A light blue outline of a world map is centered on a solid blue background. The map shows the continents of North America, South America, Europe, Africa, Asia, and Australia.

Исключительные ситуации (Exceptions)

- ◆ Введение
- ◆ Иерархия
- ◆ Причины возникновения ошибок
- ◆ Обработка исключительных ситуаций
- ◆ Проверяемые и непроверяемые исключения
- ◆ Создание пользовательских классов исключений
- ◆ Переопределение методов и исключения

Классические языки	Java
<pre>... int statusCode = someAction(); if (statusCode){ ... обработка ошибки } else { statusCode = anotherAction(); if(statusCode) { ... обработка ошибки ... } } ...</pre>	<pre>try{ someAction(); anotherAction(); } catch(Exception e) { // обработка исключительной ситуации }</pre>

- ◆ Попытка выполнить некорректное выражение.
- ◆ Выполнение оператора `throw` Этот оператор применяется для явного порождения ошибки.
- ◆ Асинхронные ошибки во время исполнения программы (например, *OutOfMemoryException*)

Конструкция try-catch-finally:

```
try {  
    ...  
} catch(SomeExceptionClass e) {  
    ...  
} catch(AnotherExceptionClass e) {  
    ...  
} finally {  
    ...  
}
```

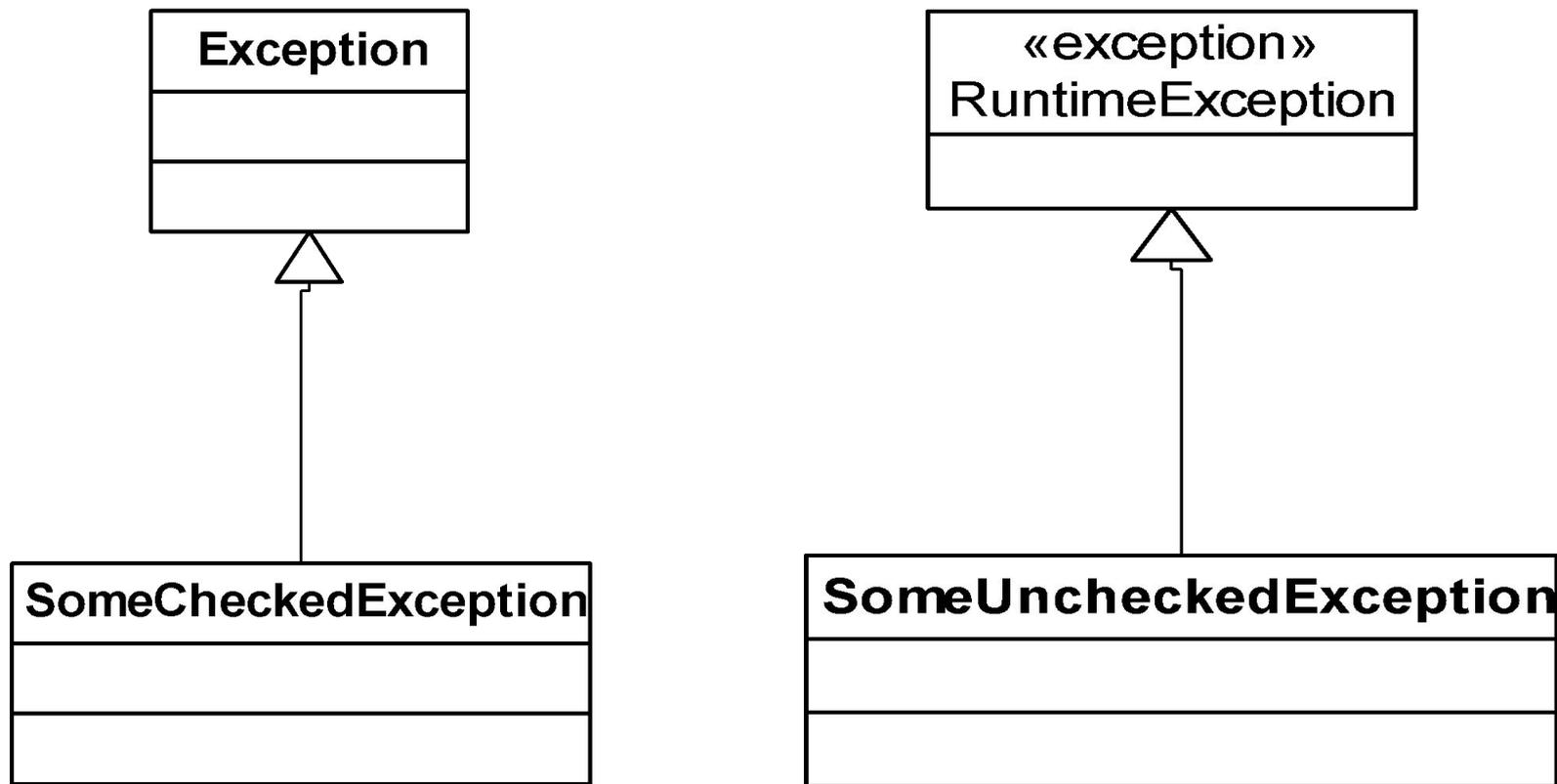
Конструкция try-catch-finally:

```
try {  
    byte [] buffer = new byte[128];  
    FileInputStream fis =  
        new FileInputStream("file.txt");  
    while(fis.read(buffer) > 0) {  
        ... обработка данных ...  
    }  
} catch(IOException es) {  
    ... обработка исключения ...  
} finally {  
    fis.flush();  
    fis.close();  
}
```

```
public int calculate(int theValue) throws Exception {  
    if( theValue < 0) {  
        throw new Exception("Параметр для  
вычисления не должен быть отрицательным");  
    }  
}
```

