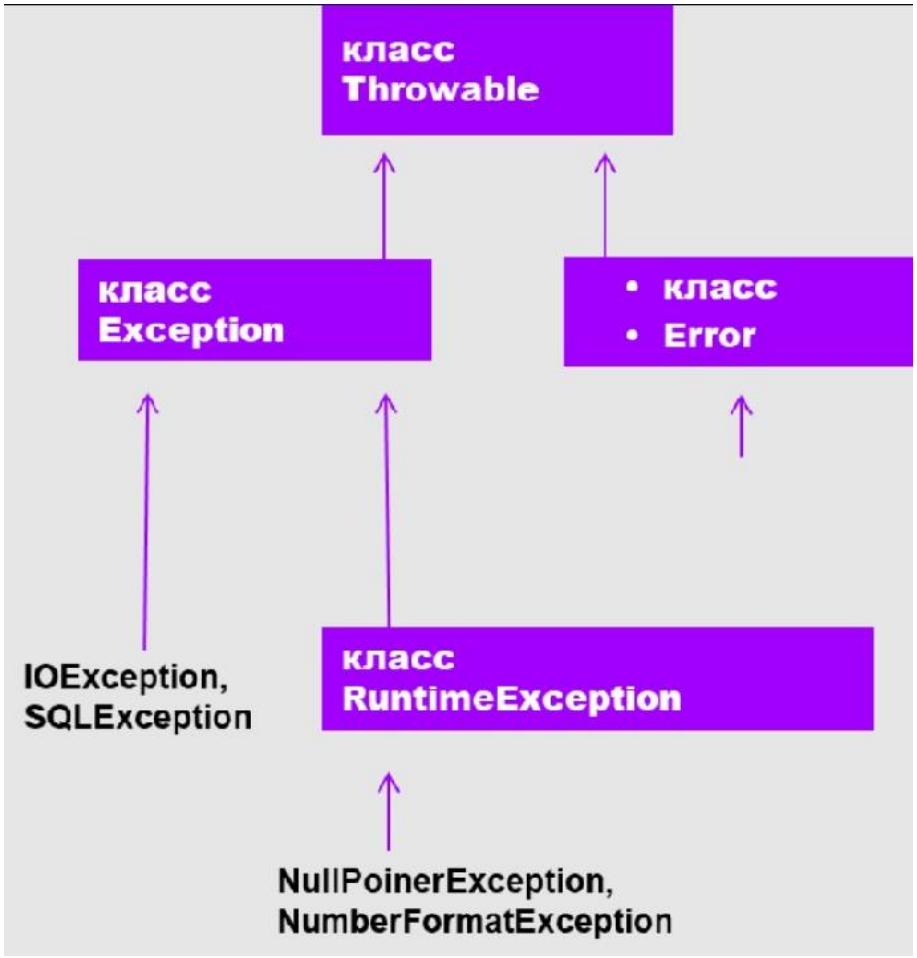
Исключения

Исключения


класс
Throwable
класс
Exception
• класс
• Error
IOException,
SQLException
класс
RuntimeException
NullPoinerException,
NumberFormatException

Исключения

Исключения делятся на:

- Checked (проверяемые) - ошибки, которые возникают из-за внешних по отношению к программе факторов, например IOException – ошибка чтения/записи являются наследниками Exception (но не RuntimeException). Требуют программной обработки.
- Unchecked (непроверяемые) - программные ошибки, например, NullPointerException. Обработка не требуется. Являются наследниками RuntimeException.
- Errors (ошибки) - системные ошибки, которые приложение не в состоянии обработать, например OutOfMemmoryError не обрабатываются. Являются наследниками Error.

Инициация исключений и обработка исключений

THROW – выбросить исключение.
Исключение может быть как checked, так и unchecked.

Например:

- throw new RuntimeException(“Число должно быть больше 0”);
- throw new FileNotFoundException(“Файл не найден”);

Обработка исключений

THROWS – пометить метод, как выбрасывающий checked исключение

public void myReadFile(String filePath) throws IOException { ... }

Обработка исключений

public class FileReader {

public void readFile(String filePath) throws FileNotFoundException {

if(<file not exists>) {

throw new FileNotFoundException(); } ... } }

public class Main {

public static void main(String[] args) throws FileNotFoundException {

FileReader fr = new FileReader();

// мы не хотим обрабатывать исключение здесь

// поэтому пробрасываем исключение дальше через throws fr.readFile(“config.xml”) } }

Обработка исключений. TRY-CATCH-FINALLY

```
try {
 //вызов метода, который может вызвать исключение;
 } catch(FileNotFoundException e) {
// обработка исключения FileNotFoundException
} catch(Exception e) {
// обработка всех остальных исключений
}
finally {
 // код, который обязательно выполнится вне зависимости от наличия исключения
}
```

Если обрабатываемые исключения находятся в иерархии, первым обрабатывается исключение-наследник.

Обработка исключений. TRY-CATCH

```
public class Main {
public static void main(String[] args) {
FileReader fr = new FileReader();
try {
fr.readFile(“config.xml”);
 } catch (FileNotFoundException|IOException e) {
 e.printStackTrace(); }
}
}
```

Создание исключения

Собственные исключения наследуют либо Exception (checked), либо RuntimeException.

```
public class ServiceException extends RuntimeException {
 public ServiceException(String message) {
super(message);
 }
}
```

Перехват checked исключения и выброс своего unchecked исключения

```
public static void main(String[] args) {
try (FileReader fr = new FileReader("")){
 // do something here }
 catch (FileNotFoundException e) {
 throw new ServiceException("Файл не найден");
 }
 catch (IOException e) { throw new
ServiceException("Ошибка работы с файлом"); } }
```