Kurs programowania

Wykład 11

Wojciech Macyna

Refleksja w Javie

Zastosowanie

- Umożliwia analizę klas w czasie wykonywania programu.

Można uzyskać następujące informacje:

- Nazwa klasy
- Modyfikatory dostępu klasy
- Pakiet klasy
- Konstruktory
- Metody
- Pola
- Interjesy

Refleksja w Javie

Przykłady zastosowania

- Wykonywanie testów JUnit - można wtedy wywołać metody z adnotacją @Test
- Analiza klas związanych z tabelami baz danych.

Wady refleksji

- Spadek efektywności działania - refleksja uniemożliwia użycie niektórych optymalizacji na maszynie wirualnej Javy.
- Problemy związane z bezpieczeństwem - refleksja wymaga uprawnień w runtime, co nie zawsze jest możliwe.
- Może powodować aspekty uboczne, np. umożliwia dostęp do pól prywatnych, co w normalnym działaniu nie jest możliwe.

Przykład

`ReflectionExample.java` - obsługa refleksji dla klasy:

`RentCar.java`

Wyrażenia lambda

Cel

Wyrażenia lambda upraszczają kod i czynią go bardziej przejrzystym, wszędzie tam, gdzie dotychczas wykorzystywaliśmy interfejsy funkcyjne (definiujące pojedynczą metodę)

Przykłady interfejsu funkcyjnego

- `Runnable` - wykorzystywany do przekazania zadania do wątka (instancji Thread).
- `Comparable` - służący do określenia sposobu sortowania obiektów.
- `ActionListener` - określający kod obsługi zdarzenia akcji.

Przykład użycia

Zamiast:

```
1  button.addActionListener(
2          new ActionListener() {
3          @Override
4          public void actionPerformed(ActionEvent e) {
5              methodToInvoke();
6          }
7      });
```

Wyrażenie lambda:

```
1  button.addActionListener(e -> methodToInvoke());
```

Ogólna zasada

Zamiast:

```
1  SomeInterface obj = new SomeInterface() {
2          @Override
3          public SomeType someMethod(T1 v1, T2 v2) {
4              return(someValue);
5          }
6      }
```

Można użyć:

```
1  SomeInterface obj = (v1, v2) -> someValue;
```

Wyrażenia lambda

Referencja do metod

W javie 8 powstała referencja do metod, jest to trochę powiązane z wyrażeniami lambda. Odwołujemy się do statycznej metody Klasa::statycznaMetoda i używamy jej w interfajsie funkcyjnym jako referencji do funkcji.
Można to samo zrobić z instancją klasy, dla metody która nie jest statyczna. W tym celu odwołujemy się objekt::mojaMetoda na stworzonym wcześniej obiekcie i nie statycznej metodzie.

Przykłady interfejsu funkcyjnego

Nie zawsze istnieje potrzeba tworzenia własnych interfejsów funkcyjnych. Można skorzystać z paczki: java.util.function.

- `UnaryOperation` - Obiekt T i zwraca obiekt T, metoda to apply().
- `BinaryOperator` - operacja na dwóch obiektach T, zwraca obiekt T, metoda to apply().
- `Consumer` - Operacja na obiekcie T, metoda to accept().
- `Supplier` - zwraca obiekt T, metoda to get().
- `Function` - Operacja na obiekcie T zwraca obiekt R, metoda to apply().
- `Predicate` - jeśli obiekt T spełnia jakieś ograniczenia, zwraca boolean, metoda to test().

Wyrażenia lambda

Przykłady

`LambdaEx1.java`

`LambdaEx2.java`

`LambdaFunction.java`

`LArg.java`

`WildGen.java`