Zadanie 4 3, 6 listopada 2025 r.

kurs języka Java

zbiory asocjacyjne

Instytut Informatyki
Uniwersytetu Wrocławskiego Paweł Rzechonek

Zadanie 1.

Zdefiniuj klasę `Pair`, która będzie przechowywać pary klucz–wartość, gdzie klucz jest identyfikatorem typu `String` a skojarzona z nim wartość to liczba rzeczywista typu `double`. Klucz powinien być publicznym polem niemodyfikowalnym (powinien składać się tylko z małych liter alfabetu angielskiego oraz nie może to być łańcuch pusty ani wartość `null`), a wartość polem prywatnym, które spoza klasy można odczytać tylko za pomocą gettera `get()` i zmodyfikować tylko za pomocą setera `set(double)`.

```
public class Pair implements Cloneable {
    public final String key;
    private double value;
    //...
}
```

W klasie tej zdefiniuj konstruktory a także metody `toString()` oraz `equals(Object)`, przy czym zakładamy, że dwie pary są równe, gdy mają takie same klucze.

W klasie `Para` zaimplementuj również interfejs `Cloneable`, który będzie wykonywać płytkie klonowanie takich obiektów.

Dodatkowo zdefiniuj klasę `PairProb`, dziedzicząca po `Pair`, w która będzie służyć do przechowywania prawdopodobieństw jakichś zdarzeń. Należy pilnować, aby w obiektach tej klasy znajdowały się tylko wartości z zakresu od 0 do 1 (próbę wpisania tam innej wartości należy sygnalizować zgłoszeniem wyjątku).

```
public class PairProb extends Pair {
    //...
}
```

Zadanie 2.

Zdefiniuj interfejs `AssocColl`, który będzie definiować narzędzia do zarządzania zbiorem zmiennych o unikatowych nazwach. Przede wszystkim dostarczyć metod, które pozwolą sprawdzić czy zmienna została zdefiniowana `search(String)`, odczytać wartość zmiennej o ile jest zdefiniowana `get(String)` oraz wstawić nową albo zaktualizować istniejącą zmienną `set(String, double)`; zdefiniuj też metodę, która udostępni w postaci tablicy nazwy wszystkich zdefiniowanych w zbiorze zmiennych `names()`.

```
public interface AssocColl {
    void set(String k, double v);
    double get(String k);
    boolean search(String k);
    String[] names();
    //...
}
```

W interfejsie tym zdefiniuj także metodę domyślną `defaultToString()` , która będzie tekstowo reprezentowała zawartość zbioru zmiennych (w interfejsie nie możemy niestety dostarczyć domyślnej implementacji metody `toString()` z klasy `Object` – w interfejsach nie wolno nadpisywać metod z klasy `Object`).

> The general rule is that an object cannot inherit two implementations of a single method. This rule applies in several situations - for example, when you try implementing two interfaces, both having the same method with default implementations.
>
> Similarly, when you provide a default implementation of the java.lang.Object's toString method in an interface, you are introducing an ambiguity, because any class implementing your interface would also inherit from Object one way or the other, so the compiler would need to decide between two implementations (despite the fact that you are trying to tell the compiler through the @override attribute that you want your default implementation to win).

Dodatkowo zdefiniuj interfejs `AssociativeCollection`, który będzie kompozycją interfejsów `Cloneable` oraz `AssocColl`, i który rozszerzy całą funkcjonalność o głębokie sklonowanie całego zbioru zmiennych `clone()` oraz o usuwanie zmiennej ze zbioru `del(String)` i informację o ilości wszystkich zmiennych w zbiorze `size()`.

```
public interface AssociativeCollection
extends Cloneable, AssocColl {
    void del(String k);
    int size();
    //...
}
```

W interfejsie tym nadpisz metodę domyślną `defaultToString()` , która będzie tekstowo reprezentowała zawartość zbioru zmiennych – zmień nieznacznie implementację tej metody w stosunku do interfejsu `AssocColl`, aby można było łatwo wychwycić różnice w wynikach.

Zadanie 3.

Zdefiniuj abstrakcyjną klasę `SetVar` służącą do przechowywania zbioru zmiennych, która będzie implementować interfejs `AssociativeCollection`. W klasie tej umieść deklarację abstrakcyjnej metody `clear()`, której zadaniem będzie usunięcie wszystkich zmiennych ze zbioru.

Następnie zdefiniuj klasę `ArraySetVar` dziedziczącą po `SetVar`, zaimplementuje wszystkie odziedziczone metody abstrakcyjne. Do implementacji zbioru zmiennych wykorzystaj w tej klasie tablicę par `Pair[]`. która będzie przechowywać zbiór zmiennych w tablicy par `Pair[]`.

```
public class ArraySetVar implements AssociativeCollection {
    protected Pair[] vars; // tablica ze zmiennymi
    protected int siz; // ile zmiennych jest aktualnie w tablicy

    public ArraySetVar clone() { ... }
    public void clear() { ... }
    //...
}
```

Rozmiar tablicy ustal w konstruktorze. Definiując metodę `toString()` wykorzystaj metodę `defaultToString()` pochodzącą z interfejsu. Implementując metodę klonowania zbioru zmiennych skorzystaj z faktu, że tablice są klonowalne (ale płytko); klonowanie ma być głębokie, więc wszystkie zmienne z oryginalnej tablicy też należy sklonować.

Zadanie 4.

Na koniec napisz program, który rzetelnie przetestuje działanie zbioru zmiennych. Pamiętaj o popranym klonowaniu zbiorów.