

Exceptions

Exceptions

Что такое исключение?

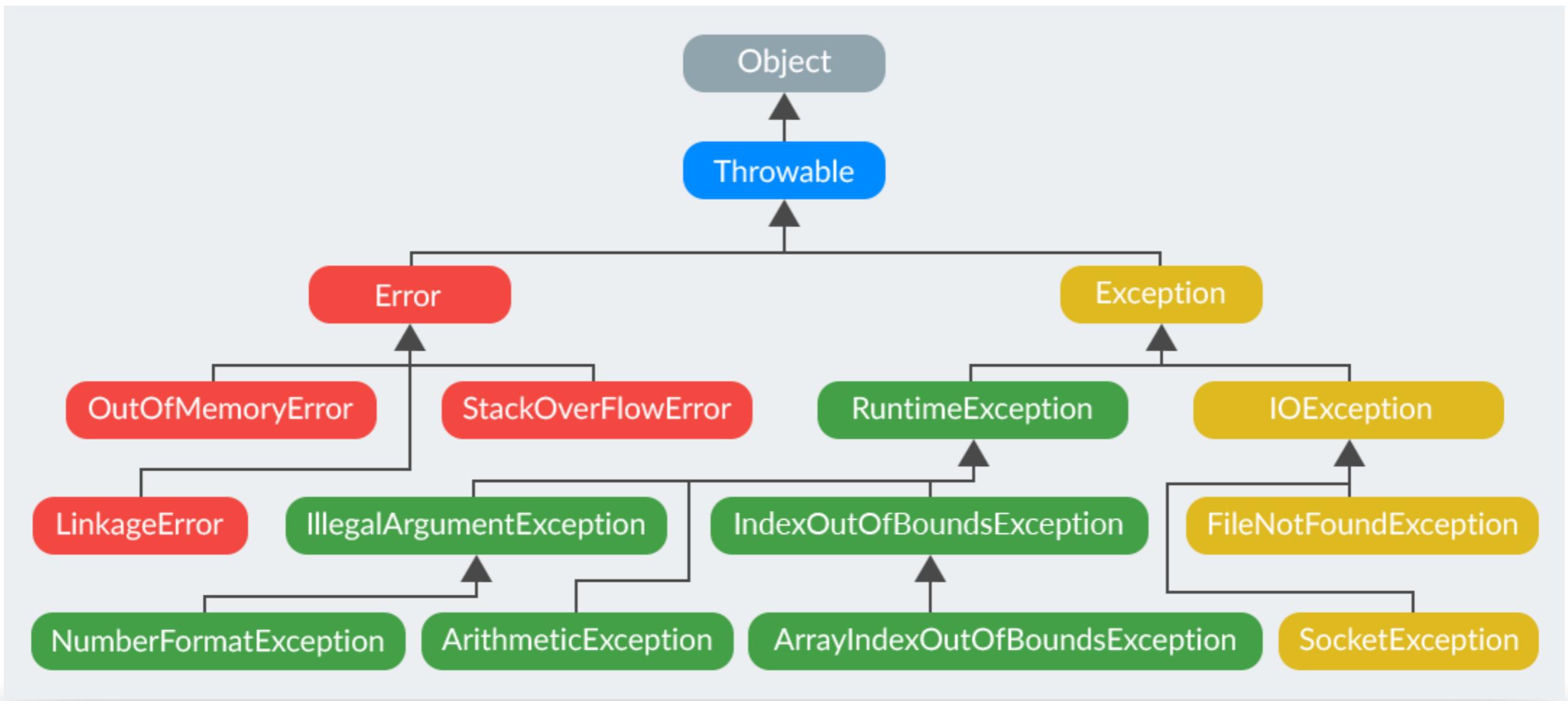
В мире программирования возникновение ошибок и непредвиденных ситуаций при выполнении программы называют **исключением**. В программе исключения могут возникать в результате неправильных действий пользователя, отсутствии необходимого ресурса на диске, или потери соединения с сервером по сети.

Причинами исключений при выполнении программы также могут быть ошибки программирования или неправильное использование **API**. В отличие от нашего мира, программа должна четко знать, как поступать в такой ситуации. Для этого в Java предусмотрен механизм исключений.

Исключение в Java — это объект, который описывает исключительное состояние, возникшее в каком-либо участке программного кода.

