Inżynieria oprogramowania

Lecture XXX

Java™ – część II

Bartosz Walter

<Bartek.Walter@man.poznan.pl>

Plan

- Java i JVM
- Typy danych
- Klasy, obiekty, interfejsy
- Pola i metody
- Wyjątki
- Przegląd bibliotek

Błędy w programach

W czasie wykonywania kodu mogą pojawić się różne błędy

- problemy ze swoim stanem wewnętrznym po wywołaniu metody (niezgodności wartości zmiennych instancyjnych)
- błąd w danych przekazanych metodzie (np wartość null)
- błąd realizacji jakiejś operacji (błąd operacji plikowej, błąd sieciowy, brak zasobu etc.)
- naruszenie podstawowych zasad kontraktu metody

Kody błędu

Różne sposoby kontroli błędów w programach

- zmienna globalna **int errno**
- zwracanie kodu błędu z metody (0 == OK)

ale

- można zignorować
- sprawdzając, trzeba wiedzieć czego szukać
- sprawdzanie wyczerpujące powoduje zaciemnienie kodu
- trzeba znać wszystkie możliwe przyczyny błędów

Wyjątki

Wyjątek jest zgłaszany w razie wystąpienia nieoczekiwanego błędu.

Wyjątek ten może być wychwycony przez klauzulę, która została zarejestrowana wcześniej. Jeśli wyjątek nie zostaje wychwycony, to zostaje on obsłużony przez domyślną procedurę obsługi, która zwykle wypisuje komunikat na temat miejsca, gdzie dany wyjątek został zgłoszony.

Wyjątki

```
try {
    // niebezpieczna operacja
} catch (Exception1 ex) {
    // jeżeli to wyjątek typu Exception1, to tutaj próbujemy z niej się wydostać
} catch (Exception2 ex) {
    // jeżeli Exception2 – to tutaj
}
```

Throw vs. throws

Do zgłaszania wyjątków służy klauzula **throw**, która wymaga podania obiektu reprezentującego wyjątek.

Wyjątki kontrolowane, zgłoszone przez daną metodę, podaje się w klauzuli **throws** w postaci list oddzielonych przecinkami (ściśle pilnowane).

Throw vs. throws

```
public int Fib(int no) throws IllegalArgumentException {
    if (no < 0) {
        throw new IllegalArgumentException(„Liczba mniejsza od 0”);
    } else {
        if (no < 2) {
            return 1;
        } else {
            return Fib(no – 2) + Fib (no – 1);
        }
    }
}
```

Obsługa wyjątków

Co można zrobić z wyjątkiem?

- wychwycić i obsłużyć
- wychwycić i przekształcić na inny wyjątek zadeklarowany w klauzuli naszej metody (*exception chaining*)
- wymienić ten wyjątek w klauzuli throws naszej metody i pozwolić na jego propagację do miejsca wywołania naszej metody, nie zajmując się jego obsługą (chyba że w klauzuli finally)

Klauzula *finally*

```
try {
    // niebezpieczna operacja
} catch (Exception1 ex) {
    // jeżeli to wyjątek typu Exception1, to tutaj próbujemy z niej się wydostać
} catch (Exception2 ex) {
    // jeżeli Exception2 – to tutaj
} finally {
    // wykona się ZAWSZE po try lub catch, niezależnie czy wyjątek się pojawi, czy nie.
}
```

Klauzula *finally*

- Kod, który może generować wyjątek umieszczamy w bloku **try**.
- Po bloku **try** występują klauzule **catch**, które są przeszukiwane po wystąpieniu wyjątku w bloku **try** w celu rozpoznania wyjątku. Jeśli wyjątek zostanie rozpoznany zostaje on wyłapany przez **catch** i wykonywane są instrukje przeyporządkowane danej klauzuli **catch**. Jeśli wyjątek nie zostanie rozpoznany przez żadną klauzulę **catch** jest on propagowany na zewnątrz bloku **try**.
- Po bloku **try** i klauzulach **catch** może występować klauzula **finally**. Jeśli występuje, to jest ona wykonywana zawsze po wszystkich czynnościach związanych z obsługą klauzul **catch** niezależnie od tego, czy wyjątek został wygenerownay i wychwycony, czy nie wychwycony, czy blok **try** zakończył się w wyniku przepływu sterowania, czy też w wyniku działania instrukcji *break* czy *continue*.

