Wyjątki

Tomasz Borzyszkowski

Wyjątki podstawy

Wyjątek w Java jest obiektem, który opisuje sytuację błędną powstałą w kodzie. Zaistnienie sytuacji błędnej w metodzie powoduje utworzenie obiektu reprezentującego wyjątek i zgłoszenie go przez metodę, w której błąd wystąpił. Następnie metoda może sama obsłużyć wyjątek lub przesłać go do obsługi przez inne metody/obiekty.

Wyjątki mogą być zgłaszane przez maszynę wirtualną Javy lub przez kod użytkownika. Wyjąki zgłaszane przez maszynę wirtualną są związane z tzw. *błędami fatalnymi*, natomiast zgłaszane przez użytkownika z błędami związamymi z logiką programu.

Składnia programu obsługującego wyjątki bazuje na pięciu słowach kluczowych: **`try`**, **`catch`**, **`throw`**, **`throws`** i **`finally`**.
Monitorowany kod znajduje się w bloku **`try`**. Zgłoszenie wyjątku wewnątrz bloku **`try`** powoduje powstanie obiektu-wyjątku i jego ewentualne przejęcie przez odpowiedni blok **`catch`**.
O reszcie na kolejnych slajdach.

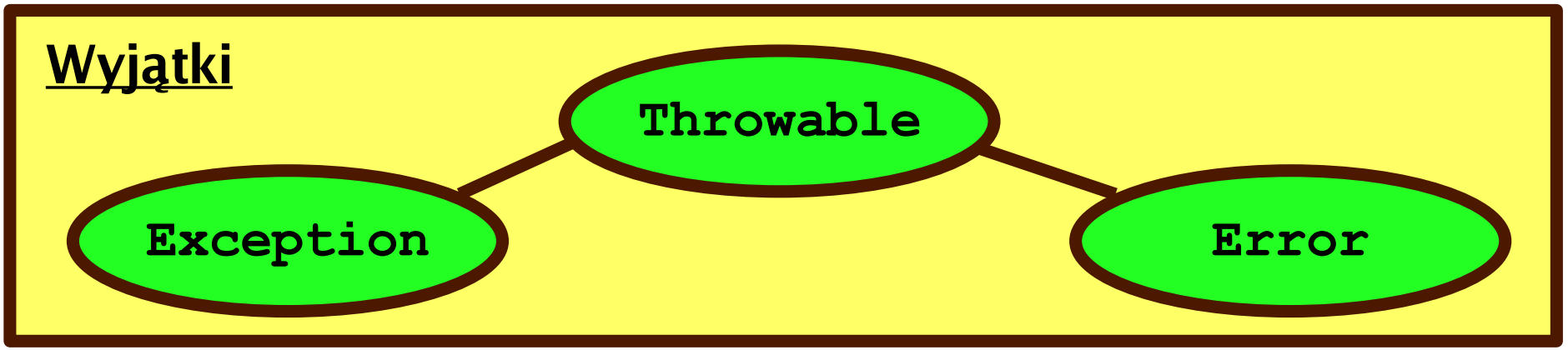
Wyjątki składnia

```
try {
  // monitorowany blok kodu
}
catch (ExceptionType1 exOB){
  // to się wykona, jeżeli zostanie zgłoszony
  // wyjątek typu ExceptionType1
}
catch (ExceptionType2 exOB){
  // to się wykona, jeżeli zostanie zgłoszony
  // wyjątek typu ExceptionType2
}
finally {
  // kod wykonywany po zakończeniu bloków
  // try i catch; służy do zwalniania zasobów
}
```

Wyjątki typy

Zobacz: **Exc{0,1}.java**

Wszystkie wyjątki są podklasami standardowej klasy **`Throwable`**.



Klasa **`Exception`** i potomne, służą do opisywania sytuacji błędnych, które mogą być spowodowane przez kod użytkownika lub mogą być przez kod użytkownika wykryte i obsłużone.
Ważną podklasą klasy **`Exception`** jest klasa **`RuntimeException`**.
Jest ona (i jej podklasy) odpowiedzialna za błędy takie jak:
dzielenie przez zero, indeks tablicy poza zakresem, itp.

Klasa **`Error`** i potomne, są używane przez maszynę wirtualną do zgłaszania *błędów fatalnych*, takich jak: przepełnienie stosu, ...

`try & catch`

Zobacz: **Exc{2,3,4,5}.java**

Mechanizm wyłapywania wyjątków oferowany przez maszynę wirtualną jest przydatny podczas testowania programu. Oddając wersję ostateczną często chcemy sami obsłużyć wyjątek.

Obsługa wyjątków przez program pozwala programiście:

- Przewidzieć jakich błędów można się spodziewać
- Obsłużyć błędy w sposób nie powodujący przerwania pracy programu

Aby samemu obsłużyć błąd powodujący przerwanie programu należy umieścić go w bloku **`try`**, a następnie w bloku **`catch`** umieścić typy wyjątków, na które chcemy reagować oraz związać z nimi kod obsługujący zgłoszony wyjątek.

