При обработке исключений в Java применяются следующие **ключевые слова**: — **try** – служит для определения блока кода, в котором может произойти исключение;

— **catch** – необходим для определения блока кода, где происходит **обработка исключения**;

— **finally** – применяется для определения блока кода, являющегося необязательным, однако при его наличии он выполняется в любом случае вне зависимости от того произошло ли исключение в блоке try.

Кроме того:

1. Для возбуждения исключения используем **throw**.

2. Для предупреждения в сигнатуре методов о том, что метод может выбросить исключение, применяем **throws**.

**Try with resourse**

Данная конструкция, которая появилась в Java 7, позволяет использовать блок *try-catch* не заботясь о закрытии ресурсов, используемых в данном сегменте кода. Ресурсы объявляются в скобках сразу после try, а компилятор уже сам неявно создаёт секцию finally, в которой и происходит освобождение занятых в блоке ресурсов. Под ресурсами подразумеваются сущности, реализующие интерфейс java.lang.Autocloseable. Он имеет метод close, который освободит ресурсы не зависимо от того, были ли исключения.

Общий вид конструкции:

try(/\*объявление ресурсов\*/) {

//...

} catch(Exception ex) {

//...

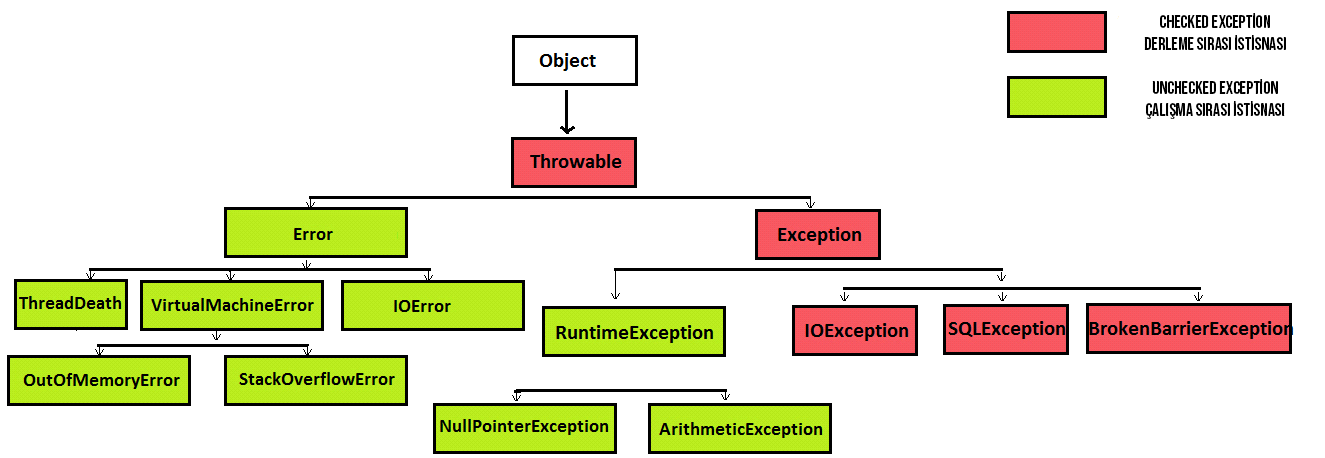
} finally {

//...

}

Стоит заметить, что блоки catch и явный finally выполняются уже после того, как закрываются ресурсы в неявном finally.

**Иерархия исключений**



Исключительные ситуации, которые возникают в программе, делят на 2 группы:

1. Ситуации, при которых восстановление нормальной дальнейшей работы невозможно.

2. Ситуации с возможностью восстановления.

К первой группе можно отнести случаи, при которых возникают исключения, которые унаследованы из класса **Error**. Это ошибки, возникающие во время выполнения программы при сбое работы **Java Virtual Machine**, переполнении памяти либо сбое системы, а также **RuntimeException**.

Это **unchecked** исключения. Остальные классы исключений **checked,** и их можно обработать в блоке try/catch.