Lab 9

Programowanie obiektowe - Java

Zad 1 (7 pkt)

- (1 pkt) Utwórz interfejs generyczny `AdditiveValue` z metodami:
  - `getValue`
  - domyślnie zaimplementowaną metodą add wyrzucającą wyjątek `NotImplementedException` - czy wyjątek trzeba deklarować?
- (1 pkt) Utwórz enumerator `CommonLongValues` implementujący interfejs `AdditiveValue<Long>` i zawierający pola: `ZERO`, `ONE`, `MINUS_ONE`
  - Przykładowo: wywołanie `CommonLongValues.ZERO.getValue()` powinno zwrócić 0
- (1 pkt) Utworzyć fabrykę `AdditiveLongValueFactory` (klasa nie może być dziedziczona ani instancjonowana), która zawiera metodę `create` (statyczną) do generowania klas implementujących `AdditiveValue<Long>` przy czym argumentem metody jest wartość typu `long`.
- (2 pkt) Utworzyć klasę `LongValue`, która nadpisuje domyślną metodę `add` w której tworzy nowy obiekt `LongValue` oraz przyjmuje wartosć typu `long` w konstruktorze. Dodatkowo należy utworzyć klasę `MutableLongValue` działającą analogicznie, jednak modyfikującą własną wartość. Należy nadpisać metodę `toString` w klasach, aby zawracały przechowywaną wartość.
- (1 pkt) Utworzyć analogiczną do `LongValue` klasę `StringValue`. Klasa ta posiada dodatkowo przeciążony konstruktor umożliwiający podanie wielu argumentów klasy String (`String...`)
- (1 pkt) Przeciążyć metodę `add` w klasie `StringValue` dwoma metodami:
  - Pierwsza ma za zadanie dodawać wartość typu `Long` do łańucha znakowego
  - Druga ma za zadanie przyjmować argumenty typu `Number` i zwracać błąd, który należy deklarować (`extends Exception`)

Do każdego przykładu proszę zamieścić stosowny przykład działania

Zad 2 (3 pkt)

Skopiuj do projektu (do paczki mysteroius) klasę

```
package mysterious;

import java.util.Random;
```

```
public class MysteriousClass {

    private int pickedNumber;
    private int otherNumber;

    public MysteriousClass() {
        this.pickedNumber = new Random().nextInt(1000);
        this.otherNumber = new Random().nextInt(1000);
    }

    public static String greet(String name) {
        return “Hello, “ + name + “!”;
    }

    protected int getOtherNumber() {
        return otherNumber;
    }

    private int add(int number) {
        return number + pickedNumber;
    }
}
```

Następnie bez jej modyfikacji:

- (1 pkt) Załaduj klasę za pomocą klasy `ClassLoader` (dostępna w metodzie `getClass().getClassLoader()` i utwórz jej instancję przy pomocy zwróconego obiektu (`newInstance`)
- (1 pkt) Odczytać nazwy metod deklarowanych przez klasę. Wywołać metodę `add` na utworzonym obiekcie (należy użyć metody `invoke` z załadowanej klasy)
- (1 pkt) Odczytać nazwy pól deklarowanych przez klasę. Odczytać wartość pola `pickedNumber`

Uwaga przed dostępem do prywatnych elementów klasy należy na nich użyć metody `.setAccessible(true)`