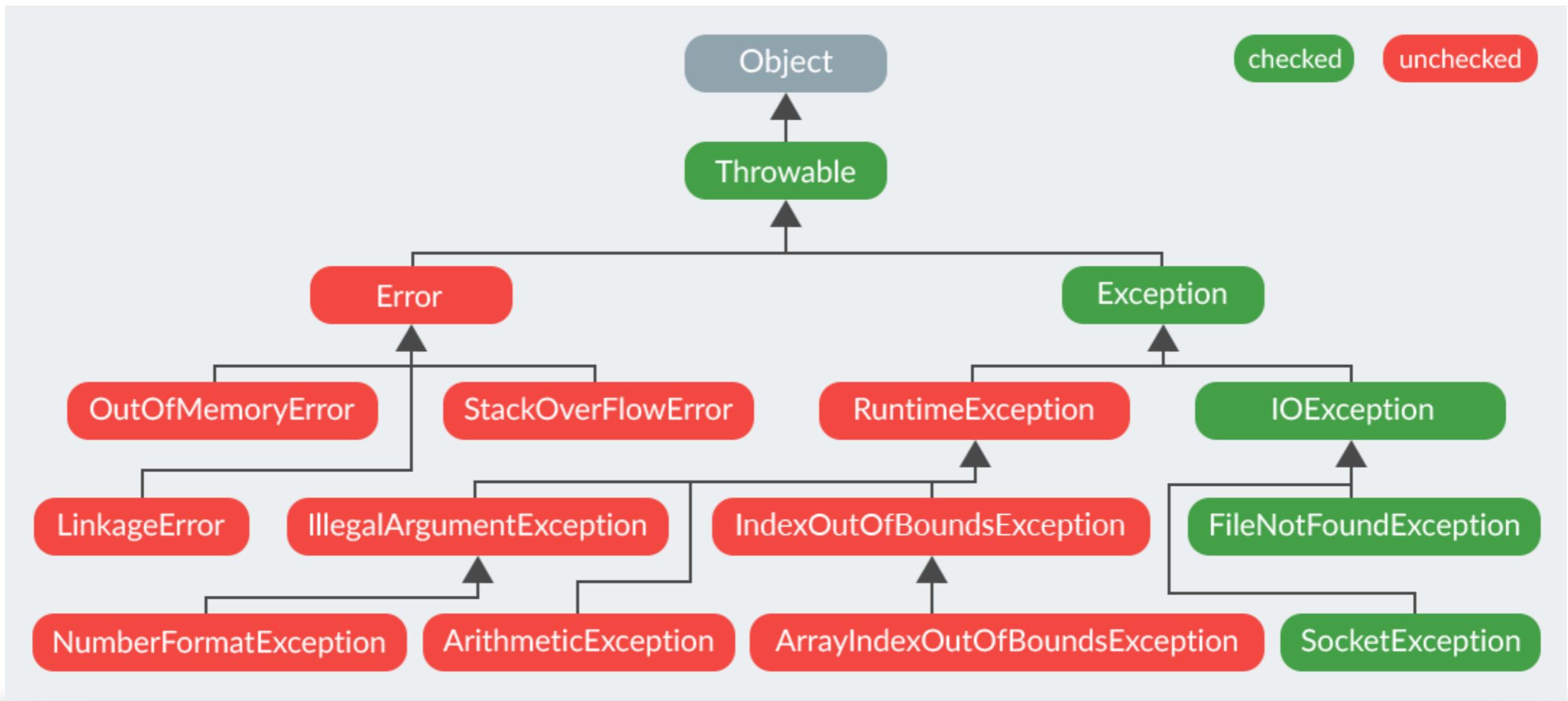
Exceptions

Виды исключений:



Exceptions

Виды исключений:



Exceptions

Кратко о ключевых словах:

Обработка исключений в Java основана на использовании в программе следующих ключевых слов:

- **try** – определяет блок кода, в котором может произойти исключение;
- **catch** – определяет блок кода, в котором происходит обработка исключения;
- **finally** – определяет блок кода, который является необязательным, но при его наличии выполняется в любом случае независимо от результатов выполнения блока try.

Эти ключевые слова используются для создания в программном коде специальных обрабатывающих конструкций: **try{catch**, **try{catch}{finally}**, **try{finally}**.

- **throw** – используется для возбуждения исключения;
- **throws** – используется в сигнатуре методов для предупреждения, о том что метод может выбросить исключение.

```
1 //метод считывает строку с клавиатуры
2
3 public String input() throws MyException {//предупреждаем с помощью throws,
4 // что метод может выбросить исключение MyException
5     BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
6     String s = null;
7     //в блок try заключаем код, в котором может произойти исключение, в данном
8     // случае компилятор нам подсказывает, что метод readLine() класса
9     // BufferedReader может выбросить исключение ввода/вывода
10    try {
11        s = reader.readLine();
12        // в блок catch заключаем код по обработке исключения IOException
13    } catch (IOException e) {
14        System.out.println(e.getMessage());
15        // в блоке finally закрываем поток чтения
16    } finally {
17        // при закрытии потока тоже возможно исключение, например, если он не был открыт, поэтому "об
18        try {
19            reader.close();
20            // пишем обработку исключения при закрытии потока чтения
21        } catch (IOException e) {
22            System.out.println(e.getMessage());
23        }
24    }
25
26    if (s.equals("")) {
27        // мы решили, что пустая строка может нарушить в дальнейшем работу нашей программы, например,
28        throw new MyException("String can not be empty!");
29    }
30    return s;
31 }
```

Exceptions

```
1 class Main {  
2     public static void main(String[] args) {  
3         int a = 4;  
4         System.out.println(a/0);  
5     }  
6 }
```

Exceptions

```
1 class Main {  
2     public static void main(String[] args) {  
3         int a = 4;  
4         System.out.println(a/0);  
5     }  
6 }
```

```
1 Exception in thread "main" java.lang.ArithmetiException: / by zero  
2 at Main.main(Main.java:4)
```

Exceptions

```
1 class Main {  
2     public static void main(String[] args) {  
3         int a = 4;  
4         try {  
5             System.out.println(a/0);  
6         } catch (ArithmetricException e) {  
7             System.out.println("Произошла недопустимая арифметическая операция");  
8         }  
9     }  
10 }
```

Exceptions

```
1 import java.util.Scanner;
2 class Main {
3     public static void main(String[] args) {
4         int[] m = {-1,0,1};
5         Scanner sc = new Scanner(System.in);
6         try {
7             int a = sc.nextInt();
8             m[a] = 4/a;
9             System.out.println(m[a]);
10        } catch (ArithmetricException e) {
11            System.out.println("Произошла недопустимая арифметическая операция");
12        } catch (ArrayIndexOutOfBoundsException e) {
13            System.out.println("Обращение по недопустимому индексу массива");
14        }
15    }
16 }
```

Exceptions

```
1 import java.util.Scanner;
2 class Main {
3     public static void main(String[] args) {
4         int[] m = {-1, 0, 1};
5         int a = 1;
6         Scanner sc = new Scanner(System.in);
7         try {
8             a = sc.nextInt();
9             m[a-1] = 4/a;
10            System.out.println(m[a]);
11        } catch (ArithmetcException e) {
12            System.out.println("Произошла недопустимая арифметическая операция");
13        } catch (ArrayIndexOutOfBoundsException e) {
14            System.out.println("Обращение по недопустимому индексу массива");
15        } catch (Exception e) {
16            System.out.println("Произошло ещё какое-то исключение");
17        }
18    }
19 }
```

Collection

Collection

```
1 public static void main(String[] args) {  
2     String user = "Max";  
3     System.out.println("Hello, " + user);  
4 }
```

