

Obiektowe Języki Programowania

Wyjątki

Łukasz Rybka · Gdańsk 2015

“ Podstawa ideologii Javy jest założenie, że **"źle sformułowany kod nie zostanie wykonany"**.

— Bruce Eckel

“ Aby system był niezawodny,
każdy jego komponent musi być
niezawodny.

— Bruce Eckel

Proces zgłaszania wyjątku

Proces zgłaszania wyjątku

Tworzony jest obiekt wyjątku

Proces zgłaszania wyjątku

Tworzony jest obiekt wyjątku

Aktualna ścieżka wykonania jest przerywana
i "rzucany" jest obiekt wyjątku

Proces zgłaszania wyjątku

Tworzony jest obiekt wyjątku

Aktualna ścieżka wykonania jest przerywana
i "rzucany" jest obiekt wyjątku

Mechanizm obsługi wyjątków wyszukuje tzw.
procedurę obsługi wyjątku - kodu
obsługującego wyjątkową sytuację

Blok try...catch

```
public class Main {  
    public static void main(String[] args) {  
        try {  
            // ...  
        } catch (ExceptionType1 name) {  
            // ...  
        } catch (ExceptionType2 name) {  
            // ...  
        }  
    }  
}
```


Blok try...catch

Blok try...catch

Każdy blok catch działa jak metoda z jednym parametrem - obiektem wyjątku danego typu

Blok try...catch

Każdy blok catch działa jak metoda z jednym parametrem - obiektem wyjątku danego typu
Bloków catch może być dowolna liczba

Blok try...catch

Każdy blok catch działa jak metoda z jednym parametrem - obiektem wyjątku danego typu
Bloków catch może być dowolna liczba
Tylko pierwszy pasujący block catch jest wykonywany

Blok try...catch

```
public class Main {  
    public static void main(String[] args) {  
        try {  
            someMethod();  
        } catch (IndexOutOfBoundsException e) {  
            System.err.println("IndexOutOfBoundsException");  
        } catch (SQLException e) {  
            System.err.println("Caught SQLException");  
        } catch (IOException e) {  
            System.err.println("Caught IOException");  
        }  
    }  
}
```

Blok try...catch w Java SE 7+

```
public class Main {  
    public static void main(String[] args) {  
        try {  
            someMethod();  
        } catch (IndexOutOfBoundsException e) {  
            System.err.println("IndexOutOfBoundsException");  
        } catch (IOException | SQLException e) {  
            System.err.println("Caught SQLException or IOException");  
        }  
    }  
}
```

Specyfikacja wyjątków

```
public class Main {  
    void someMethod() throws IndexOutOfBoundsException,  
        IOException, SQLException {  
  
        // Some method code...  
    }  
}
```

Specyfikacja wyjątków

Specyfikacja wyjątków

Definiowana w sygnaturze metody

Specyfikacja wyjątków

Definiowana w sygnaturze metody

Wymaga podania wszystkich wyjątków typu

Checked nie obsługiwanych w ciele metody

Specyfikacja wyjątków

Definiowana w sygnaturze metody

Wymaga podania wszystkich wyjątków typu

Checked nie obsługiwanych w ciele metody

Wyjątki Unchecked (RuntimeException) nie muszą być podawane w sygnaturze metody

Specyfikacja wyjątków

Definiowana w sygnaturze metody

Wymaga podania wszystkich wyjątków typu

Checked nie obsługiwanych w ciele metody

Wyjątki Unchecked (RuntimeException) nie muszą być podawane w sygnaturze metody

Wymusza na kliencie obsługę wyjątku lub przekazania informacji w sygnaturze metody

Ponowne rzucenie wyjątku

```
public class Main {  
    public static void main(String[] args) {  
        try {  
            someMethod();  
        } catch (Exception e) {  
            System.out.println(e.printStackTrace());  
            e.printStackTrace(System.out);  
  
            throw e;  
        }  
    }  
}
```

Rzucenie wyjątku

```
public class Main {  
    void someMethod(int index) throws IndexOutOfBoundsException {  
        // ...  
  
        throw new IndexOutOfBoundsException("Invalid index!");  
    }  
}
```

Sekwencja wyjątków

```
public class Main {  
    void someMethod(Integer index) throws RuntimeException {  
        IndexOutOfBoundsException cause = new  
IndexOutOfBoundsException("Invalid index!");  
        cause.initCause(new NullPointerException("Argument is  
null!"));  
        throw new RuntimeException(cause);  
    }  
}
```

Tworzenie własnych wyjątków

```
class VeryImportantException extends Exception {  
}  
  
class ExtendedMessageException extends Exception {  
    @Override  
    public String getMessage() {  
        return "ExtendedMessageException!!!!\n" + super.getMessage();  
    }  
  
    @Override  
    public String toString() {  
        return "ExtendedMessageException{}";  
    }  
}
```


Blok try...catch...finally

```
public class Main {  
    public static void main(String[] args) {  
        try {  
            // ...  
        } catch (ExceptionType1 name) {  
            // ...  
        } catch (ExceptionType2 name) {  
            // ...  
        } finally {  
            // ...  
        }  
    }  
}
```

Bloku finally: zagubiony wyjątek

```
class InitException extends Exception {}

class DisposeException extends Exception {}

public class Main {
    void init() throws InitException { }

    void dispose() throws DisposeException { }

    public static void main(String[] args) {
        try {
            init();
        } finally {
            dispose();
        }
    }
}
```

Bloku finally: zagubiony wyjątek

```
class InitException extends Exception {}

class DisposeException extends Exception {}

public class Main {
    void init() throws InitException { }

    void dispose() throws DisposeException { }

    public static void main(String[] args) {
        try {
            init();
        } finally {
            try {
                dispose();
            } catch (DisposeException e) {
                // ...
            }
        }
    }
}
```

Bloku finally: tłumienie wyjątków

```
public class Main {  
    public static void main(String[] args) {  
        try {  
            throw new RuntimeException();  
        } finally {  
            return;  
        }  
    }  
}
```

ANY
QUESTIONS
?

Pytania?