laboratorium: zadanie 12 *termin: 11/12 stycznia 2012 r.*

KURS PROGRAMOWANIA W JAVIE

DRZEWO BST

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zadanie 1.

Zdefiniuj klasę sparametryzowaną do pamiętania zbioru dynamicznego w postaci drzewa binarnych poszukiwań `DrzewoBST<T>`. Klasa ta ma być opakowaniem dla homogenicznej struktury tworzonej wewnątrz na obiektach typu `Węzeł`. Twoja klasa powinna implementować operacje słownikowe (sprawdzać czy element o zadanej wartości istnieje `szukaj`, dodawać nowy element do zbioru `wstaw` i usuwać element zadany element `usuń`) zdefiniowane w interfejsie `Słownik<T>`.

```
class DrzewoBST<T extends Comparable<T>> implements Słownik<T>
{
    private class Węzeł <T extends Comparable<T>>
    {
        Węzeł<T> lewy, prawy, ojciec;
        T dane;
        // ...
    }

    private Węzeł<T> korzeń;

    // ... metody słownikowe
    public String toString () { /*...*/ }
}
```

Przy próbie włożenia do drzewa wartości `null` należy zgłosić wyjątek `NullPointerException`. Dopisz też metody podające ilość elementów w zbiorze `ile()` i usuwającej wszystkie elementy z drzewa `czyść()`.

Zdefiniowana przez Ciebie klasa `DrzewoBST<T>` powinna prawidłowo działać w warunkach pracy współbieżnej, gdy wiele wątków jednocześnie pracuje na takim drzewie.

Zadanie 2.

Do klasy reprezentującej drzewo BST dopisz metodę `iterator()`, która będzie tworzyła i zwracała iterator związany z danym drzewem. Zdefiniuj więc klasę iteratora dedykowaną dla Twojego drzewa i implementującą interfejs `Iterator<T>`. Iterator ten powinien być wrażliwy na wszelkie zmiany w drzewie, po którym iteruje — jeśli w trakcie iteracji po drzewie zostanie na nim dokonana jakakolwiek zmiana, to następne użycie iteratora powinno skutkować zgłoszeniem wyjątku `ConcurrentModificationException`.

Zadanie 3.

Napisz program konsolowy, rzetelnie testujący działanie zdefiniowanego przez Ciebie drzewa BST. Program ten ma symulować działanie *producentów i konsumentów* (uruchom kilkanaście/kilkadziesiąt wątków dla producentów i konsumentów). Producent ma produkować losowe dane (na przykład losowe łańcuchy znaków) a konsument pobierać dane z wylosowanego przedziału.

Producenci i konsumenci mają wykonywać swoje działania w losowych odstępach czasu. Wszystkie produkty mają być umieszczane w albo pobierane z jednego drzewa BST. Każdy producent i konsument ma wyświetlić zawatość listy po wykonanej przez siebie pracy za pomocą niesynchronizowanej metody `toString()`, która korzysta z iteratora.