

PROGRAMOWANIE OBIEKTOWE



Wyjątki

dr inż. Barbara Fryc

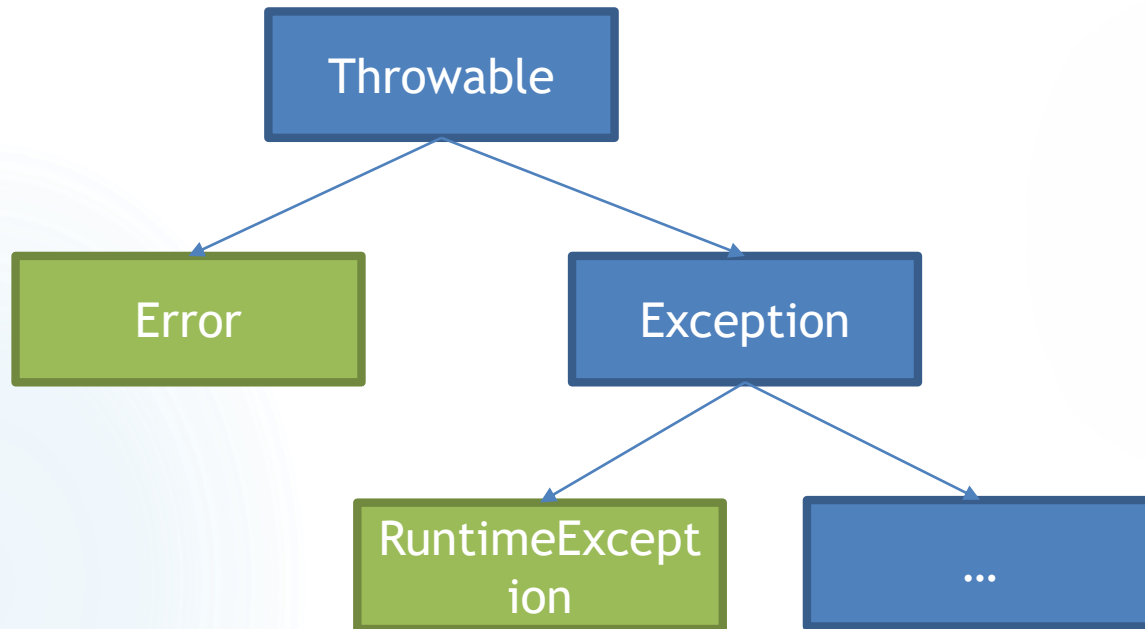
OBSŁUGA SYTUACJI WYJĄTKOWYCH

Wyjątek - klasa reprezentująca określoną sytuację wyjątkową.


Przykłady sytuacji wyjątkowych:

- ▶ Brak miejsca na dysku
- ▶ Dzielenie przez zero
- ▶ Błąd komunikacji sieciowej
- ▶ Błąd odczytu pliku
- ▶ Błąd konwertowania XML na obiekt

WYJĄTKI - CO TO JEST?



 Wyjątki niesprawdzane (unchecked)

 Wyjątki sprawdzane (checked)

