www.performance-lab.ru

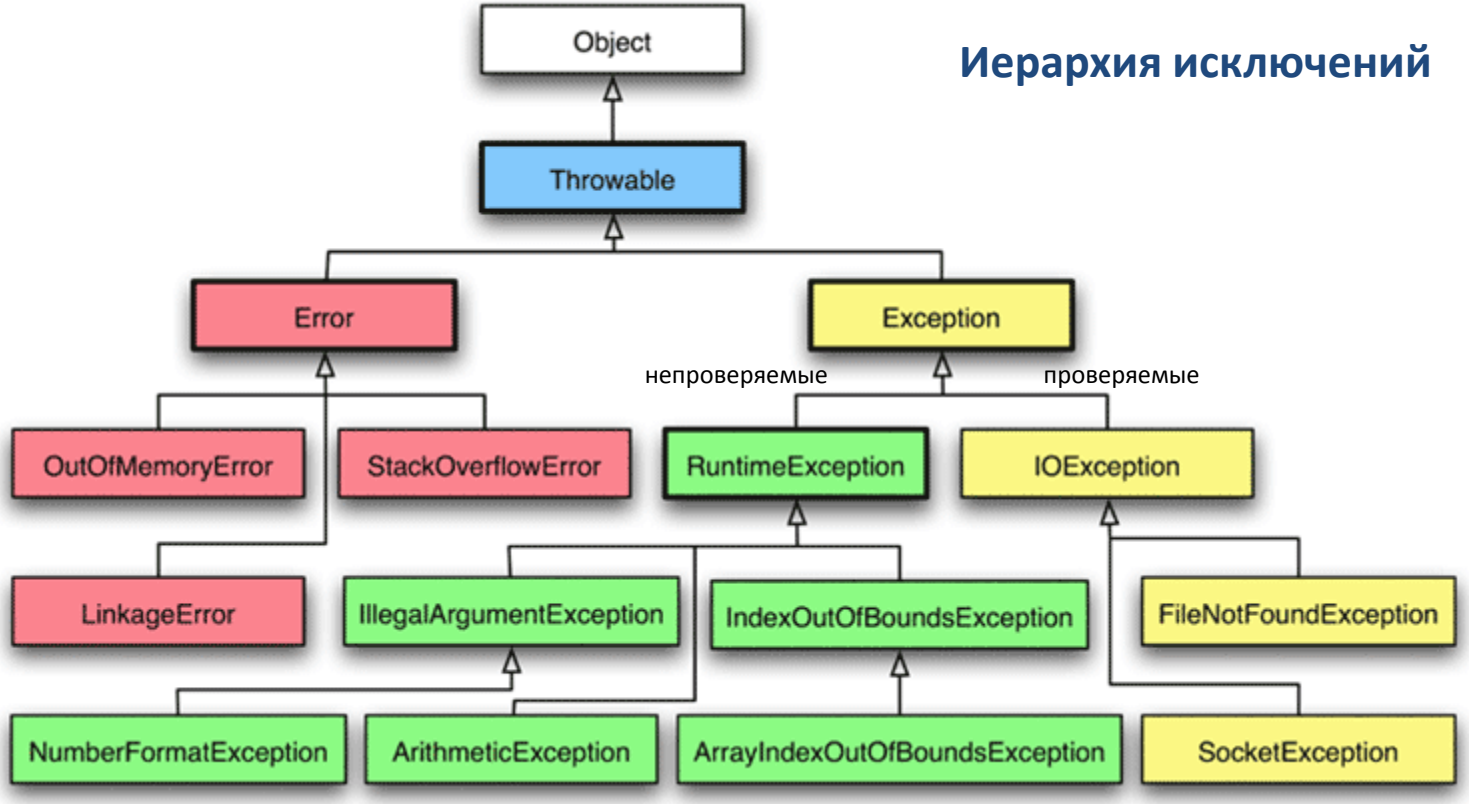
Определение

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.

Исключения

Иерархия исключений



Исключения

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.

Перехват исключений

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

Определение

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.

Параметризованные классы

```
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");
```

Коллекции

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

Итератор коллекции

```
List<Integer> collection = new ArrayList();
//Fill collection
for (
    Iterator it = collection.iteration();
    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);
```

Обзор коллекций

List:

Наиболее используемые реализации: ArrayList, LinkedList.
Интересные методы: indexOf, lasIndexOf, sublist()

Queue:

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

Set:

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

Map:

Словари. Реализация: HashMap, LinkedHashMap, TreeMap.

Функц. интерфейсы

Обычный стиль реализации интерфейса:

```
class MyMouseAdapterImpl() implements MouseAdapter {
    @Override
    public void mouseClicked(MouseEvent e) {
        //Implementation goes here...
    }
};
someObject.addMouseListener(new MyMouseAdapterImpl());
```

Функц. интерфейсы

Анонимный стиль реализации интерфейса:

```
someObject.addMouseListener(new MouseAdapter {
    @Override
    public void mouseClicked(MouseEvent e) {
        //Implementation goes here...
    }
});
```

Функц. интерфейсы

Лямбды:

```
someObject.addMouseListener(e -> {
    //Implementation goes here...
});
```

Собственная лямбда:

```
@FunctionalInterface
public interface TestLyambda{
    int doSomething();
}
```

```
class SomeClass(){
public void someMethod(
  TestLyambda myvar
) {
    //Implementation
  }
}
```

Функц. интерфейсы

Ссылки на методы

- Ссылка на конструктор без агрументов: `SomeClass::new, SomeClass<T>::new;`
- Ссылка на статический метод, должны совпадать кол-во и типы аргументов: `System.out::println`

- Ссылка на метод экземпляра произвольного класса, должны совпадать кол-во и типы аргументов :

```
SomeClass obj = new SomeClass();
obj::doSomething
```

- Ссылка на метод экземпляра произвольного объекта, должны совпадать кол-во и типы аргументов : `SomeClass::doSomething`

Функц. интерфейсы

Старый стиль реализации интерфейса:

```
class MyMouseAdapterImpl() implements MouseAdapter {
    @Override
    public void mouseClicked(MouseEvent e) {
        //Implementation goes here...
    }
};
someObject.addMouseListener(new MyMouseAdapterImpl());
```

Функц. интерфейсы

Анонимный стиль реализации интерфейса:

```
someObject.addMouseListener(new MouseAdapter {
    @Override
    public void mouseClicked(MouseEvent e) {
        //Implementation goes here...
    }
});
```

PerformanceLab

Обеспечение производительности ИТ

Спасибо за внимание!