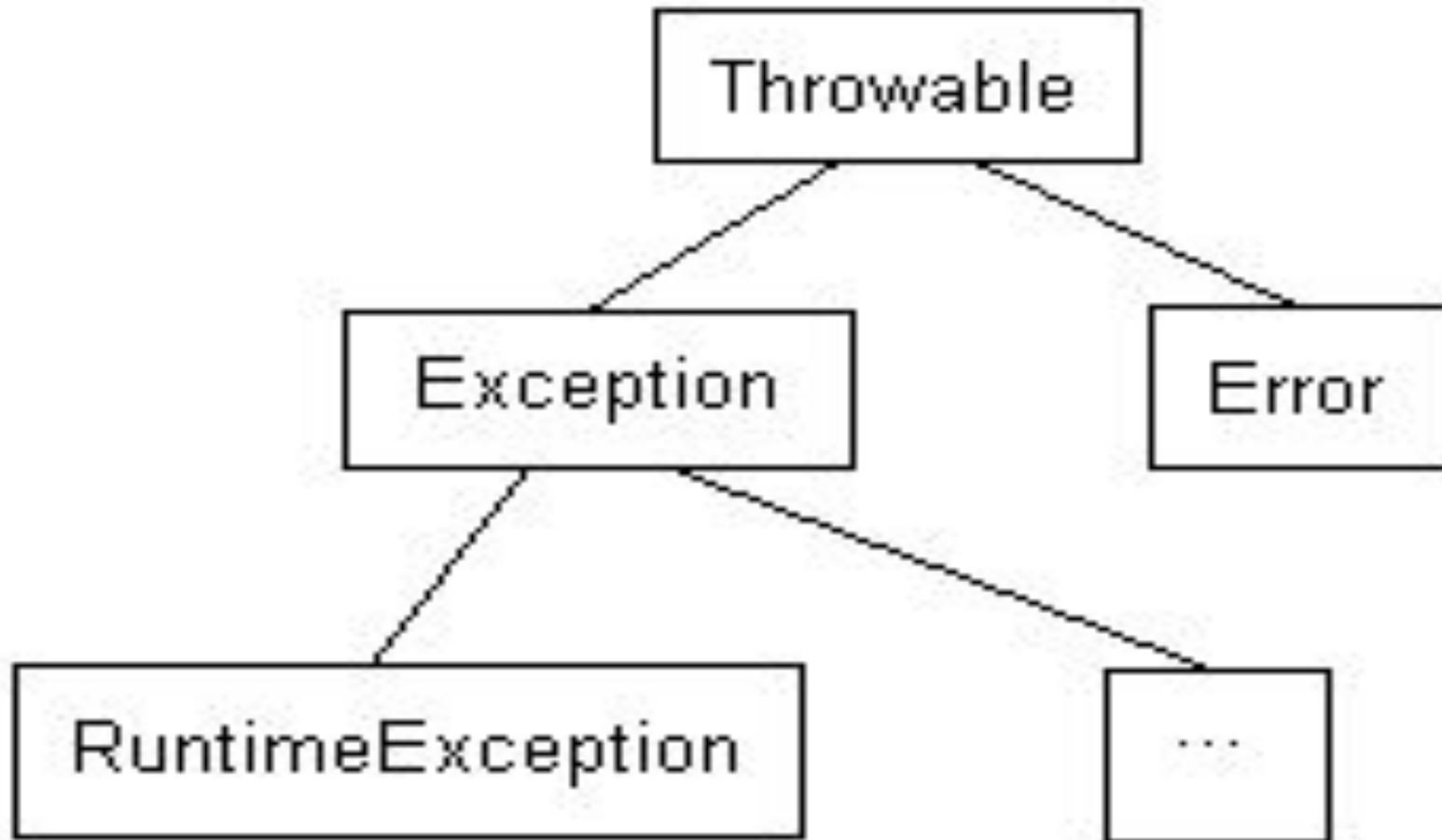
```
try {  
    ...  
} catch(IOException ex) {  
    // Обработка исключительной ситуации  
    // Повторное возбуждение исключительной  
    // ситуации  
    throw ex;  
}
```

```
try {  
    ...  
    throw new IOException();  
    ...  
} catch(Exception e) {  
    ...  
}
```



ClassNotFoundException
CloneNotSupportedException
...

NullPointerException,
ArrayIndexOutOfBoundsException
...



Неправильная конструкция	Правильная конструкция
<pre>try { ... } catch(Exception e) { ... } catch(IOException ioe) { ... } catch(UserException ue) { ... }</pre>	<pre>try { ... } catch(UserException ue) { ... } catch(IOException ioe) { ... } catch(Exception e) { ... }</pre>

```
public class UserException extends Exception {  
    public UserException() {  
        super();  
    }  
    public UserException(String description) {  
        super(description);  
    }  
}
```

Использование:

```
throw new UserException("Дополнительное описание");
```

При переопределении методов следует помнить, что если переопределяемый метод объявляет список возможных исключений, то переопределяющий метод не может расширять этот список, но может его сужать.

Метод базового класса	Корректно	Некорректно
<code>public void makeIt() throws SomeException</code>	<code>public void makeIt()</code>	<code>public void makeIt() throws SomeException. SomeMoreException</code>

Вопросы ?