**Исключение** – это ошибка, возникающая в процессе выполнения программы.

**Обработчик исключений** – это автоматизированный код, выполняющийся при возникновении ошибки.



Все исключения являются потомками класса ***Throwable***. У него есть два подкласса – ***Exception*** (ошибки программы) и ***Error*** (ошибки машины java и нехватки памяти). Проверяемые исключения обязательны к обработке (иначе код не скомпилируется) и наследуются от ***Exception***, а непроверяемые – на усмотрение разработчика и наследуются от ***RuntimeException*** или ***Error***.

**Стек-трейс** (***stack trace***) – сообщение об ошибке с информацией о месте и причине её появления. Он реализует логику последним вошел, первым вышел.

Основным блоком обработки исключений является инструкция ***try-catch***. Когда в блоке *try* возникает исключение, оно перехватывается соответствующей инструкцией *catch* и обрабатывается в соответствии с ней. Одному блоку *try* могут соответствовать множество блоков *catch* (для разных типов исключений).

Для выполнения операций, связанных с нормальным завершением программы, используется инструкция ***finally***. Она выполняется всегда и не зависит от наличия необработанных исключений. Не рекомендуется использовать в блоке *finally* инструкцию *return*.

*try {*

*// код, из-за которого может произойти исключение }*

*catch (Throwable throwable) { // параметр - тип возможного исключения*

*// код, который выполнится, если произойдёт исключение указанного типа }*

*finally {*

*// необязательный блок, который выполнится в любом случае }*

Условие перехвата исключений суперкласса справедливо и для его подклассов. Исключения обрабатываются в порядке их расположения в программе, поэтому если указать первым тип *Throwable* (суперкласс всех исключений), то все остальные блоки обработки исключений проигнорируются.

Метод ***getMessage()*** возвращает краткое описание ошибки, которое можно вывести/записать.

Метод ***printStackTrace()*** выводит на экран стек-трейс об ошибке.

Метод ***getStackTrace()*** возвращает массив элементов повлекших вызов исключения. С помощью дополнительных методов ***getClassName()***, ***getMethodName()***, ***getFileName()*** и ***getLineNumber()*** можно получить конкретную информацию для каждого элемента массива.

Блоки *try* могут быть вложенными один в другой. Если исключение во внутреннем блоке не будет отработано, то оно распространится на внешний блок.

Для ручной генерации исключений используется инструкция ***throw***. Объект исключения должен быть объектом класса, производного от *Throwable*. Сформированный при этом объект исключения доступен для повторной генерации исключений.

*throw new ArithmeticException();*

Для создания собственного исключения необходимо создать подкласс, производный от ***Exception***, который порожден классом *Throwable*. В новый подкласс необязательно добавлять реализацию каких-либо методов – его можно использовать в качестве исключения сразу по создании.

*public class NonlntResultException extends Exception {*

*int n;*

*int d;*

*public NonlntResultException(int i, int j) { // пишем свой конструктор исключения*

*n = i;*

*d = j; }*

*public String toString() {*

*return "Результат операции " + n + " / " + d + “ не является целым числом"; } }*

*public class CustomExceptDemo {*

*public static void main(String args[]) {*

*int numer[] = { 4, 8, 15, 32, 64, 127, 256, 512 };*

*for(int i=0; i < numer.length; i++) {*

*try {*

*if((numer[I ]% 2) != 0)*

*throw new NonlntResultException(numer[i], denom[i]); }*

*catch (NonlntResultException exc) {*

*System.out.println(exc); } } } }*

Инструкция ***throws*** позволяет обрабатывать исключения вне метода, в котором они возникают (его отлавливает вышестоящий метод или JVM). Он может использоваться в методах и конструкторах.

*public void methodWithException() throws FirstException { // могут указываться через запятую*

*if (какое-то условие) {*

*throw new FirstException(); } } // сгенерировать исключение*

Начиная с JDK 7 блок *try* дополнительно поддерживает:

1. Автоматизированный процесс освобождения ресурсов (например файлы).
2. **Групповой перехват** нескольких исключений одной инструкцией *catch*. Для этого неявно используется ключевое слово final и явно побитовый оператор ИЛИ.

*catch (final ArithmeticException | ArrayIndexOutOfBoundsException e) {*

*// блок кода }*

1. **Окончательное (или утонченное) повторное генерирование** **исключений**, которое ограничивает процесс лишь теми проверяемыми типами исключений, которые генерируются в блоке *try* и не обрабатываются в предыдущих блоках *catch*, а также относится к подтипу или супертипу указываемого параметра. Для этого используется ключевое слово ***final***.