15. Wyjątki

1 Wyjątki

Wyjątki opisują pewną nieprawidłową, błędną lub wyjątkową sytuację. Wiążą się z nimi dwie czynności:

- wyrzucanie (inaczej zgłoszenie) wyjątku (ang. *throw*),
- obsłużenie (inaczej złapanie) wyjątku (ang. *catch*).

Wyjątki w języku Java są obiektami klasy `Throwable` z pakietu `java.lang` lub jej dowolnej podklasy (najczęściej `Exception`). Wyróżniamy dwa typy wyjątków: niekontrolowane i kontrolowane.

Zadanie 01 Zapoznaj się z kodem zawartym w pliku `StackAndExceptions.java`. Skompiluj go, a następnie uruchom bez argumentów programu i z kilkoma argumentami:

```
java StackAndExceptions one two three four five
```

Jaki i jakiego typu wyjątek zostaje zgłoszony gdy program jest uruchamiany z więcej niż trzema argumentami? Jak do tej pory obsługiwałeś tego typu wyjątkowe sytuacje?

1.1 Rzucanie wyjątków

Rzucenie obiektu wyjątku odbywa się poprzez słowo kluczowe `throw`:

```
if (array.length > 3)
    throw new RuntimeException("Too many elements.");
```

Zadanie 02 Zmodyfikuj metodę `push()` klasy `Stack` tak, aby rzucała wyjątek klasy `RuntimeException`, gdy nie można dodać już elementu do stosu. Czy poza zmianą komunikatu błędu zgłaszanego przez wirtualną maszynę Javy, działanie programu zmieniło się?

1.2 Obsługa wyjątków

Za obsługę wyjątków odpowiada klauzula `try-catch`:

```
try {
    // kod, w ktorym moze zostac rzucony wyjatek
}
catch (Exception e) {
    // kod wykonany, gdy zostanie zgloszony wyjatek
}
```

Po słowie kluczowym `catch` następuje deklaracja zmiennej dla obiektu wyjątku, z której będzie można korzystać wewnątrz bloku `catch`. Jest to klasa wyjątku który chcemy obsłużyć lub jego klasa nadrzędna.

Zadanie 03 Obsłuż wyjątek `RuntimeException` w metodzie `main()` w taki sposób, aby wypisać na ekran odpowiedni komunikat bez przerywania całego programu.

1.3 Propagacja wyjątków

Podczas wystąpienia wyjątku w metodzie *A*, jej wykonanie jest przerywane i sterowanie powraca do metody, w której metoda *A* została wywołana. Jeżeli to miejsce kodu znajduje się w bloku `try`, wyjątek jest obsługiwany, w przeciwnym przypadku wyjątek propagowany jest dalej, aż do obsłużenia przez wirtualną maszynę Javy.

Każda metoda, w której może wystąpić wyjątek kontrolowany, a która sama go nie obsługuje, musi zawierać o tym informacje, np.:

```
1 void foo() throws Exception {
2     if (somethingBadHappened())
3         throw new Exception("Something bad happened!");
4 }
```

Zadanie 04 Zamień klasę wyjątku rzucanego w metodzie `push()` z `RuntimeException` na `Exception`. Czy teraz można skompilować kod programu?

Wyjątek `Exception` to wyjątek kontrolowany, więc dodaj do metody `push()` klauzulę `throws`.

Zadanie 05 Dopisz klasy `FullStackException` oraz `EmptyStackException` dziedziczące po klasie `Exception` i przesłaniające metodę `getMessage()`, która zwraca komunikat wyjątku.

Zadanie 06 Zmodyfikuj metody `push()` oraz `pop()` tak, aby w sytuacjach wyjątkowych zgłaszały wcześniej utworzone wyjątki. Dodaj metodę `swap()`, która zamieni miejscami dwa górne elementy stosu. w jej implementacji wykorzystaj poprzednie metody.

Użyj jej w metodzie `main()`, pamiętaj o dodaniu klauzuli `throws` w odpowiednich miejscach.

Zadanie 07* Zmodyfikuj program tak, aby elementy do stosu były wczytywane z pliku (jedna linia — jeden element) o nazwie podawanej jako parametr programu. Klasa `Stack` powinna posiadać metodę `readItemsFromFile(String fileName)`, która zostanie wykorzystana w konstruktorze jednoargumentowym.

Wyjątki związane z błędnym czytaniem pliku powinny być obsługiwane w metodzie `main()`.

Zadanie 08 dodatkowe (1 pkt) Zaimplementuj powyższy program z klasą `Stack` w języku C++ (bez zadania 07.), definiując i obsługując własne klasy wyjątków.