Collection

```
1 public static void main(String[] args) {  
2     String user = "Max";  
3     System.out.println("Hello, " + user);  
4 }
```

```
1 import java.util.Arrays;  
2 class Main {  
3     public static void main(String[] args) {  
4         String[] users = new String[2];  
5         users[0] = "Max";  
6         users[1] = "John";  
7         System.out.println("Hello, " + Arrays.toString(users));  
8     }  
9 }
```

Collection

Collection

Коллекция элементов

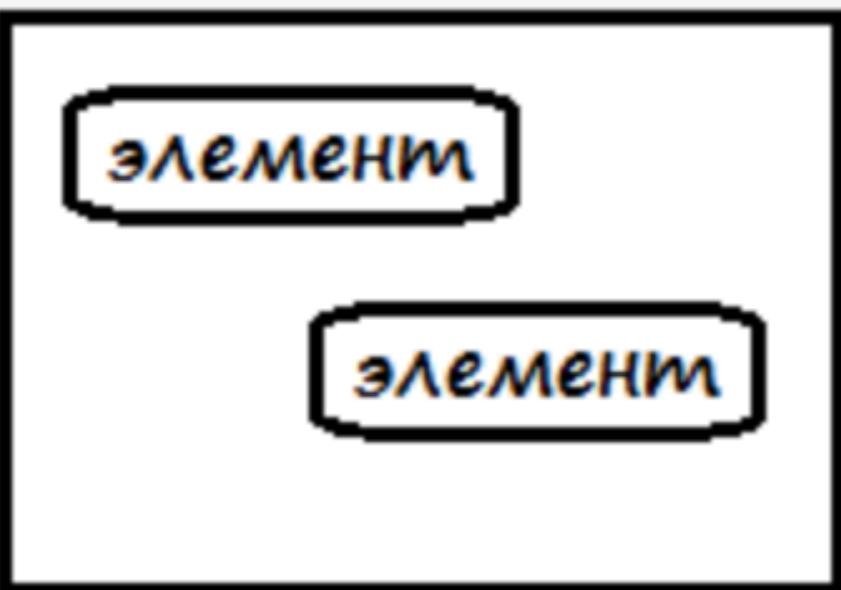
The diagram shows a large rectangular box labeled 'Коллекция элементов'. Inside this box are two smaller rectangular boxes, each containing the word 'элемент'.

Мы хотим:

- Добавлять (add)
- Удалять (remove)
- Очищать (clear)
- Проверить наличие (contains)
- Узнать размер (size)
- Итерироваться (iterator)
- Получить как массив (toArray)

