

# **Основы программирование: Введение в Java**

**Лекция 10. Коллекции.**

**Власенко Олег Федосович**

# Задача 1 – Знакомство с ArrayList

Создать коллекцию ArrayList. Добавить в нее 3 элемента типа String. Вывести элементы, содержащиеся в коллекции в консоль при помощи итератора.

# Задача 1 – Решение

```
ArrayList v = new ArrayList();  
v.add("10");  
v.add("2");  
v.add("30");
```

```
Iterator iterator = v.iterator();  
while (iterator.hasNext()) {  
    Object obj = iterator.next();  
    System.out.println(obj);  
}
```

# Пример

Необходимо на основе текстового файла «Текст» создать файл со словарем «Словарь».

Пример:

«Текст»:

*Мама мыла раму. Мама мыла яблоко*

«Словарь»:

*мама*

*мыла*

*раму*

*яблоко*

# Пример (идея реализации)

Алгоритм:

- 1). Читаем файл «Текст» посимвольно, вычленяя слова из текста.
- 2). Каждое слово добавляем в КОЛЛЕКЦИЮ, если его там не было до этого.
- 3). Коллекцию сохраняем в файле «Словарь»

# Пример: коллекция **Vector**

```
Collection dict = new Vector();
```

1) слово читается из файла «Текст»

```
String word = ...;
```

2) Слово добавляется в коллекцию, если его там не было

```
if (!dict.contains(word)) {
```

```
    dict.add(word);
```

```
}
```

3) Коллекция сохраняется в файле

```
BufferedWriter bw = new BufferedWriter(...);
```

```
for (Object word : dict) {
```

```
    bw.write((String)word);
```

```
    bw.newLine();
```

```
}
```

```
bw.close();
```

# Пример:

## коллекция ArrayList

```
Collection dict = new ArrayList();
```

1) слово читается из файла «Текст»

```
String word = ...;
```

2) Слово добавляется в коллекцию, если его там не было

```
if (!dict.contains(word)) {
```

```
    dict.add(word);
```

```
}
```

3) Коллекция сохраняется в файле

```
BufferedWriter bw = new BufferedWriter(...);
```

```
for (Object word : dict) {
```

```
    bw.write((String)word);
```

```
    bw.newLine();
```

```
}
```

```
bw.close();
```

# Чем отличаются Vector и ArrayList?

<http://www.quizful.net/interview/java/vector-and-arraylist-difference>

Вопрос

В чем принципиальное отличие классов  
Vector и ArrayList

Ответ

Методы класса Vector синхронизированы,  
в то время как ArrayList - нет.



# Потоки и синхронизация

- Коротко про потоки
- Коротко про синхронизацию

# Разбор кода примера

- DictionaryLoaderTest
- DictionaryLoader.saveDictToFile
  - Работа с файлами при записи
- DictionaryLoader.loadTextToDict
  - Разбор алгоритма
    - Флаги – inWord

# Обработка исключений

- Exception
  - *IOException*
    - FileNotFoundException
  - Правила перехвата с учетом иерархии классов исключений

# Демонстрация работы

Collection dict = new LinkedList();

vs

Collection dict = new HashSet();

# Пример: коллекция TreeSet

```
Collection dict = new TreeSet();
```

1) слово читается из файла «Текст»

```
String word = ...;
```

2) Слово добавляется в коллекцию, если его там не было

```
if (!dict.contains(word)) {
```

```
    dict.add(word);
```

```
}
```

3) Коллекция сохраняется в файле

```
BufferedWriter bw = new BufferedWriter(...);
```

```
for (Object word : dict) {
```

```
    bw.write((String)word);
```

```
    bw.newLine();
```

```
}
```

```
bw.close();
```

