Обработка ошибок и исключений

**Исключение** – это ошибка которая нарушает нормальную работу программы. Каждый раз, когда происходит ошибка создается объект-исключение, ход работы останавливается, а JRE пытается найти того, кто сможет обработать донное исключение.

**Обработчик исключений** – это блок кода, который может обрабатывать объект-исключение.

**Создание объекта-исключение** называется “бросать исключение”.

**Поиск обработчика исключения** проходит довольно просто, сначала проверяется метод, в котором произошла ошибка, если обработчик не найден, то переходим к методу, который вызвал данный метод ищем в нем, и так далее.Если обработчик исключений был не найден программа, то программа завершает работу и выводит информацию об исключении.

**Обработать исключение** — значит “поймать исключение”.

Основные элементы обработки исключений

**throw** – это ключевое слово которое используется чтобы бросить исключениево время работы программы. Иногда нам самим нужно бросить исключения в некоторых ситуациях. Для этого и используется данное слово.

**throws** – это ключевое слово используется в случае если нам нужно бросить исключение в методе и не обрабатывать его, а перебросить для обработки в другую часть. Если мы сами выбрасываем исключения с помощью throw или компилятор ведёт что оно может возникнуть, то компилятор не даст нам запустить программу пока в методе не появится обработчик исключений. Но если перед телом метода указать данное слово и написать все исключения, которые могут возникнуть, то при появлении исключения оно перейдут к блоку, который вызвал данный метод.

**try-catch** – это блок который используется для обработки исключений. В теле блока try описан код который должен работать, но в нем может появится исключение. А блоки catch стоящие после try ловят исключения и исполняют код внутри их тела.

**finally** – это блок который может завершать конструкцию try-catch, но он не обязателен. Этот блок выполнится всегда не смотря на исключения.

Иерархия исключений

**Все исключения делятся на два типа:**

**checked (контролируемые/проверяемые исключения)** - должны обрабатываться блоком catch или описываться в сигнатуре метода (например, throws IOException). Наличие такого обработчика/модификатора сигнатуры проверяются на этапе компиляции;

**unchecked (неконтролируемые/непроверяемые исключения),** к которым относятся ошибки Error, обрабатывать которые не рекомендуется, и исключения времени выполнения, представленные классом RuntimeException и его наследниками;

**Throwable** – это основной интерфейс от который является родителем всех исключений.

**Методы:**

**addSuppressed(Throwable t**) - добавляет заданное исключение в список подавляемых исключений.

**fillInStackTrace()** - возвращает объект класса Throwable, содержащий полную трассировку стека.

**getCause()** - возвращает исключение, лежащее в основе текущего исключения.

**getLocalizedMessage()** - возвращает локализованное описание исключения.

**getMessage()** - возвращает описание исключения.

**getStackTrace()** - возвращает массив, содержащий поэлементную трассировку стека в виде объектов класса StackTraceElement.

**getSuppressed()** - получает подавленные исключения, связанные с вызывающим исключением, и возвращает массив, который содержит результат.

**initCause(Throwable причина исключения)** - связывает входной параметр причина исключения с вызывающим исключением, указывая его как причину этого вызывающего исключения. Возвращает ссылку на исключение. Метод используется при создании цепочек исключений.

**printStackTrace()** - выводит трассировку стека.

**printStackTrace(PrintStream поток вывода)& printStackTrace(PrintWriter поток вывода)** - Метод направляет трассировку стека в заданный поток вывода.

**setStackTrace(StackTraceElement элементы[])** - устанавливает трассировку стека для заданных элементов.

**toString()** - возвращает объект типа String содержащий описание исключения.

**Exception** – это класс который является реализацией Throwable. Предназначен для задания, исключительных условий, которые перехватываются программой. Если требуется объявить свой класс исключение, он наследуется от данного класса. К исключениям данного типа относятся исключения, на которые нельзя не отреагировать. Например, не открылся файл.

**Error** – это класс который является реализацией Throwable. Предназначен для описания исключительных ошибок в среде java. Таких как недостаточно памяти или переполнение стека.

**RuntimeExeption** – это класс который расширяет Exception. К данным исключениям относится ошибки разработчика, к примеру деление на ноль или выход за границы массива.

**Подклассы непроверяемых исключений от класса RuntimeException:**

**ArithmeticException** – арифметическая ошибка (например, деление на ноль);

**ArrayIndexOutOfBoundsException** – индекс за пределами массива;

**ArrayStoreException** – присваивание элементу массива объекта несовместимого типа;

**ClassCastException** – неправильное приведение типов;

**EnumConstantNotPresent** – попытка воспользоваться неопределенным значением перечисления;

**IllegalArgumentException** – недопустимый аргумент при вызове метода;

**IllegalMonitorStateException** – недопустимая контрольная операция;

**IllegalStateException** – неверное состояние среды или приложения;

**IllegalThreadStateException** – несовместимость запрашиваемой операции с текущим состоянием потока выполнения;

**IndexOutOfBoundsException** – выход индекса некоторого типа за допустимые границы;

**NegativeArraySizeException** – создание массива отрицательного размера;

**NullPointerException** – неправильное использование пустой ссылки;

**NumberFormatException** – неправильное преобразование символьной строки в числовой формат;

**SecurityException** – попытка нарушения безопасности;

**StringIndexOutOfBounds** – попытка доступа по индексу за пределами символьной строки;

**TypeNotPresentException** – тип не найден;

**UnsupportedOperationException** – найдена неподдерживаемая операция.

**Ниже приведен их перечень проверяемых исключений:**

**При создании таких исключений их нужно включать в оператор** **throws метода.**

**ClassNotFoundException** – класс не найден;

**CloneNotSupportedException** – попытка клонировать объект из класса, который не реализует интерфейс Cloneable;

**IllegalAccessException** – запрещен доступ к классу;

**InstantiationException** – попытка создать объект абстрактного класса или интерфейса;

**InterruptedException** – один поток выполнения прерван другим потоком;

**NoSuchFieldException** – запрашиваемое поле не существует;

**NoSuchMethodException** – запрашиваемый метод не существует;

**ReflectiveOperationException** – суперкласс исключений, связанных с рефлексией.