Collection

Коллекция элементов

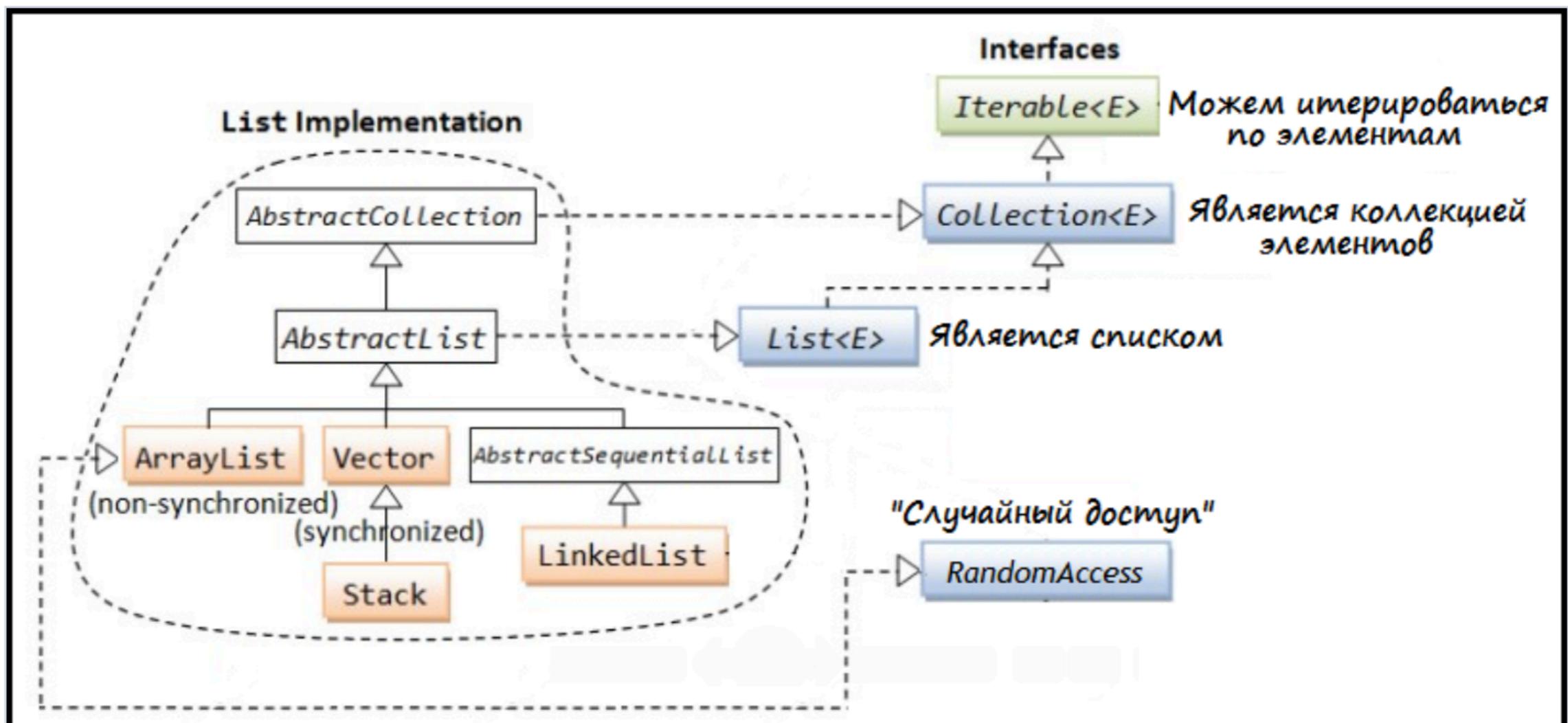


Коллекция элементов



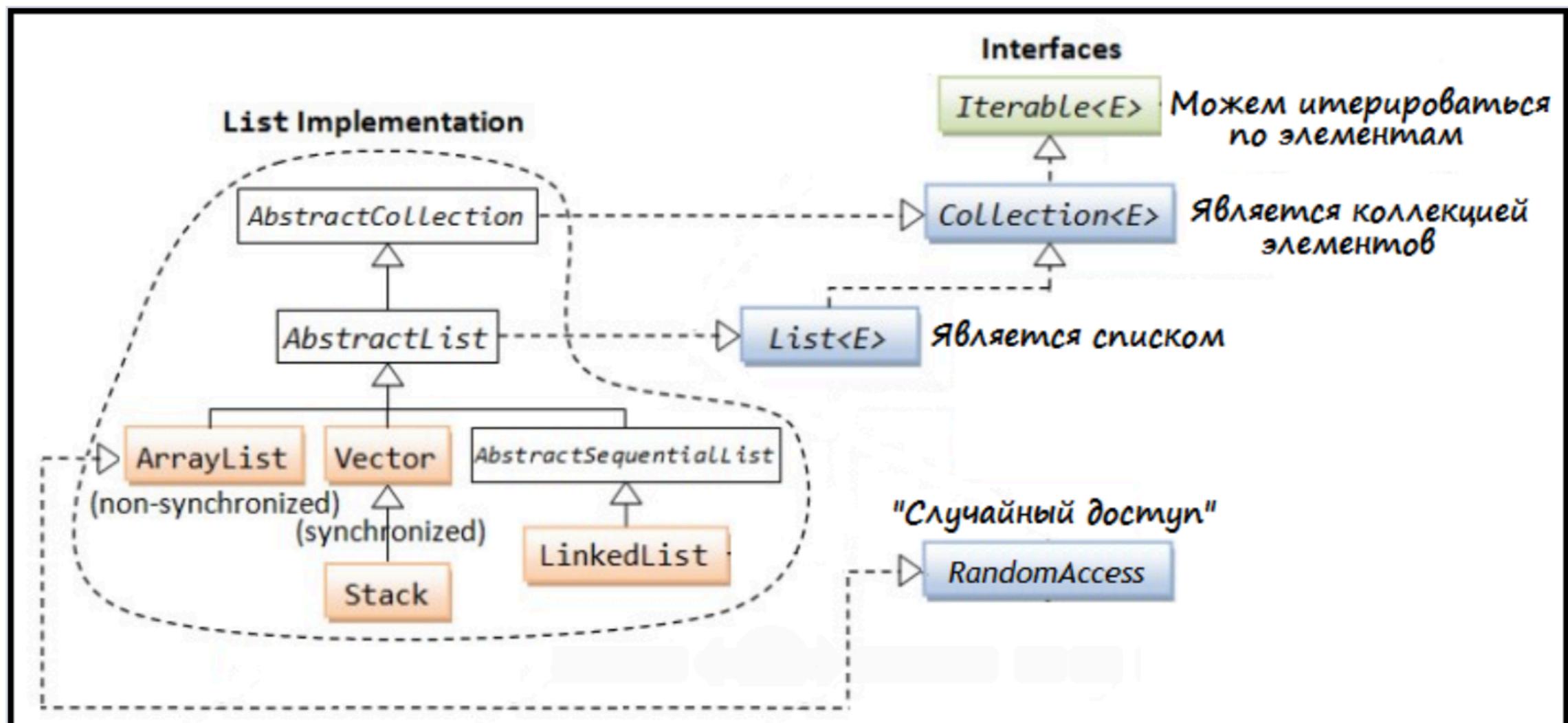
- Добавить из одной в другую (addAll)
- Содержит все элементы из другой (containsAll)
- Удалить все элементы из другой (removeAll)
- Оставить только общие (retainAll)

Collection



Список (List)