# Чем отличаются Vector и TreeSet?

Время работы с Vector: 758 мс

Время работы с TreeSet: 104 мс

# Пример:

## разные коллекции

```
Collection dict = new Vector();
```

```
Collection dict = new ArrayList();
```

```
Collection dict = new LinkedList();
```

```
Collection dict = new TreeSet(); // элементы упорядочены
```

```
    // по значению, но не по порядку добавления
```

```
Collection dict = new HashSet(); // порядок обхода неопределен
```

```
Collection dict = new LinkedHashSet(); // порядок обхода
```

```
    // элементов = порядку добавления элементов
```

1) слово читается из файла «Текст»

```
String word = ...;
```

2) Слово добавляется в коллекцию, если его там не было

```
if (!dict.contains(word)) {
```

```
    dict.add(word);
```

```
}
```

3) Коллекция сохраняется в файле

```
BufferedWriter bw = new BufferedWriter(...);
```

```
for (Object word : dict) {
```

```
    bw.write((String)word);
```

```
    bw.newLine();
```

```
}
```

```
bw.close();
```

# Пример:

## разные коллекции – время работы

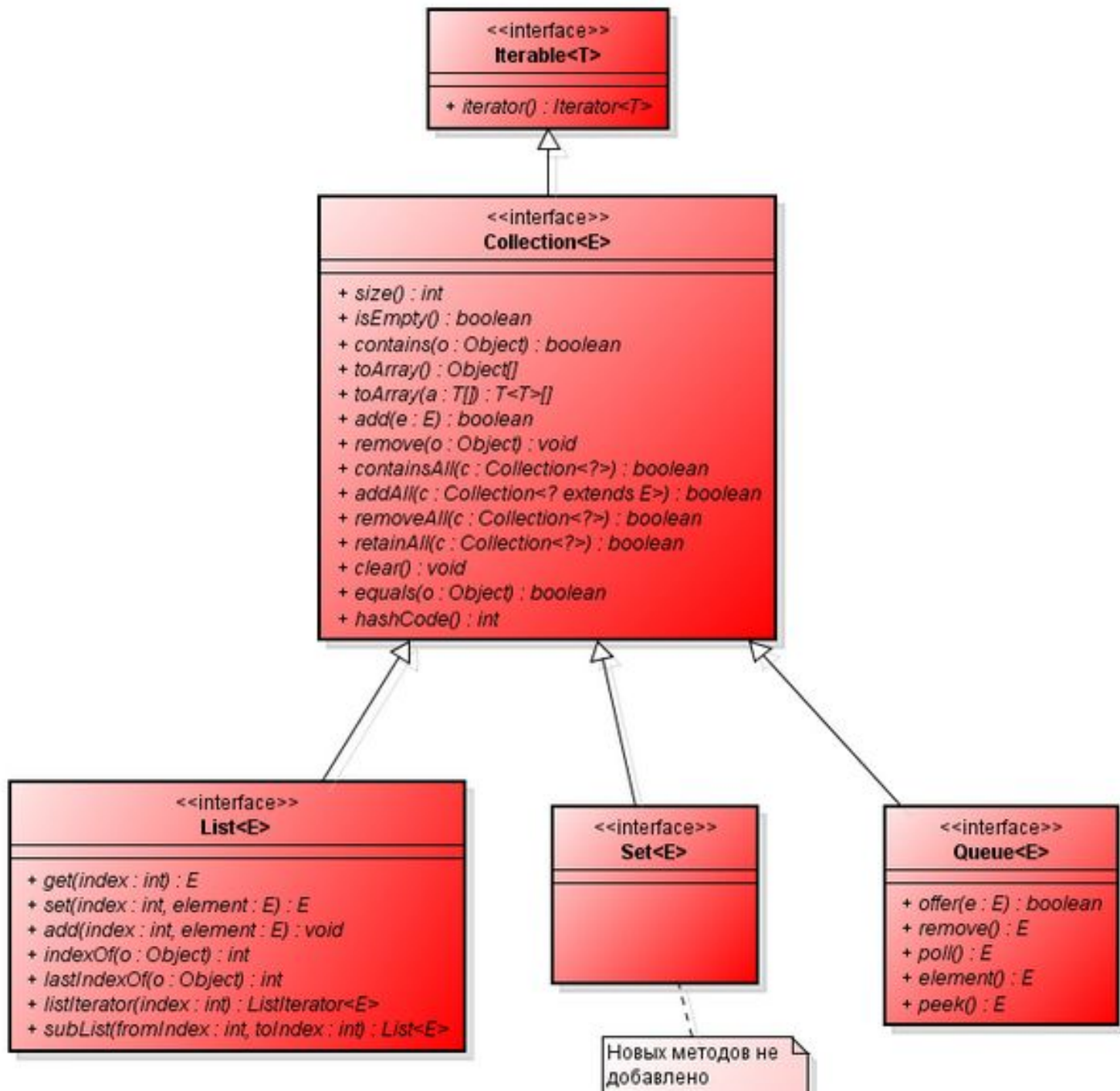
	170 Kb	4,4 Mb
ArrayList	0.713	16.151
LinkedList	0.753	18.828
Vector	0.697	16.020
TreeSet	0.059	0.532
LinkedHashSet	0.051	0.397
HashSet	0.044	0.372

