



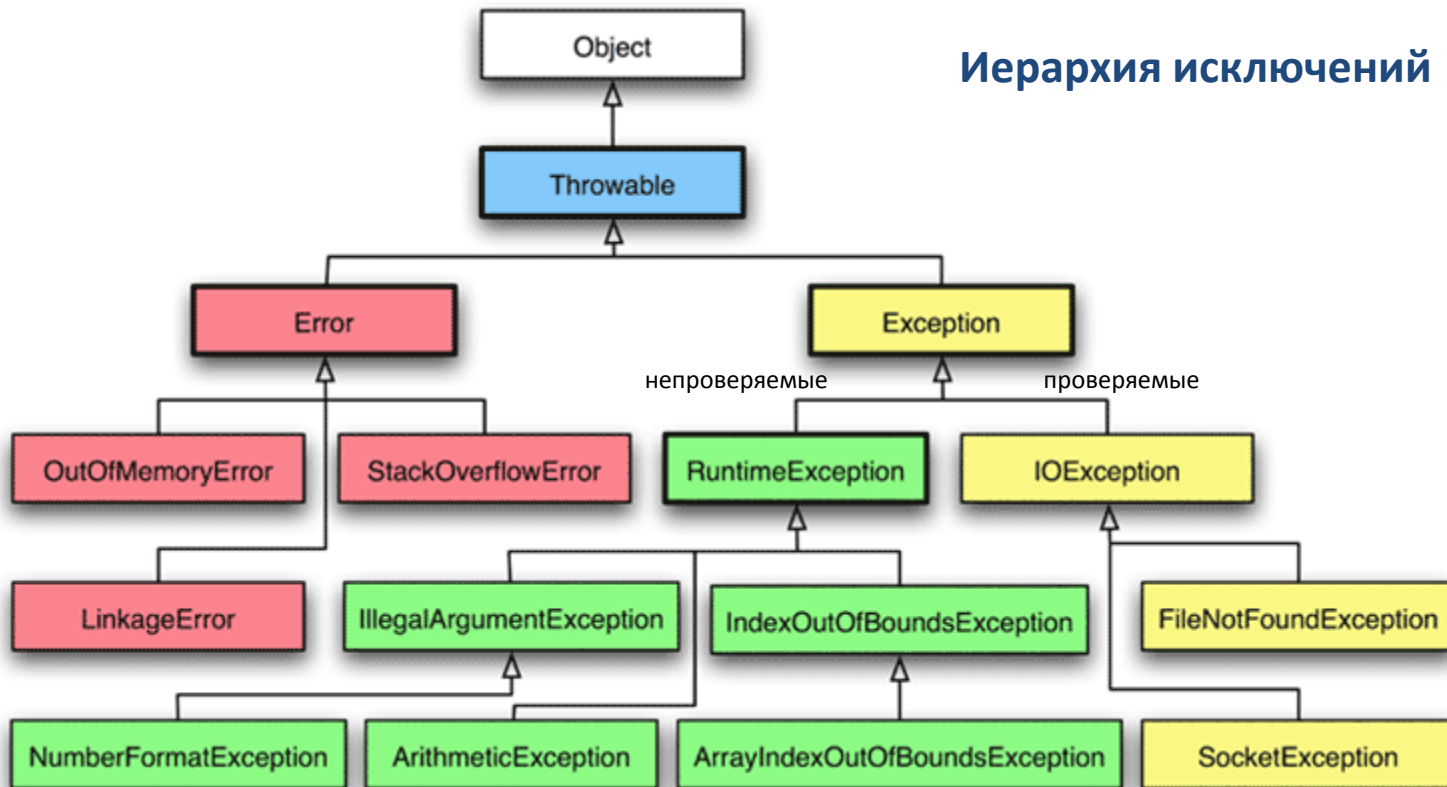
**PerformanceLab**  
Обеспечение производительности ИТ

[www.performance-lab.ru](http://www.performance-lab.ru)

**Исключение** - это проблема(ошибка) возникающая во время выполнения программы. Исключения могут возникать во многих случаях, например:

- Пользователь ввел некорректные данные.
- Файл, к которому обращается программа, не найден.
- Сетевое соединение с сервером было утеряно во время передачи данных.

## Иерархия исключений



Проверяемые исключения должны быть обработаны либо декларированы:

```
public class ExceptionDemo {  
    public void someMethod() throws IOException {  
        throw new IOException("Что-то пошло не так");  
    }  
}
```

Непроверяемые исключения считаются труднопрогнозируемыми. Декларироваться не должны. Пример: `NullPointerException`.

```
try {  
    String str = null;  
    int l = str.length();  
} catch (NullPointerException e) {  
    System.err.println(e.getMessage());  
}  
  
catch (Exception | RuntimeExcetion e) { /* ... */ }  
  
finally {  
    str = "";  
}
```

**Обобщённое программирование** — это такой подход к описанию данных и алгоритмов, который позволяет их использовать с различными типами данных без изменения их описания.

**generics (дженерики)** или «контейнеры типа T» — подмножество обобщённого программирования.



```
class Box<T> {  
    private T val;  
  
    public Box(T arg) { val = arg; }  
    public String toString() { return "Value is: " + val;}  
    public T getValue() { return val; }  
    public void ifCons(Consumer<? super T> cons) { /* ... */}  
    public T ifSup(Supplier<? extends T> sup) { /* ... */}  
    public static <T> T getMiddle(T... a) {  
        return a[a.length / 2];  
    }  
}
```

```
Box<String> = new Box<>("test");
```

Все коллекции являются строго типизированными.  
Базовый интерфейс коллекций: `java.util.Collection<E>`.  
Начиная с 7 версии существуют потокобезопасные реализации коллекций в пакете `java.util.concurrent`.  
Большинство реализаций динамически изменяемы.  
Все коллекции имеют итератор, при чем не разрешается изменять коллекцию во время итерации по ней.



```
List<Integer> collection = new ArrayList();  
//Fill collection  
for (  
    Iterator it = collection.iterator();  
    it.hasNext();  
    Integer element = it.next()  
)  
{ System.out.println(element); }  
//Записи идентичны  
for (Integer element : collection) {  
    System.out.println(element);  
}  
//Записи идентичны  
collection.forEach(System.out::println);
```

## List:

Наиболее используемые реализации: ArrayList, LinkedList.

Интересные методы: indexOf, lastIndexOf, sublist()

## Queue:

Стандартная реализация FIFO. Методы: add/offer, remove/poll, element/peek.

Реализации: ArrayDeque, LinkedList.

## Set:

Обеспечивает уникальность элементов. Реализации: HashSet: хранит элементы в случайном порядке. LinkedHashSet обеспечивает порядок элементов. TreeSet: сортированный список.

## Map:

Словари. Реализация: HashMap, LinkedHashMap, TreeMap.



**PerformanceLab**  
Обеспечение производительности ИТ

**Спасибо за внимание!**