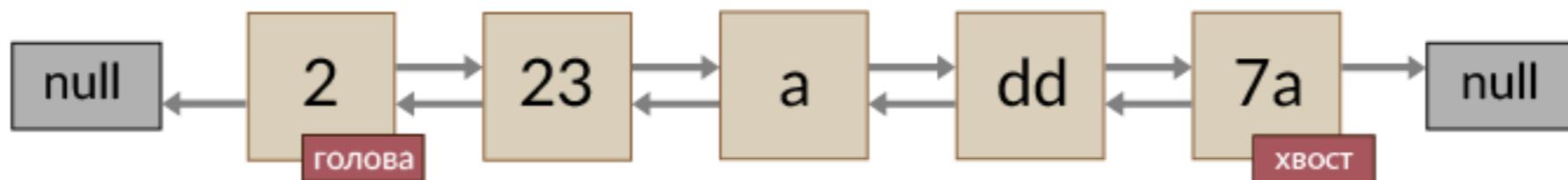
List



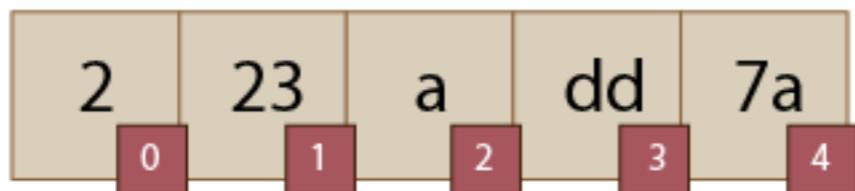
List

ArrayList vs. LinkedList

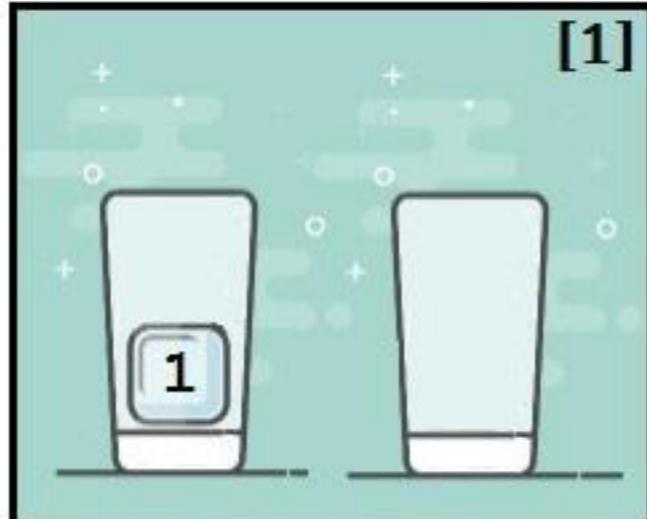
Связный список (LinkedList)



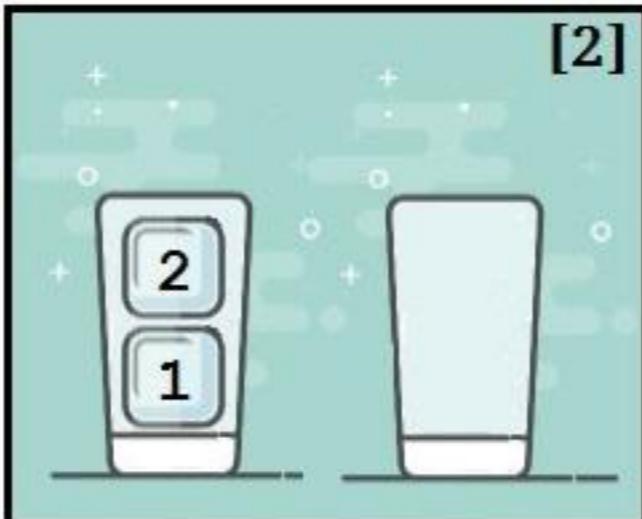
Массив (Array и ArrayList)



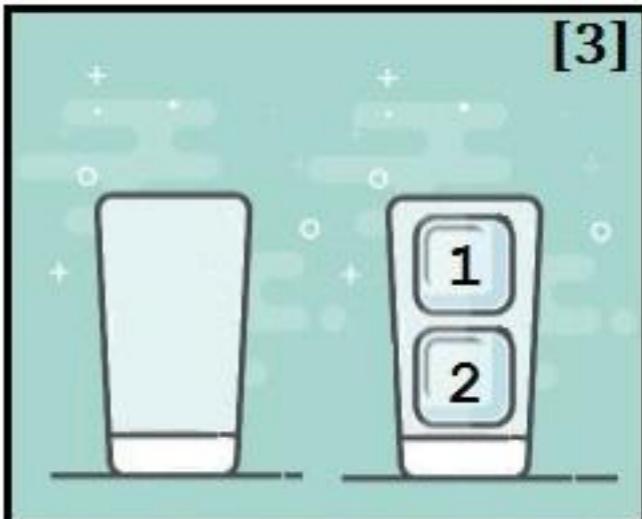
List



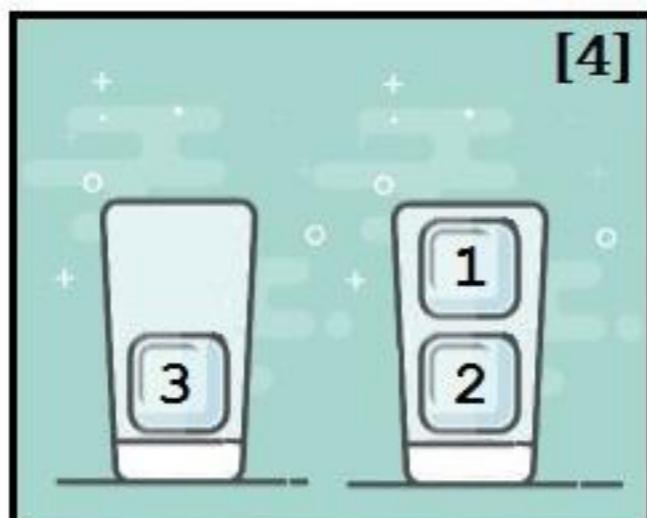
Складываем элементы в левый стэк.
Если слева 1 элемент - проблем нет



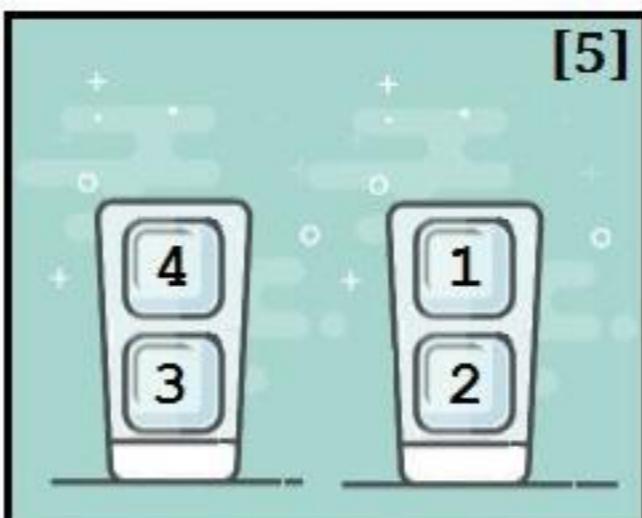
Доступен только последний элемент.
Если нужен элемент 1 - перекладываем



