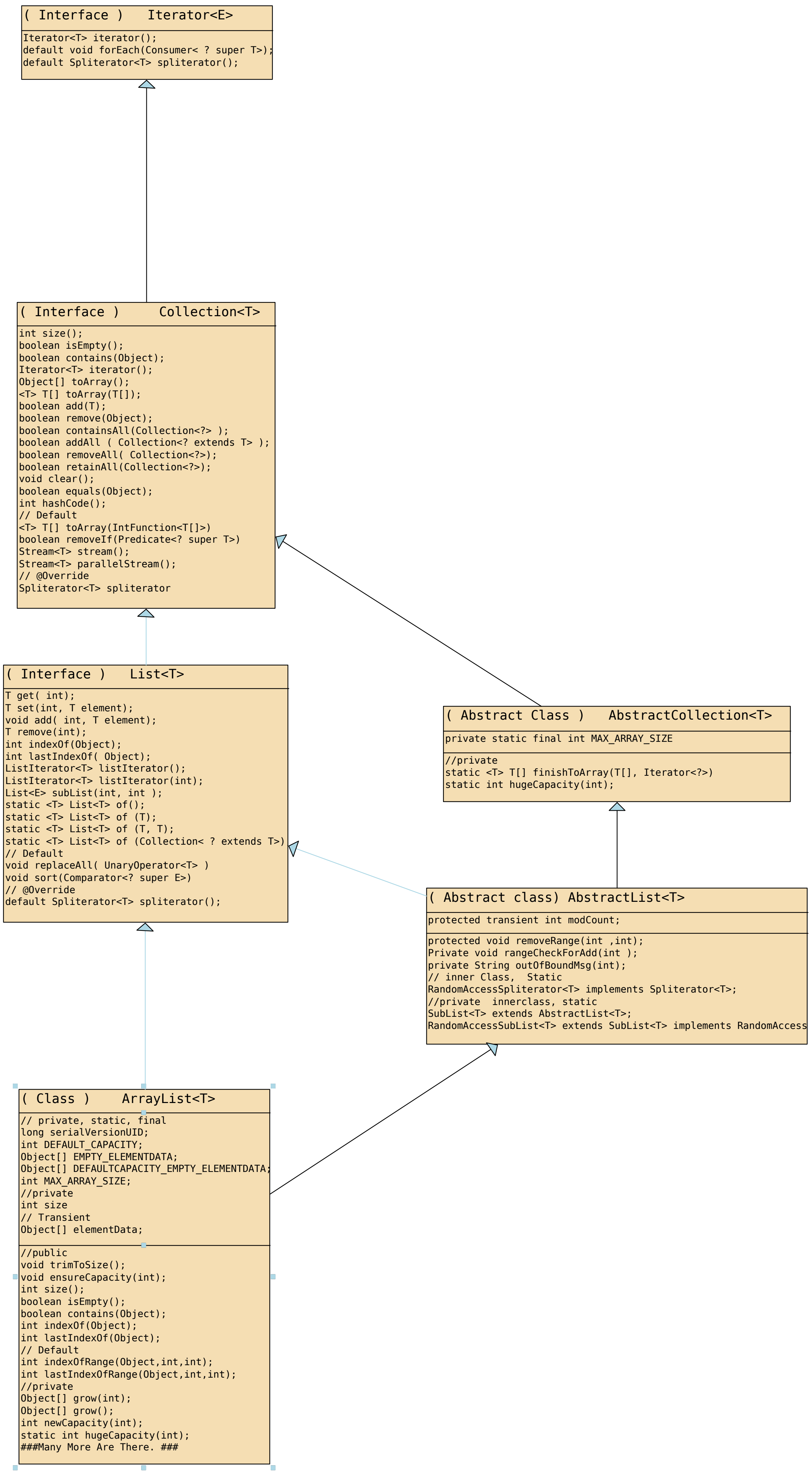

```
( Interface )   Iterator<E>
Iterator<T> iterator();
default void forEach(Consumer< ? super T>);
default Spliterator<T> spliterator();
```

```
( Interface )     Collection<T>
int size();
boolean isEmpty();
boolean contains(Object);
Iterator<T> iterator();
Object[] toArray();
<T> T[] toArray(T[]);
boolean add(T);
boolean remove(Object);
boolean containsAll(Collection<?> );
boolean addAll ( Collection<? extends T> );
boolean removeAll( Collection<?>);
boolean retainAll(Collection<?>);
void clear();
boolean equals(Object);
int hashCode();
// Default
<T> T[] toArray(IntFunction<T[]>)
boolean removeIf(Predicate<? super T>)
Stream<T> stream();
Stream<T> parallelStream();
// @Override
Spliterator<T> spliterator
```

```
( Interface )   List<T>
T get( int);
T set(int, T element);
void add( int, T element);
T remove(int);
int indexOf(Object);
int lastIndexOf( Object);
ListIterator<T> listIterator();
ListIterator<T> listIterator(int);
List<E> subList(int, int );
static <T> List<T> of();
static <T> List<T> of (T);
static <T> List<T> of (T, T);
static <T> List<T> of (Collection< ? extends T>)
// Default
void replaceAll( UnaryOperator<T> )
void sort(Comparator<? super E>)
// @Override
default Spliterator<T> spliterator();
```

```
( Abstract Class )   AbstractCollection<T>
private static final int MAX_ARRAY_SIZE
//private
static <T> T[] finishToArray(T[], Iterator<?>)
static int hugeCapacity(int);
```

```
( Abstract class) AbstractList<T>
protected transient int modCount;
protected void removeRange(int ,int);
Private void rangeCheckForAdd(int );
private String outOfBoundMsg(int);
// inner Class,  Static
RandomAccessSpliterator<T> implements Spliterator<T>;
//private  innerclass, static
SubList<T> extends AbstractList<T>;
RandomAccessSubList<T> extends SubList<T> implements RandomAccess
```

```
( Class )     ArrayList<T>
// private, static, final
long serialVersionUID;
int DEFAULT_CAPACITY;
Object[] EMPTY_ELEMENTDATA;
Object[] DEFAULTCAPACITY_EMPTY_ELEMENTDATA;
int MAX_ARRAY_SIZE;
//private
int size
// Transient
Object[] elementData;
//public
void trimToSize();
void ensureCapacity(int);
int size();
boolean isEmpty();
boolean contains(Object);
int indexOf(Object);
int lastIndexOf(Object);
// Default
int indexOfRange(Object,int,int);
int lastIndexOfRange(Object,int,int);
//private
Object[] grow(int);
Object[] grow();
int newCapacity(int);
static int hugeCapacity(int);
###Many More Are There. ###
```