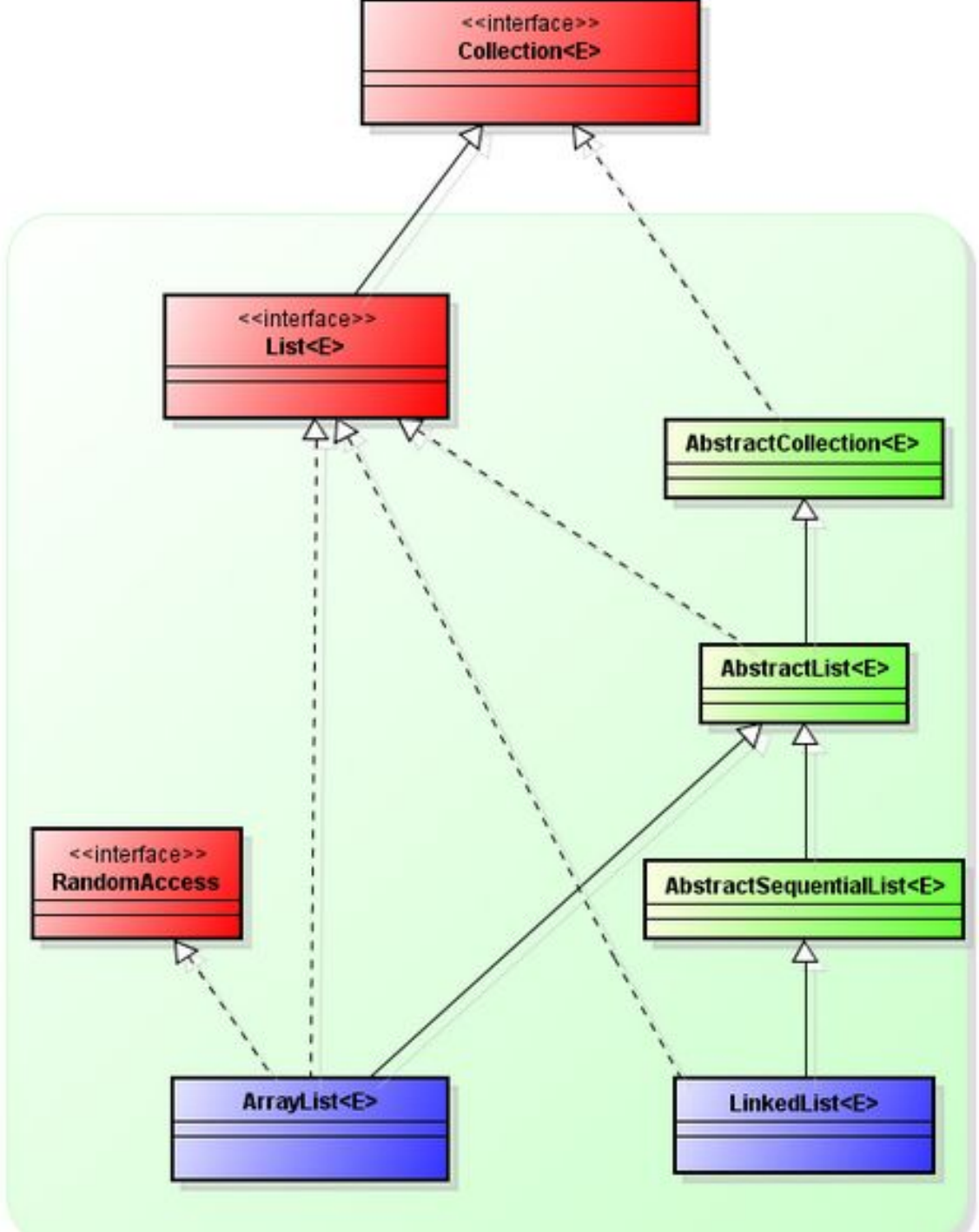


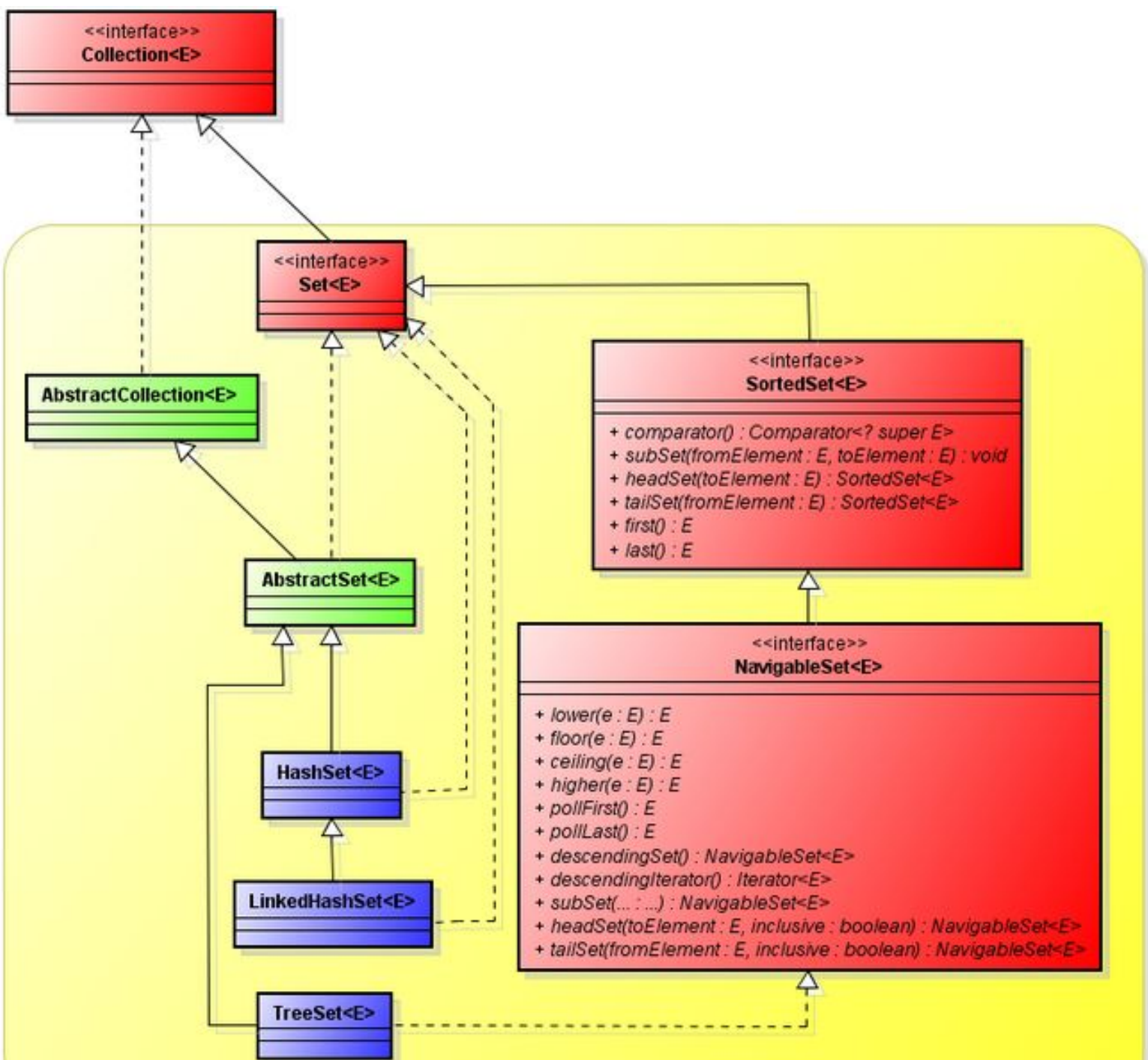
# Иерархии коллекций

- Источник:

<http://www.quizful.net/post/Java-Collections>







# Итераторы

```
public interface Iterator<E>
```

Modifier and Type	Method and Description
<b>boolean</b>	<u><a href="#">hasNext()</a></u> Returns true if the iteration has more elements.
<u><a href="#">E</a></u>	<u><a href="#">next()</a></u> Returns the next element in the iteration.
<b>void</b>	<u><a href="#">remove()</a></u> Removes from the underlying collection the last element returned by this iterator (optional operation).

# Generics

- Что хранит

```
List li1 = new ArrayList ();
```

- Что хранит

```
List<Integer> li = new ArrayList<Integer>();
```

# Класс Object

Modifier and Type	Method and Description
protected <u>Object</u>	<u>clone()</u> Creates and returns a copy of this object.
boolean	<u>equals(Object obj)</u> Indicates whether some other object is "equal to" this one.