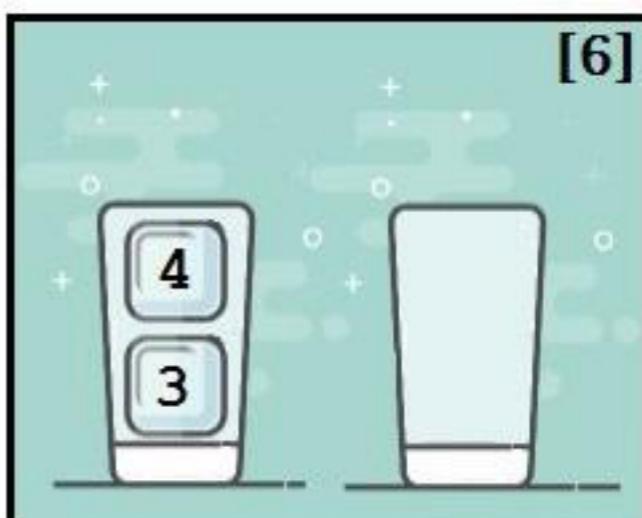
Переложив все - элемент 1 последний



Новые элементы всегда в левый



Вытаскиваем очередь справа
Заполняем очередь слева



Если справа пусто = случай 1 или 2

Очередь (Queue)

Queue

Summary of Queue methods

	<i>Throws exception</i>	<i>Returns special value</i>
Insert	<code>add(e)</code> Добавь	<code>offer(e)</code> Предложим
Remove	<code>remove()</code> Удали	<code>poll()</code> Срезать верхушку
Examine	<code>element()</code> Дай элемент	<code>peek()</code> Посмотреть

Dequeue

Deques can also be used as

LIFO

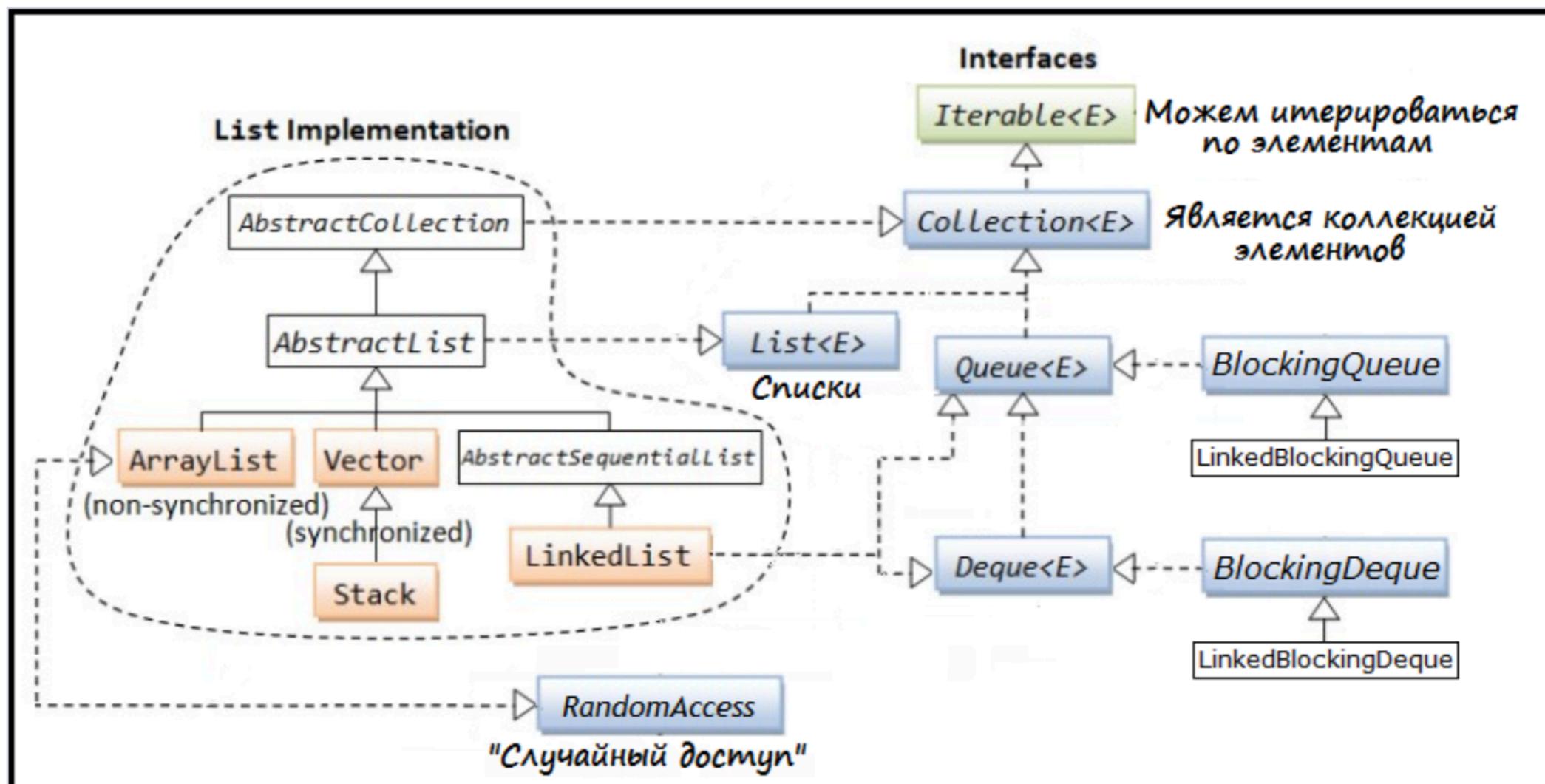
A man with a beard and short hair, wearing a dark suit jacket over a patterned shirt, stands in a room with wooden paneling. He is looking towards the camera with a neutral expression. In his right hand, he holds a white smartphone. His left arm is extended forward, palm facing outwards, as if he is gesturing or about to catch something. The background is slightly blurred, showing more of the room's interior.

