Урок 6.
Исключения

Гарбузов Николай
Donriver Inc.

DonRiver

Agenda

- Концепция обработки исключений
- Блок try-catch-finally
- Операторы throw & throws
- Типы исключений
- Стандартные исключения
- Создание собственных исключений
- Многократный перехват исключений
- Более точное повторное генерирование исключений

Концепция обработки исключений

Исключение (exception)

- объект, описывающий исключительную (ошибочную) ситуацию

Концепция обработки исключений

Исключение (exception)

- объект, описывающий исключительную (ошибочную) ситуацию
- может быть обработано сразу или передано на обработку вверх по стеку вызова

Концепция обработки исключений

Исключение (exception)

- объект, описывающий исключительную (ошибочную) ситуацию
- может быть обработано сразу или передано на обработку вверх по стеку вызова
- может быть сгенерировано вручную или автоматически JVM

Блок try-catch-finally

```
try {
        // вызов метода,
        // который может кинуть исключения
} catch (Тип_Исключения_1 ex1){
        // обработка ex1
} catch (Тип_Исключения_2 ex2){
        // обработка ex2
}
…
finally{
        // код, который должен выполниться
        // всегда после блока try
}
```

Блок try-catch-finally

```
try {
        // вызов метода,
        // который может кинуть исключения
} catch (Тип_Исключения_1 ex1){
        // обработка ex1
} catch (Тип_Исключения_2 ex2){
        // обработка ex2
}
…
finally{
        // код, который должен выполниться
        // всегда после блока try
}
```

Блок try-catch-finally

```
try {
        // вызов метода,
        // который может кинуть исключения
} catch (Тип_Исключения_1 ex1){
        // обработка ex1
} catch (Тип_Исключения_2 ex2){
        // обработка ex2
}
…
finally{
        // код, который должен выполниться
        // всегда после блока try
}
```

Операторы throw & throws

- throws используется для объявления списка исключений, которых может кинуть метод

```
public void doCredit(Customer customer, Payment payment)
            throws InvalidArgumentException, PaymentProcessingException,
                    BankVerificationException {
     // тело метода
}
```

