Exception handling

Problem

Problem

```
int[] numbers = new int[3];
numbers[4] = 45;
System.out.println(numbers[4]);
```

Problem

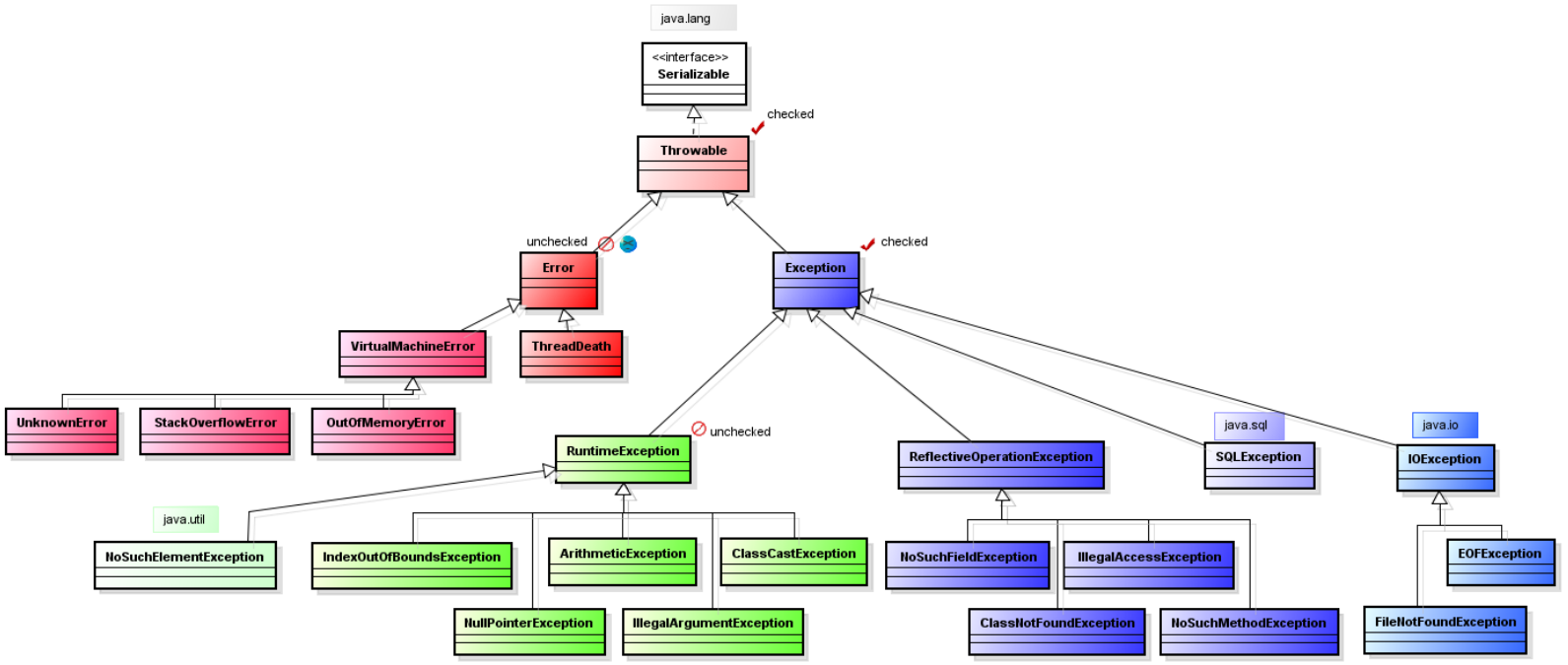
Что делать если произошла нештатная ситуация, ошибка во время выполнения программы?

Например:

- Пользователь ввел некоректные данные
- Не смогли подключиться к стороннему сервису
- Не найден файл с которым выполняются какие-то действия
- Не выполняются какие-то бизнес требования

Throwable

Hierarchy


java.lang
<<interface>>
Serializable
checked
Throwable
unchecked
Error
checked
Exception
VirtualMachineError
ThreadDeath
UnknownError
StackOverflowError
OutOfMemoryError
unchecked
RuntimeException
java.sql
java.io
ReflectiveOperationException
SQLException
IOException
java.util
NoSuchElementException
IndexOutOfBoundsException
ArithmeticException
ClassCastException
NullPointerException
IllegalArgumentException
NoSuchFieldException
IllegalAccessException
ClassNotFoundException
NoSuchMethodException
EOFException
FileNotFoundException

Throwable

- Error
- Exception

Error

- представляют собой более серьёзные проблемы
- согласно спецификации Java, не следует пытаться обрабатывать в собственной программе
- связаны с проблемами уровня JVM

Subclasses `Error`

- `StackOverflowError`
- `OutOfMemoryError`
- `UnknownError`
- etc.