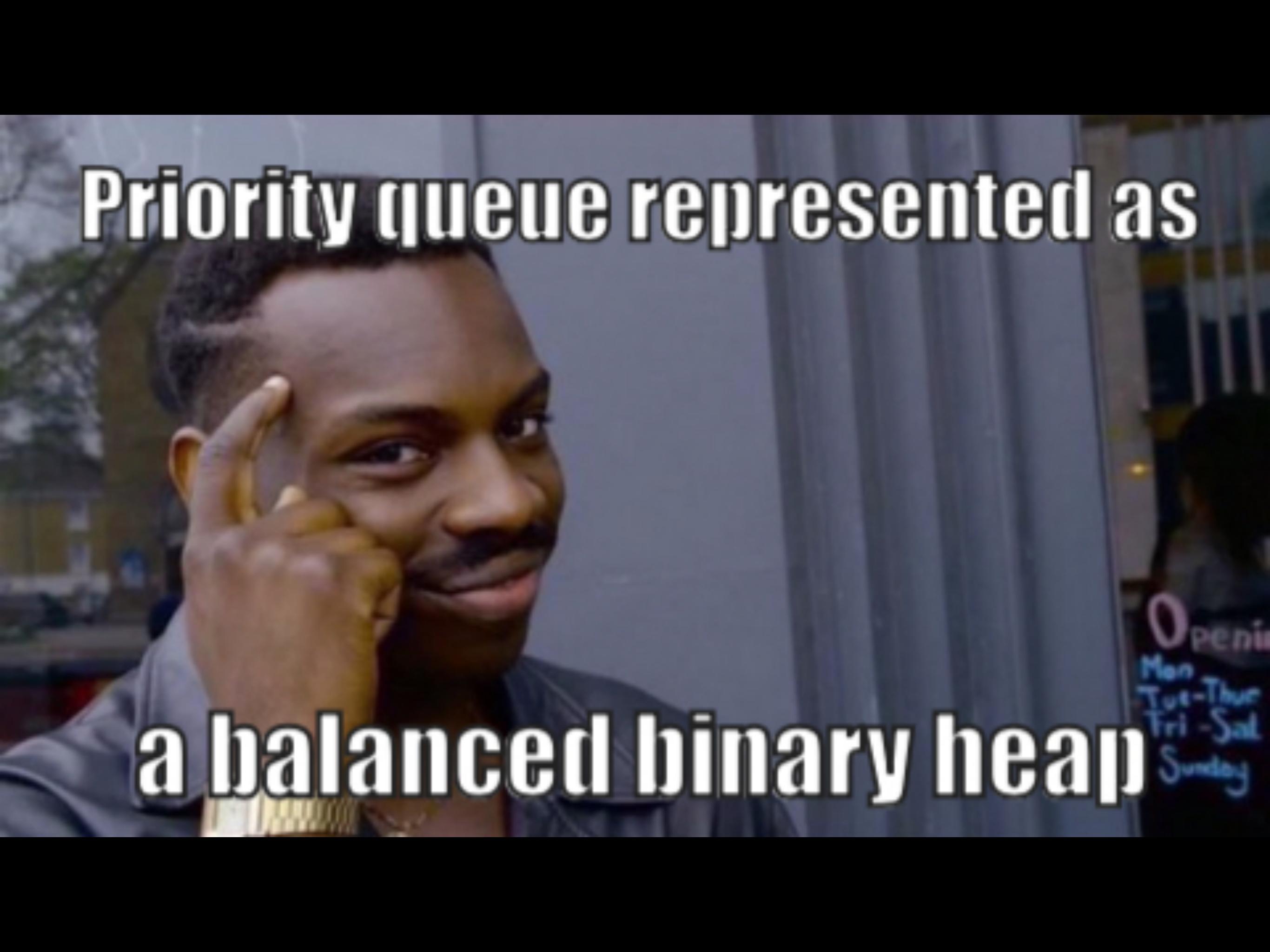
Deque

Summary of Deque methods

	First Element (Head)		Last Element (Tail)	
	<i>Throws exception</i>	<i>Special value</i>	<i>Throws exception</i>	<i>Special value</i>
Insert	<code>addFirst(e)</code>	<code>offerFirst(e)</code>	<code>addLast(e)</code>	<code>offerLast(e)</code>
Remove	<code>removeFirst()</code>	<code>pollFirst()</code>	<code>removeLast()</code>	<code>pollLast()</code>
Examine	<code>getFirst()</code>	<code>peekFirst()</code>	<code>getLast()</code>	<code>peekLast()</code>

Deque

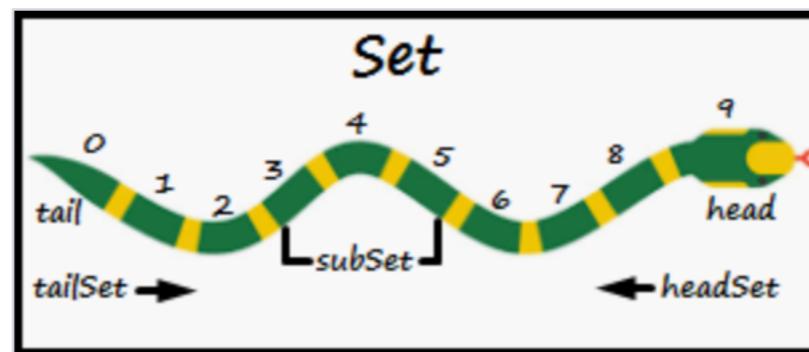


A close-up photograph of a Black man with short hair, resting his chin on his right hand and looking slightly to the side with a thoughtful expression.