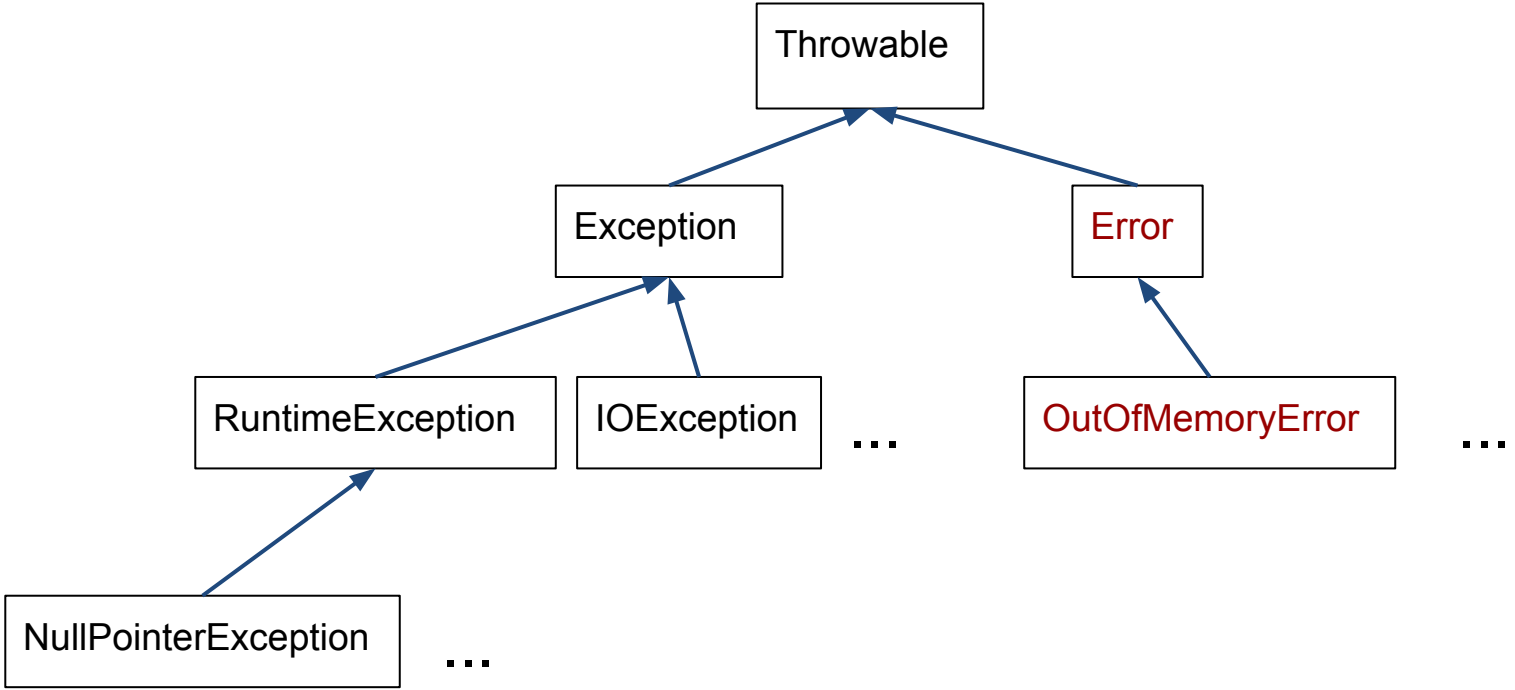
Операторы throw & throws

- throws используется для объявления списка исключений, которых может кинуть метод
- throw используется для генерации исключения в коде

```
public void doCredit(Customer customer, Payment payment)
        throws InvalidArgumentException, PaymentProcessingException,
                BankVerificationException {
    if (customer == null) throw new IllegalArgumentException(...);
    …
    int itemPrice = payment.amount / payment.quantity; // NPE
    if (!bankService.verifyPayment(payment)){
        throw new BankVerificationException(...);
    }
    ...
}
```

DonRiver

Типы исключений



Стандартные исключения

Exception:

- ClassNotFoundException
- IOException
- InterruptedException

Стандартные исключения

RuntimeException:

- NullPointerException
- ArithmeticException
- ArrayIndexOutOfBoundException
- IllegalArgumentException
- SecurityException
- TypeNotPresentException

Стандартные исключения

Error:

- AssertionError
- OutOfMemoryError
- StackOverflowError
- NoClassDefFoundError

Создание собственных исключений

- Исключение - экземпляр класса, унаследованного от Throwable

Создание собственных исключений

- Исключение - экземпляр класса, унаследованного от Throwable
- Для того чтобы создать собственное исключение - нужно унаследоваться от Throwable, Exception, RuntimeException или Error

Создание собственных исключений

- Исключение - экземпляр класса, унаследованного от Throwable
- Для того чтобы создать собственное исключение - нужно унаследоваться от Throwable, Exception, RuntimeException или Error

Создание собственных исключений

- Исключение - экземпляр класса, унаследованного от Throwable
- Для того чтобы создать собственное исключение - нужно унаследоваться от Throwable, Exception, RuntimeException или Error

```
class MyException extends RuntimeException {

    public MyException() {
    }

    public MyException(String message) {
        super(message);
    }
}
```

Многократный перехват исключений

- Позволяет перехватывать несколько исключений в одном catch блоке
- Переменная для многократного перехвата считается завершенной (final по умолчанию)

Многократный перехват исключений

- Позволяет перехватывать несколько исключений в одном catch блоке
- Переменная для многократного перехвата считается завершенной (final по умолчанию)

```
try {
        // вызов метода
} catch (Тип_Исключения_1 | Тип_Исключения_2 ex12){
        // обработка ex1 или ex2
}
```

Более точное повторное генерирование исключений

- Позволяет указать не причину исключения, но исключение которое перетирается новым
- Используют suppressedExceptions поле из Throwable

Более точное повторное генерирование исключений

Q:
Зачем?

Более точное повторное генерирование исключений

Q:

Зачем?

A:

```
try {
        // вызов метода
} catch (Тип_Исключения_1 ex1){
        throw new Тип_исключения2 (ex1);
} finally{
        try {
                // вызов метода
        } catch (Тип_Исключения_3 ex3){
                throw new Тип_исключения4 (ex3);
        }
}
```

DonRiver

Более точное повторное генерирование исключений

Q:

Зачем?

A:

```
try {
        // вызов метода
} catch (Тип_Исключения_1 ex1){
        throw new Тип_исключения2 (ex1);
} finally{
        try {
                // вызов метода
        } catch (Тип_Исключения_3 ex3){
                throw new Тип_исключения4 (ex3);
        }
}
```

DonRiver

Более точное повторное генерирование исключений

Q:

Зачем?

A:

```
Exception ex=null;
try {  // вызов метода
} catch (Тип_Исключения_1 ex1){
        throw (ex = new Тип_исключения2 (ex1));
} finally{
        try {  // вызов метода
        } catch (Тип_Исключения_3 ex3){
                Exception ex4 = new Тип_исключения4 (ex3);
                ex4.addSuppressed(ex);
                throw ex4;
        }
}
```

Home Work

- Добавить обработку негативных кейсов в матричный калькулятор, используя свои исключения