Należy pamiętać, że po obsłudze wyjątku przez blok **`try`**/**`catch`** program nie wraca do komendy następnej w bloku **`try`** lecz przechodzi do wykonania pierwszej instrukcji za blokiem **`try`**/**`catch`**.

Zagnieżdżony `try`

Instrukcja **`try`** może występować w bloku instrukcji innej instrukcji **`try`**. Konstrukcja taka powoduje, że wyjątki zgłaszane przez wewnętrzy blok **`try`** będą posiadały swój kontekst wywołania, inny niż wyjątki bloku zewnętrznego. Jeżeli wewnętrzny blok **`try`** zgłosi wyjątek, dla którego nie posiada odp. sekcji **`catch`**, będzie przeszukiwany kontekst bloku zewnętrznego w poszukiwaniu odp. sekcji **`catch`**.

Zobacz: **Exc6.java**

Zagnieżdżone instrukcje **`try`**/**`catch`** nie muszą występować w tak jawny sposób jak w poprzednim przykładzie.
Zagnieżdżenia takie otrzymamy, gdy wewnątrz bloku **`try`** wywołamy metodę zawierającą swoją własną instrukcję **`try`**/**`catch`**.

Zobacz: **Exc7.java**

Instrukcja `throw`

Zobacz: **Exc8.java**

Dotychczas obsługiwaliśmy jedynie wyjątki zgłaszane przez środowisko Javy. Istnieje jednak możliwość zgłaszania wyjątków przez nasz program. Służy do tego komenda:

```
throw Obiekt_klasy_Throwable;
```

Obiekt ***`Obiekt_klasy_Throwable`*** musi być klasy **`Throwable`** lub potomnej. Wykonanie komendy **`throw`** powoduje natychmiastowe przerwanie sekwencyjnego wykonania programu.
Wykonanie programu przenosi się do najbliższej sekcji obsługi zgłoszonego wyjątku.
Jeżeli takiej sekcji nie ma, to program zostanie zatrzymany, a domyślny program obsługi wypisze ścieżkę wywołań metod aż do zgłoszonego wyjątku.

Istnieją dwie metody otrzymania obiektu klasy **`Throwable`**, pierwsza polega na użyciu parametru sekcji **`catch`**, drugi na utworzeniu obiektu operatorem **`new`**.

Deklaracja `throws`

Jeżeli metoda zgłasza wyjątek, którego sama nie obsługuje, to musi opisać takie zachowanie w swoim nagłówku.
Służy do tego słowo kluczowe **`throws`**, które umieszczamy zaraz po deklaracji metody. Po tym słowie wymieniamy typy wszystkich wyjątków zgłaszanych przez metodę, za wyjątkiem **`Error`** i **`RuntimeException`** i ich podklas.

Ogólna postać definicji metody zgłaszającej nieobsługiwane wyjątki jest następująca:

```
typ nazwa_metody(lista-parametrów)
  throws lista-wyjątków
{
   // ciało metody
}
```

Zobacz: **Exc9{a,b}.java**

Sekcja `finally`

Zobacz: **Exc{A,B}.java**

W momencie zgłoszenia wyjątku normalne, liniowe wykonanie programu zostaje zaburzone. W szczególnych wypadkach może to nawet oznaczać natychmiastowy powrót z metody do sekcji **`catch`** metody wywołującej (być może w innym obiekcie).
Takie zachowanie może sprawiać problemy metodom rezerwującym zasoby. Trzeba by wówczas pamiętać o zwalnianiu zasobów w każdej sekcji **`catch`**.

Pomóc w takich przypadkach może sekcja **`finally`**. Jest ona sekcją kodu wykonywaną zaraz po sekcji **`try`**/**`catch`** ale przed kodem następującym po sekcji **`try`**/**`catch`**.
Sekcja **`finally`** wykona się zawsze po sekcji **`try`**/**`catch`**, niezależnie od tego, czy jakiś wyjątek został zgłoszony w bloku **`try`** oraz od tego czy jakiś wyjątek został obsłużony przez sekcję **`catch`**.
Takie zachowanie bloku **`finally`** nadaje się doskonale do zwalniania zasobów rezerwowanych przez metodę.

Tworzenie własnych wyjątków

Java posiada wbudowane wyjątki obsługujące najczęściej spotykane błędy. Jednak często zachodzi potrzeba zdefiniowania nowych wyjątków specyficznych dla naszego programu.

Utworzenie nowej klasy wyjątku jest proste. Wystarczy zdefiniować klasę dziedziczącą po klasie **`Exception`**. Nowo zdefiniowana klasa nie musi nawet niczego implementować, ale może.

Najczęściej implementowane zmiany w klasach nowych wyjątków, to:

- Dodatkowe zmienne instancyjne przechowujące stan sytuacji błędnej
- Pokrywanie standardowych metod klasy **`Throwable`** takich, jak: **`getLocalizedMessage()`**, **`getMessage()`**, **`printStackTrace()`** i **`toString()`**

Zobacz: **ExcC.java**