Priority queue represented as

a balanced binary heap

Набор (Set)

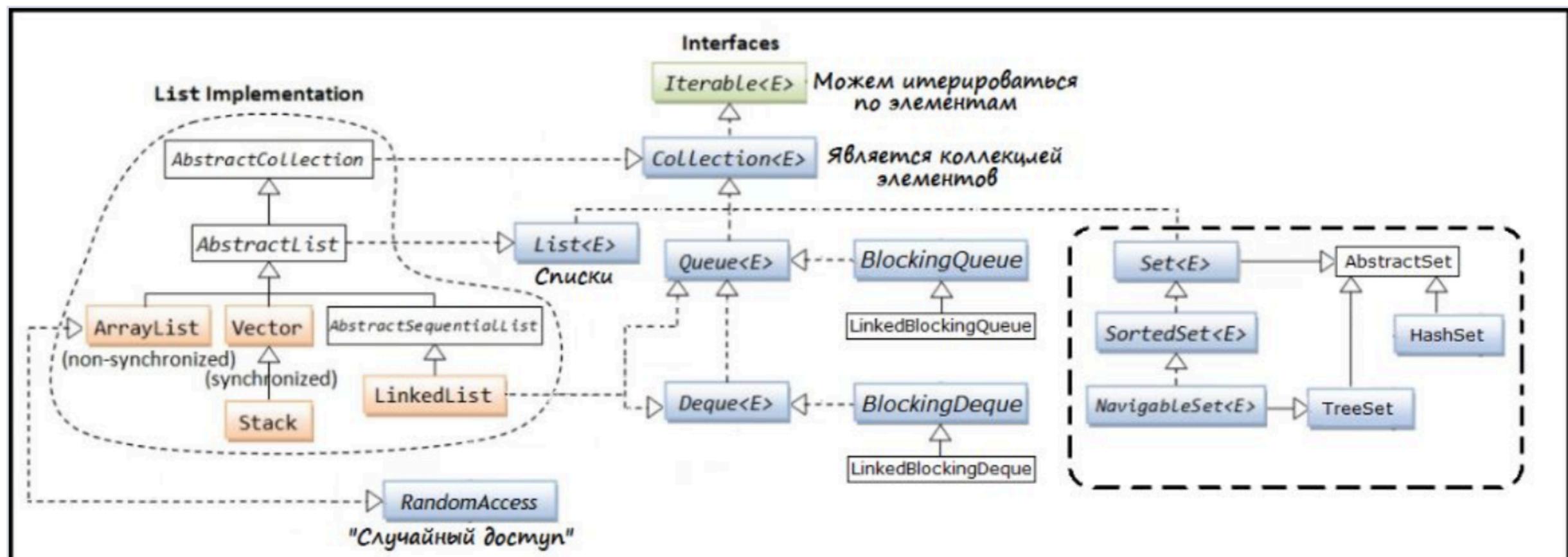


collection that contains



no duplicate elements

Set

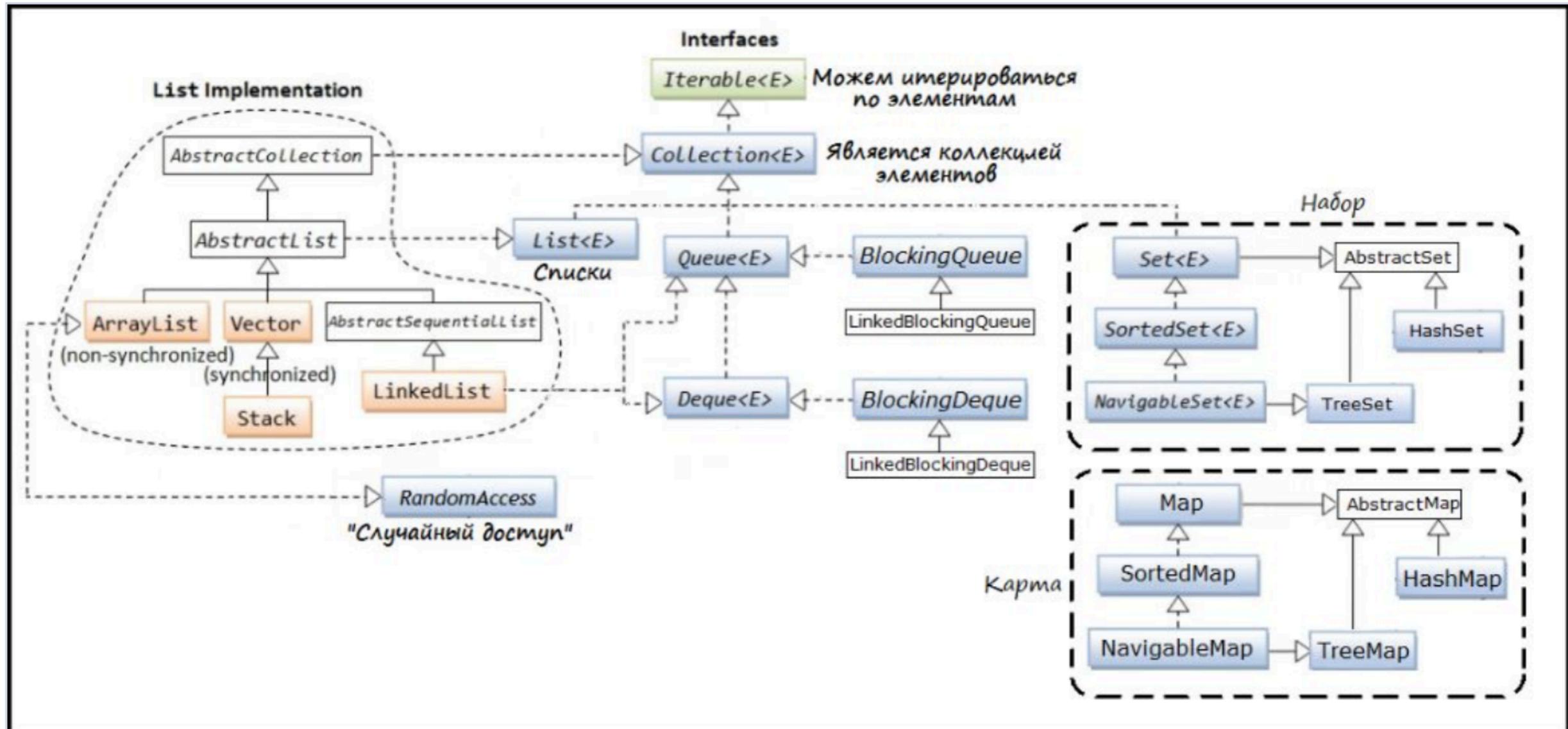


Карты (Map)

Карты (Map)

key	value
1	A
2	B
3	C

Map



Conclusion