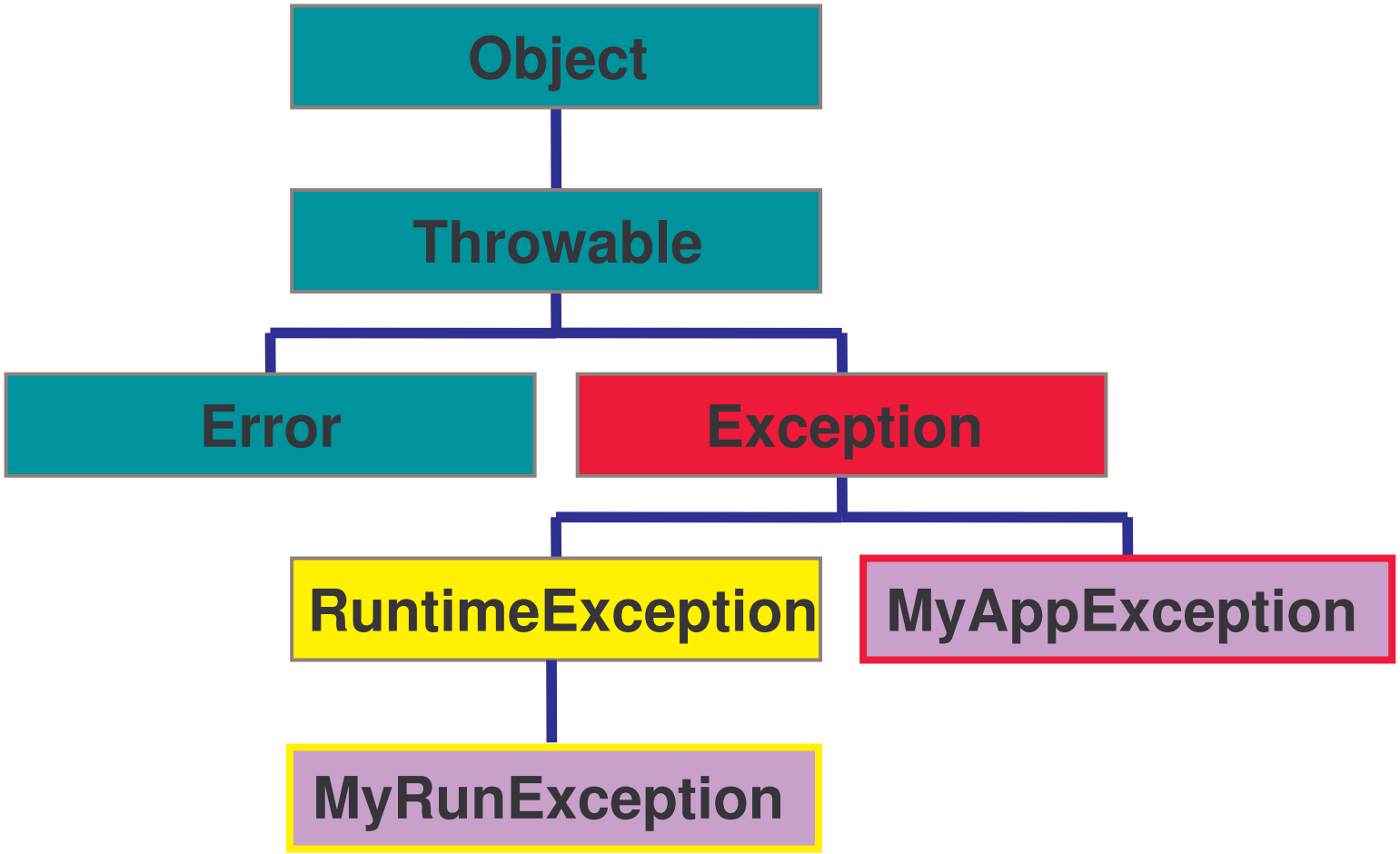
Wyjątki sprawdzane i niesprawdzane

W Javie istnieją dwa rodzaje wyjątków

- **sprawdzane** (ang. *checked exceptions*) - kompilator jest w stanie sprawdzić, że każda metoda zgłasza tylko te wyjątki, które są jawnie wymienione w jej deklaracji.

- **niesprawdzane** (ang. *uncheckcked exceptions*) - standardowe wyjątki zgłaszane przez system wykonawczy (są podklasami klas *RuntimeException* i *Error*). Każda metoda może je zgłosić i nie musi to być uwzględnione w jej kontrakcie.

Hierarchia wyjątków w Javie



Praktyczne metody wyjątków

- **String getMessage()** - zwraca napis podany kontruktorowi wyjątku *Exception*
- **String toString()** - zwraca napis reprezentujący nazwę wyjątku (klasy wyjątku)
- **void printStackTrace()** - wypisuje na strumień błędów tzw. *call stack trace* czyli sekwencję metod, która została wywołana do momentu wyrzucenia wyjątku

```
java.lang.Throwable
    at boo.hoo.StackTrace.bar(StackTrace.java:223)
    at boo.hoo.StackTrace.foo(StackTrace.java:218)
    at boo.hoo.StackTrace.main(StackTrace.java:54)
