**Определение исключений**

Исключения (exceptions) в Java представляют собой события, которые нарушают нормальный поток выполнения программы. Они возникают, когда программа сталкивается с ситуацией, которую она не может обработать обычными средствами. Например, попытка деления на ноль, обращение к несуществующему элементу массива или ошибка ввода-вывода могут вызвать исключения. Исключения позволяют программе справиться с этими непредвиденными ситуациями, предлагая механизмы для их перехвата и обработки.

#### Различие между ошибками (errors) и исключениями (exceptions)

* **Ошибки (Errors)**
  + **Определение**: Ошибки представляют собой серьезные проблемы, которые возникают на уровне среды выполнения и из-за которых выполнение программы не может продолжаться.
  + **Примеры**: OutOfMemoryError (недостаток памяти), StackOverflowError (переполнение стека).
  + **Обработка**: В большинстве случаев ошибки являются необрабатываемыми и указывают на серьезные проблемы, требующие вмешательства на уровне системы. Обычно программы не пытаются их перехватывать или обрабатывать, так как это проблемы, с которыми они не могут справиться.
* **Исключения (Exceptions)**
  + **Определение**: Исключения представляют собой условия, которые программа может обнаружить и с которыми может справиться во время выполнения.
  + **Примеры**: IOException (ошибка ввода-вывода), NullPointerException (попытка использования null-ссылки), ArithmeticException (арифметическая ошибка, например деление на ноль).
  + **Обработка**: Исключения можно перехватывать и обрабатывать, используя механизмы try-catch. Это позволяет программе продолжать выполнение или завершить его контролируемым образом.

Таким образом, исключения в Java предназначены для обработки ошибок, возникающих во время выполнения программы, в то время как ошибки представляют собой более серьезные проблемы, часто требующие вмешательства на уровне системы или среды выполнения.

### Иерархия исключений

В Java исключения организованы в виде иерархической структуры классов, где все они наследуются от базового класса Throwable.

#### 1. ****Throwable****

Throwable — это базовый класс для всех объектов, которые могут быть выброшены как исключения. Этот класс разделяется на две основные категории:

* **Error**
* **Exception**

#### 2. ****Error****

Error — это подтип Throwable, представляющий серьезные ошибки, которые приложения обычно не обрабатывают. Эти ошибки указывают на проблемы, происходящие на уровне виртуальной машины Java (JVM) или в самой системе, и они не предназначены для перехвата и обработки в программе.

* **Примеры ошибок**:
  + OutOfMemoryError — недостаток памяти.
  + StackOverflowError — переполнение стека.
  + VirtualMachineError — ошибки виртуальной машины.

#### 3. ****Exception****

Exception — это подтип Throwable, представляющий исключительные условия, которые могут быть перехвачены и обработаны программой. Все исключения, которые приложения могут ожидать и обрабатывать, наследуются от этого класса.

* **Exception** делится на два подтипа:
  + IOException — ошибки ввода-вывода.
  + ClassNotFoundException — ошибки загрузки класса.
  + ArrayIndexOutOfBoundsException — выход за пределы массива.

#### 4. ****Checked Exceptions****

Checked исключения — это исключения, которые проверяются во время компиляции. Программисты обязаны либо обрабатывать эти исключения, используя конструкции try-catch, либо декларировать их в сигнатуре метода с помощью ключевого слова throws.

* **Примеры checked исключений**:
  + IOException — ошибки ввода-вывода.
  + SQLException — ошибки взаимодействия с базами данных.
  + ClassNotFoundException — ошибки загрузки класса.

#### 5. ****Unchecked Exceptions (RuntimeException)****

Unchecked исключения — это исключения, которые не проверяются во время компиляции. Они наследуются от RuntimeException. Эти исключения указывают на логические ошибки в программе и, как правило, возникают из-за неправильного использования API.

* **Примеры unchecked исключений**:
  + NullPointerException — попытка использования null-ссылки.
  + ArrayIndexOutOfBoundsException — выход за пределы массива.
  + ArithmeticException — арифметическая ошибка, например, деление на ноль.

#### Конструкция try-catch

В Java конструкция try-catch используется для перехвата и обработки исключений, возникающих во время выполнения программы. Основная цель этой конструкции — предотвратить аварийное завершение программы и обеспечить контролируемую обработку ошибок.

#### Несколько блоков catch

Java позволяет использовать несколько блоков catch для обработки различных типов исключений, которые могут возникнуть в блоке try. Каждый блок catch обрабатывает определенный тип исключения.

### Блок finally

#### Значение блока finally

Блок finally в Java является опциональной частью конструкции try-catch. Он используется для выполнения кода, который должен быть выполнен в любом случае, независимо от того, было выброшено исключение или нет. Основное применение блока finally — это освобождение ресурсов, таких как закрытие файлов, освобождение памяти или завершение сетевых соединений, чтобы избежать утечек ресурсов.