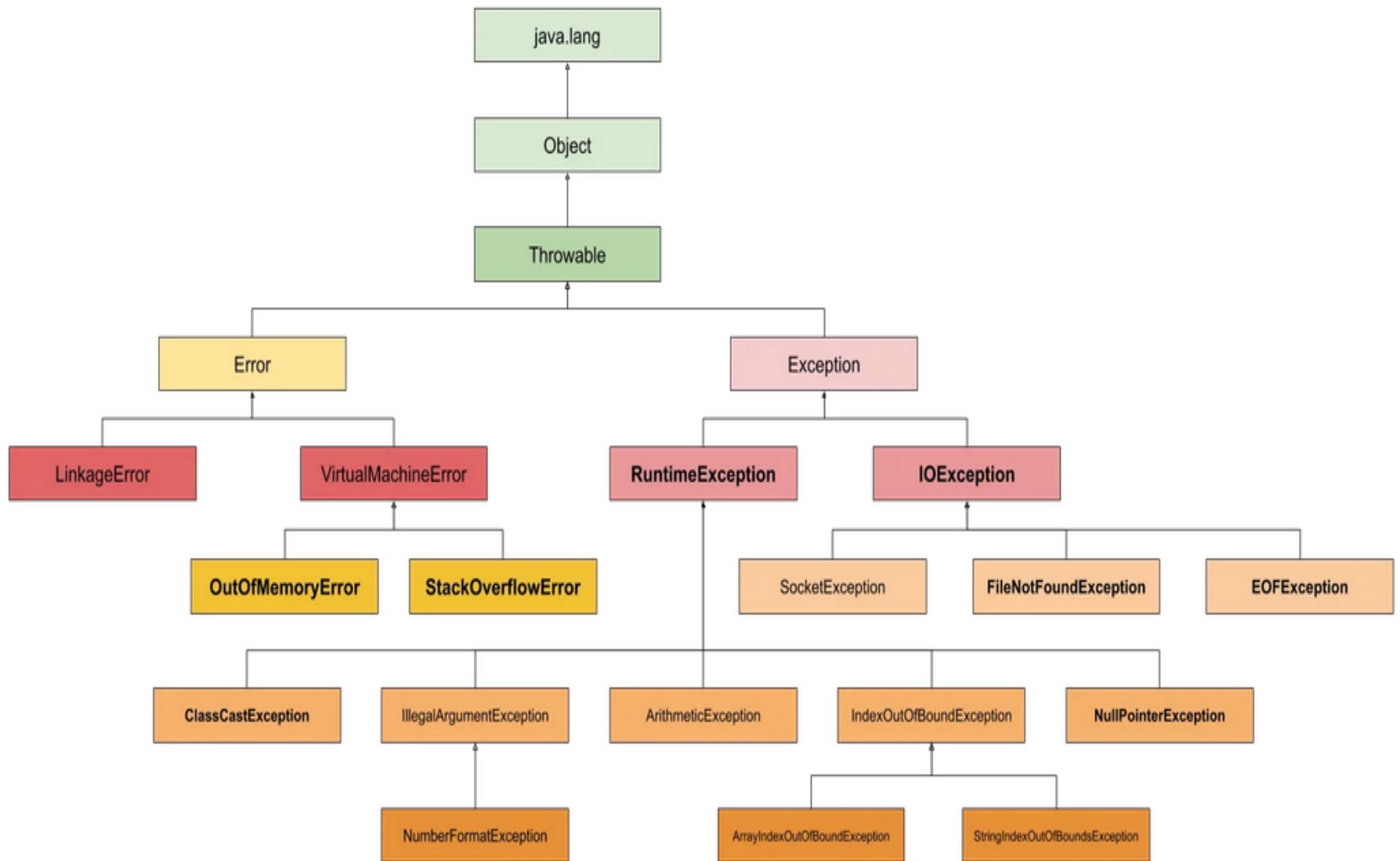
WYJĄTKI SPRAWDZANE I NIESPRAWDZANE

Wyjątki niesprawdzone (unchecked)

- Pochodne klas Error i RuntimeException
- Nie muszą być wyłapywane
- Prowadzą do przerwania działania programu (wątku)

Wyjątki sprawdzane (checked)

- Pochodne pozostałych klas Exception
- Muszą być wyłapywane
- Są obsługiwane w programie



STACKTRACE CZYLI ANATOMIA BŁĘDU

Każdy wyjątek zawiera informacje o miejscu i czasie kiedy powstał.
Zawiera też informacje o tzw. stosie wywołań swojego miejsca powstania.

```
C:\java\jdk1.8.0_201\bin\java.exe ...
```

```
pl.wsiz.lectures.MojSpecjalnyWyjątek
```

```
    at pl.wsiz.lectures.TestException.metodaPierwsza(TestException.java:11)
```

```
    at pl.wsiz.lectures.TestException.metodaDwa(TestException.java:15)
```

```
    at pl.wsiz.lectures.TestException.main(TestException.java:21)
```

```
Process finished with exit code 0
```



OBSŁUGA WYJĄTKÓW

```
try {  
    // kod który może spowodować  
    // wyjątki  
}  
catch (KlasaWyjatkul e) {  
    //obsługa wyjątku  
}  
catch (KlasaWyjatkul2 e) {  
    //obsługa wyjątku  
}
```



BLOK FINALLY

```
try {  
    kod który może spowodować  
    wyjątki  
}  
catch (Wyjątek e) {  
    obsługa wyjątku  
}  
finally {  
    kod, który zawsze musi się  
    wykonać  
}
```

Blok **finally** służy do wprowadzenia kodu który zawsze musi się wykonać.

OBSŁUGA WYJĄTKÓW CD.

```
int funkcja(...) throws KlasaWyjatkul,  
KlasaWyjatkul2 {  
    operacje, które mogą spowodować  
    wyjątki  
}
```

Gdy w jakiejś funkcji nie możemy obsłużyć wyjątku, do jej nagłówka dodajemy deklarację jakie wyjątki może ona **rzucić**.

OBSŁUGA WYJĄTKÓW CD.

Do „rzucania” nowego wyjątków służy słowo kluczowe **throw**:

...

```
throw new KlasaWyjatk(...)
```

...

Typy „rzucanych” wyjątków muszą być pochodnymi klas **Throwable** (najczęściej pochodnymi **Exception**).



RZUCANIE WYJĄTKÓW CD.

Aby powtórnie rzucić raz złapany wyjątek również używamy słowa kluczowego **throw**.

Należy to uczynić w sekcji **catch(...) {...}**

```
try {  
    kod który może  
    spowodować  
    wyjątki  
}  
catch (KlasaWyjatkul e)  
{  
    obsługa wyjątku  
    throw e  
}
```



TWORZENIE WŁASNYCH WYJĄTKÓW.

```
public class MojNowyWyjatek extends  
    Throwable {...}
```

```
public class MojNowyWyjatek extends  
    Exception {...}
```

Częściej stosowane!

Nowe wyjątki tworzymy w celu łatwego ich wychwytywania po typie w sekcji catch().

Rzadko kiedy dodajemy nowe cechy w klasach potomnych wyjątków.