`Exception`

- являются результатом проблем в программе
- в принципе решаемы и предсказуемы

`Exception`

- `RuntimeException` (unchecked exceptions)
- Another Exception (checked exceptions)

Types

- **unchecked** (**неконтролируемые**)
  - `RuntimeException` и его наследники
  - `Error` и его наследники
- **checked** (**контролируемые**)
  - все остальные

RuntimeException

- `NullPointerException` - неверное использование пустой ссылки
- `ClassCastException` - неверное приведение
- `ArrayIndexOutOfBoundsException` - выход индекса за границу массива
- `NumberFormatException` - неверное преобразование строки в числовой формат
- `ArithmeticException` - арифметическая ошибка, например, деление на нуль
- `ArrayStoreException` - присваивание элементу массива объекта несовместимого типа

Another Exceptions

- `ClassNotFoundException` - класс не найден
- `CloneNotSupportedException` - попытка клонировать объект, который не реализует интерфейс `Cloneable`
- `IllegalAccessException` - запрещен доступ к классу
- `InstantiationException` - попытка создать объект абстрактного класса или интерфейса
- `InterruptedException` - поток прерван другим потоком
- `NoSuchFieldException` - запрашиваемое поле не существует
- `NoSuchMethodException` - запрашиваемый метод не существует

Exception handling

Exception handling

```
int[] numbers = new int[3];
numbers[4] = 45;
System.out.println(numbers[4]);
```

try…catch

```
try {
    int[] numbers = new int[3];
    numbers[4] = 45;
    System.out.println(numbers[4]);
} catch (Exception e) {
    // error processing
    // usually logging
}
System.out.println("Программа завершена");
```

try…catch…finally

```
try {
    int[] numbers = new int[3];
    numbers[4] = 45;
    System.out.println(numbers[4]);
} catch (Exception e) {
    // exception processing
    // usually logging
} finally {
    // mandatory actions AFTER exception processing
}
System.out.println("Программа завершена");
```

Multiexceptions `catch`

Multiexceptions `catch`

```
int[] numbers = new int[3];
try {
    numbers[6] = 45;
    numbers[6] = Integer.parseInt("gfd");
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Выход за пределы массива");
} catch (NumberFormatException e) {
    System.out.println("Ошибка преобразования из строки в числ
}
```

Multiexceptions `catch`

```
int[] numbers = new int[3];
try {
    numbers[6] = 45;
    numbers[6] = Integer.parseInt("gfd");
} catch (Exception e) {
    System.out.println("Какой-то Exception"); // WRONG!!!
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Выход за пределы массива");
} catch (NumberFormatException e) {
    System.out.println("Ошибка преобразования из строки в числ
}
```

Multiexceptions `catch`

```
int[] numbers = new int[3];
try {
    numbers[6] = 45;
    numbers[6] = Integer.parseInt("gfd");
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Ошибка при обработке массива чисел");
} catch (NumberFormatException e) {
    System.out.println("Ошибка при обработке массива чисел");
}
```

Multiexceptions `catch`

```
int[] numbers = new int[3];
try {
    numbers[6] = 45;
    numbers[6] = Integer.parseInt("gfd");
} catch (ArrayIndexOutOfBoundsException | NumberFormatExceptior
    System.out.println("Ошибка при обработке массива чисел");
}
```

Operators for Exception handling

Operator `throw`

```
import java.util.Scanner;

public class FirstApp {
    public static void main(String[] args) {
        try {
            Scanner in = new Scanner(System.in);
            int x = in.nextInt();
            if (x >= 30) {
                throw new Exception("Число x должно быть меньше
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        System.out.println("Программа завершена");
    }
}
```

Operator `throws`

```
public static int getFactorial(int num) throws Exception {
    if (num < 1) {
        throw new Exception("The number is less than 1");
    }
    int result = 1;
    for (int i = 1; i <= num; i++) {
        result *= i;
    }
    return result;
}
```

Operator `throws`

```
public static void main(String[] args) {
    int result = getFactorial(-6); // compile error
    System.out.println(result);
}
```

Operator `throws`

```
public static void main(String[] args) {
    try {
        int result = getFactorial(-6);
        System.out.println(result);
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
```

How do without `throws`

```
public static int getFactorial(int num) {
    int result = 1;
    try {
        if (num < 1) {
            throw new Exception("The number is less than 1");
        }
        for (int i = 1; i <= num; i++) {
            result *= i;
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
        result = num;
    }
    return result;
}
```

Custom Exception

Custom Exception

```
class FactorialException extends Exception {
    private int number;

    public int getNumber() {
        return number;
    }

    public FactorialException(String message, int num) {
        super(message);
        number = num;
    }
}
```

Custom Exception

```
class Factorial {
    public static int getFactorial(int num) throws FactorialExc
        int result = 1;
        if (num < 1) {
            throw new FactorialException("The number is less th
        }
        for (int i = 1; i <= num; i++) {
            result *= i;
        }
        return result;
    }
}
```