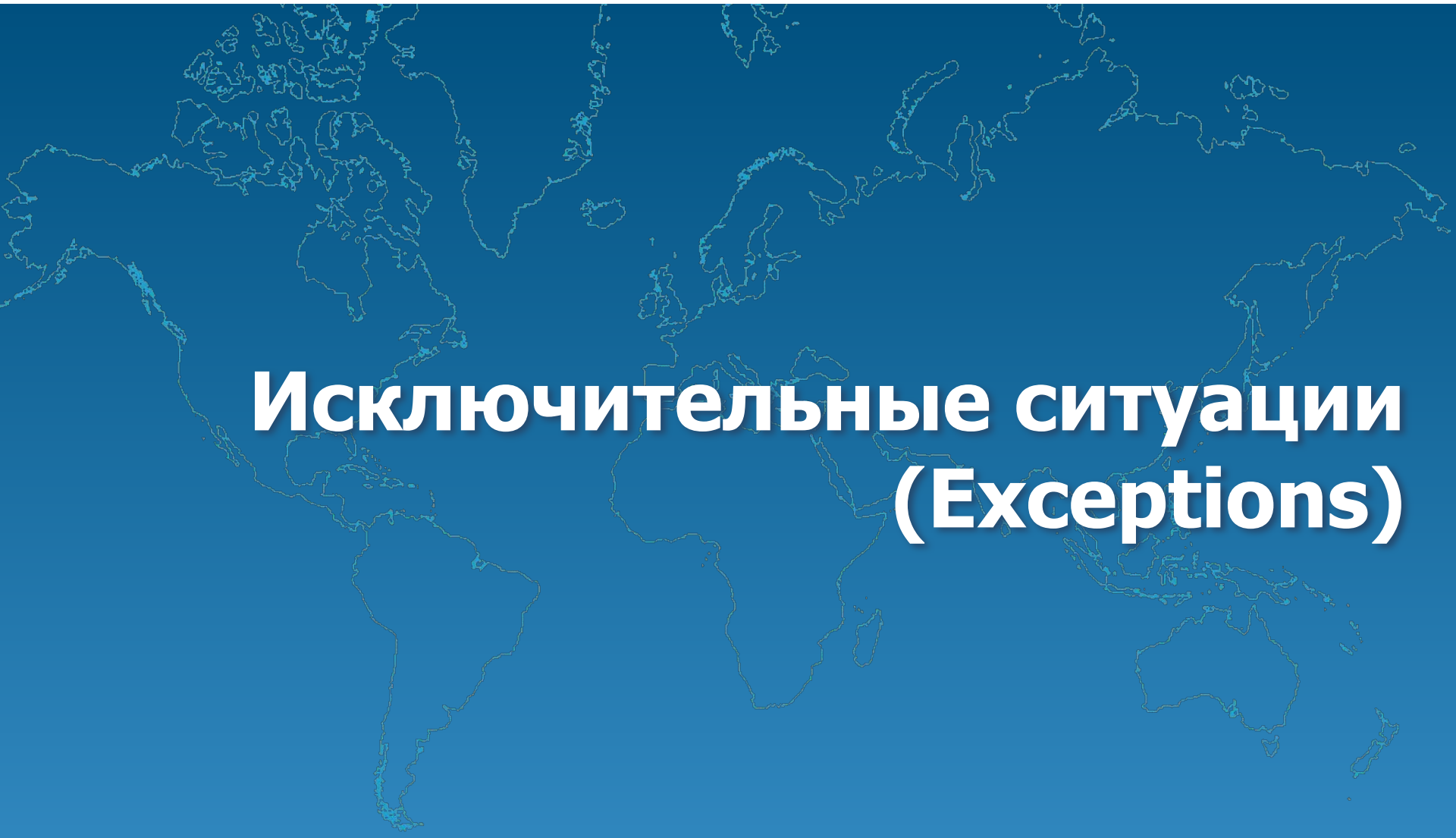
NetCracker®
Transforming the Service Layer™



Исключительные ситуации (Exceptions)

NetCracker®
Transforming the Service Layer™

Содержание

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

Введение

Классические языки	Java

...
int statusCode = someAction();
if (statusCode){
... обработка ошибки
} else {
statusCode = anotherAction();
if(statusCode) {
... обработка ошибки ...
}
}
...
	try{
	someAction();
	anotherAction();
	} catch(Exception e) {
	// обработка исключительной ситуации
	}

NetCracker®
Transforming the Service Layer™

Причины возникновения ошибок

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

Обработка исключительных ситуаций

Конструкция try-catch-finally:

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

NetCracker®
Transforming the Service Layer™

Обработка исключительных ситуаций

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();
}
```

NetCracker®
Transforming the Service Layer™

Использование оператора throw

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

NetCracker®
Transforming the Service Layer™

Использование оператора throw

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

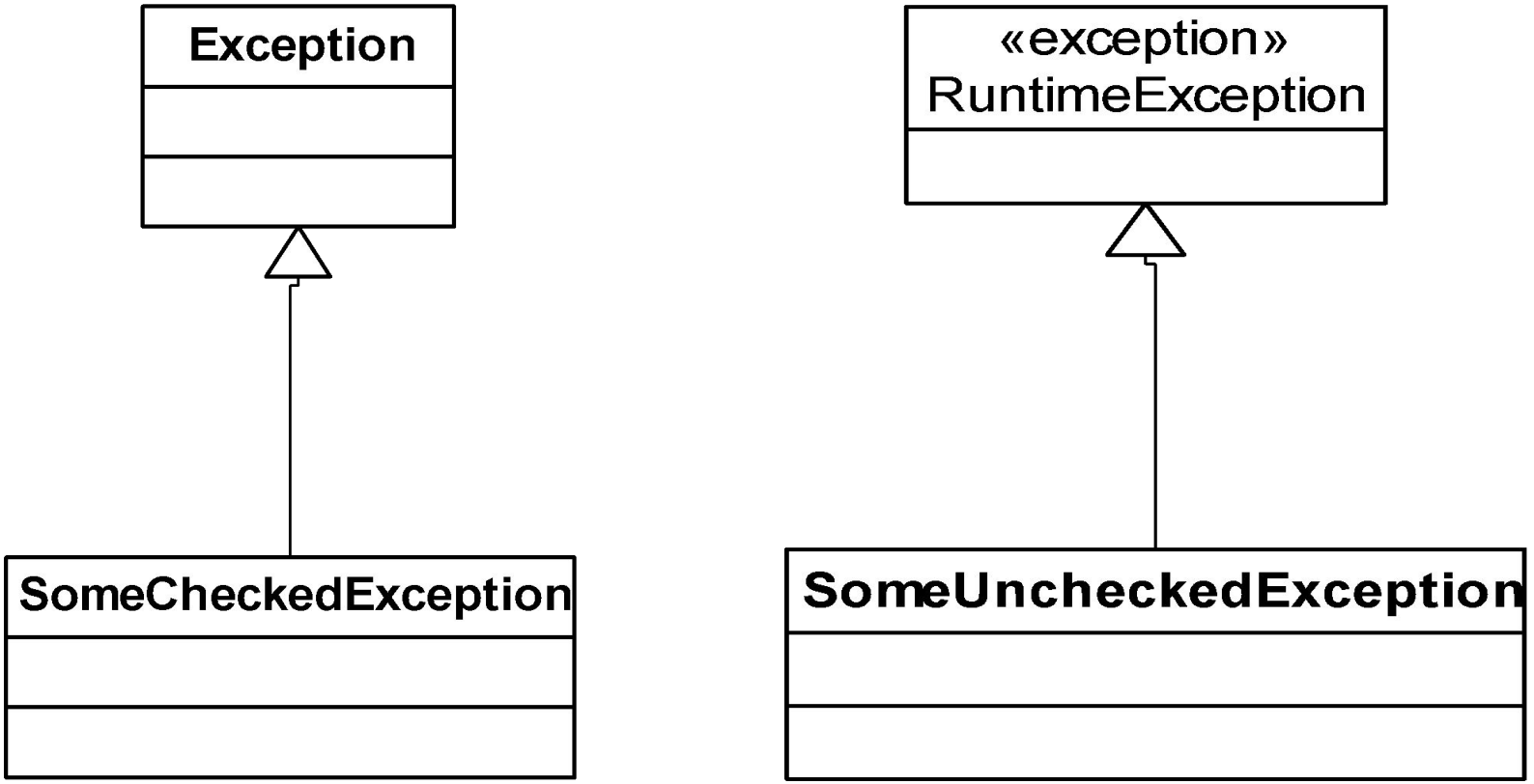
NetCracker®
Transforming the Service Layer™

Использование оператора throw

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

NetCracker®
Transforming the Service Layer™

Проверяемые и непроверяемые исключения

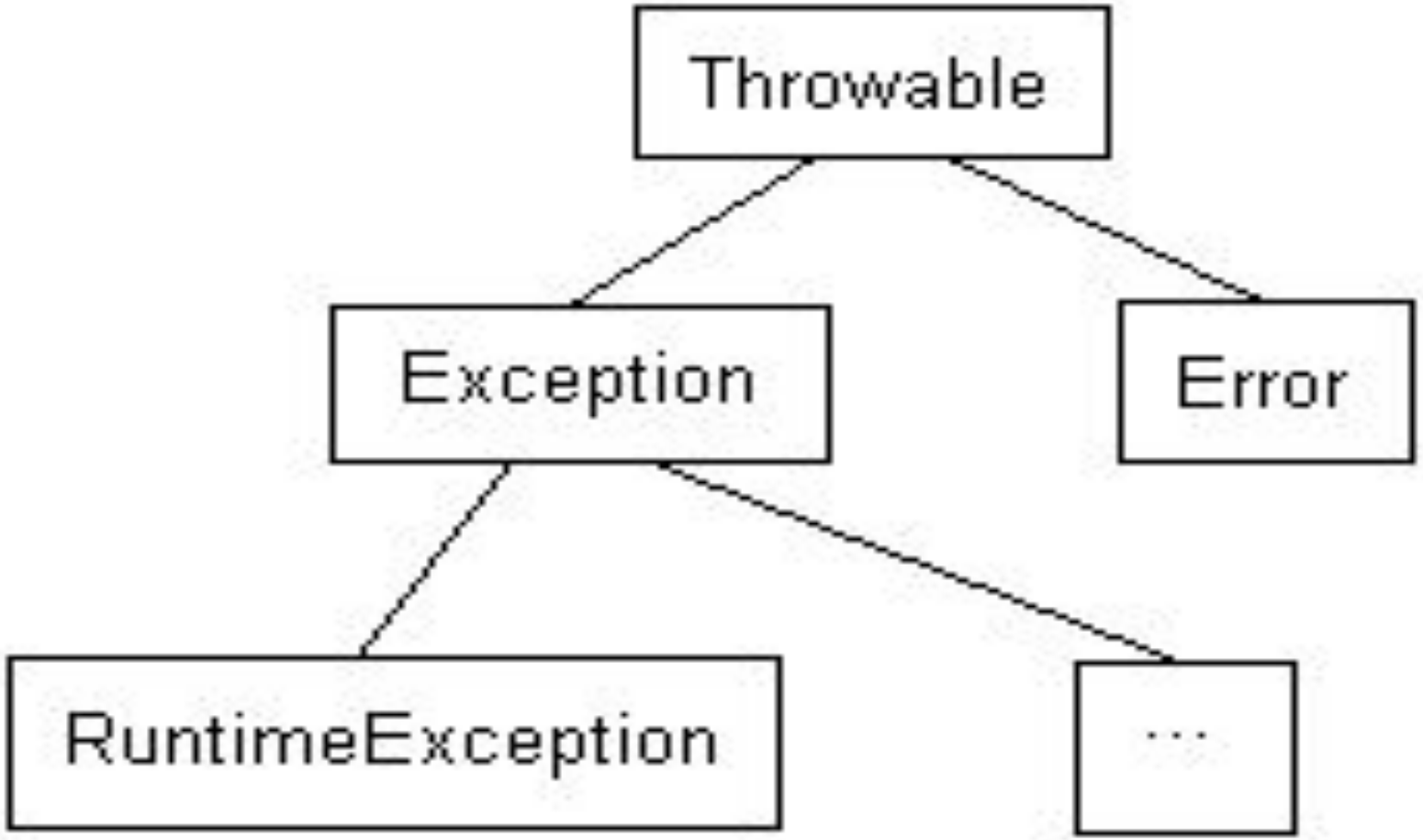


ClassNotFoundException
CloneNotSupportedException
…

NullPointerException,
ArrayIndexOutOfBoundsException
…

NetCracker®
Transforming the Service Layer™

Иерархия



NetCracker®
Transforming the Service Layer™

Последовательность обработки исключений

Неправильная конструкция	Правильная конструкция

try {
  …
} catch(Exception e) {
  …
} catch(IOException ioe) {
  …
} catch(UserException ue) {
  …
}
	try {
	  …
	} catch(UserException ue) {
	  …
	} catch(IOException ioe) {
	  …
	} catch(Exception e) {
	…
	}

NetCracker®
Transforming the Service Layer™

Пользовательские классы исключений

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

Использование:

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

NetCracker®
Transforming the Service Layer™

Переопределение методов и исключения

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
public void makeIt() throws SomeException	public void makeIt()	public void makeIt() throws SomeException. SomeMoreException

NetCracker®
Transforming the Service Layer™

Исключительные ситуации (Exceptions)

Вопросы ?