protected void	<u>finalize()</u> Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
<u>Class</u> <?>	<u>getClass()</u> Returns the runtime class of this Object.
int	<u>hashCode()</u> Returns a hash code value for the object.

# Класс Object (2)

Modifier and Type	Method and Description
<b>void</b>	<a href="#"><u>notify()</u></a> Wakes up a single thread that is waiting on this object's monitor.
<b>void</b>	<a href="#"><u>notifyAll()</u></a> Wakes up all threads that are waiting on this object's monitor.
<a href="#"><u>String</u></a>	<a href="#"><u>toString()</u></a> Returns a string representation of the object.



# Класс Object

Modifier and Type	Method and Description
<b>void</b>	<a href="#">wait()</a> Causes the current thread to wait until another thread invokes the <a href="#">notify()</a> method or the <a href="#">notifyAll()</a> method for this object.
<b>void</b>	<a href="#">wait(long timeout)</a> Causes the current thread to wait until either another thread invokes the <a href="#">notify()</a> method or the <a href="#">notifyAll()</a> method for this object, or a specified amount of time has elapsed.
<b>void</b>	<a href="#">wait(long timeout, int nanos)</a> Causes the current thread to wait until another thread invokes the <a href="#">notify()</a> method or the <a href="#">notifyAll()</a> method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed.

# Использование toString()

```
ArrayList a3 = new ArrayList();  
a3.add("Один");  
a3.add("Два");  
a3.add("Пять");  
  
System.out.println("a3 = " + a3);
```

# Коротко о структурах данных

- Динамический массив
- Список
- Хэш
- Двоичное дерево поиска

# Хэш

Пример про заказы по телефону

Какие решения возможны?

Как искать заказ за 5 секунд вручную?

Структура хэша:

1. Хэш-функция

2. Хэш-таблица

3. Схема разрешения коллизий (список)

# Спасибо за внимание!

Власенко Олег Федосович

E-mail: [vlasenko.oleg@gmail.com](mailto:vlasenko.oleg@gmail.com)

Vk: [vk.com/oleg.f.vlasenko](https://vk.com/oleg.f.vlasenko)

Телефон: 8 902 246 05 47