```

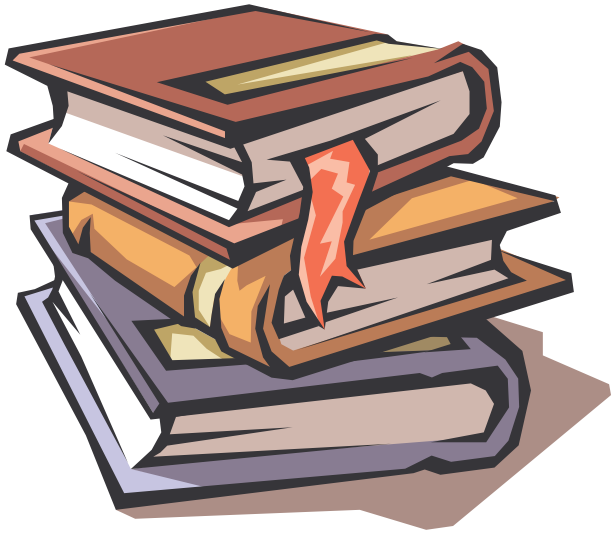
Rady

- nie rzucać wyjątków, gdy można im zapobiec
  - np. iterowanie po tablicy, aż skończą się jej elementy
- jeżeli to możliwe, tworzyć zawsze nowe wyjątki w celu ich zgłoszenia
  - throw new Exception()
- przechwytywać możliwie najbardziej szczegółowe wyjątki
  - catch (Exception ex) $\Rightarrow$ catch (SQLException ex)
- dla sytuacji błędnych związanych z logiką aplikacji stosować wyjątki sprawdzane, dla błędów, których aplikacja nie może przewidzieć - niesprawdzane
  - NotDebitAccountException vs. FileNotFoundException

Podsumowanie

- Wyjątki są obiektowym sposobem na obsługę błędów
- W Javie są dwa rodzaje wyjątków: sprawdzane i niesprawdzane
- Wyjątków nie można przeoczyć, można je przechwycić i obsłużyć

Lektura



1. *http://www.javaworld.com/javaworld/jw-07-1998/jw-07-exceptions.html*
2. *http://java.sun.com/docs/books/tutorial/essential/exceptions/*
3. *http://www.javaworld.com/javaworld/javatips/jw-javatip124.html*
4. *http://java.sun.com/developer/technicalArticles/Programming/Stacktrace/*

Q & A