https://github.com/alina-puscasu/pao_lab_2022

Programare avansata pe obiecte – laborator 8

Alina Puscasu *alina.puscasu@endava.com*

Generice

- Le putem intalni in metode, clase/interfete
- Ne permit sa refolosim aceleasi clase/metode pe diferite tipuri de date
- Reguli pentru **metodele generice**:
    - Daca folosim ca parametru un tip generic, acesta trebuie sa preceada si return type-ul metodei (il scriem intre < >)
    - Putem avea mai multe tipuri generice separate prin ‘,’
    - Naming convensions:
        1. E – Element and is mainly used by Java Collections framework.
        2. K – Key and is mainly used to represent parameter type of key of a map.
        3. V – Value and is mainly used to represent parameter type of value of a map.
        4. N – Number and is mainly used to represent numbers.
        5. T – Type and is mainly used to represent first generic type parameter.
        6. S – Type and is mainly used to represent second generic type parameter.
        7. U – Type and is mainly used to represent third generic type parameter.
        8. V – Type and is mainly used to represent fourth generic type parameter.

- Reguli pentru **clasele generice**:
    - Dupa numele clasei specificam tipul generic intre < >
    - Putem avea mai multe tipuri generice specificate intre ‘,’. Se aplica aceleasi naming convensions.

- Restrictii:
    - Ca si tipuri generice *NU* putem folosi primitive
    - Tipurile generice (T, S, etc) *NU* se pot instantia
    - Putem restrictiona tipurile generice prin:
    - **Bounded types params**:
        1. Upper bound (T extends SomeClass): tipul elementelor trebuie sa fie clasa dupa extends sau o subclasa a acesteia
        2. Lower bound (T super SomeClass): tipul elementelor trebuie sa fie clasa dupa super sau o superclasa a acesteia
    - **Wildcards**, vrem sa folosim o structura generica ca parametru insa nu vrem sa limitam tipul de date

- E o buna practica mereu sa indicam tipul obiectelor folosite in cazul instantierii claselor generice

https://github.com/alina-puscasu/pao_lab_2022

- Mai multe: https://www.tutorialspoint.com/java_generics/index.htm

Exercitiu

1. Let's say you have an integer array and a string array. You have to write a single method printArray that can print all the elements of both arrays. The method should be able to accept both integer arrays or string arrays.

The code should print the following lines:
1
2
3
Hello
World

Collection(s)

- Interfata Collection este implementata de majoritatea claselor ce desemneaza colectii din pachetul java.util
- Clasa Collections in care se gasesc majoritatea metodelor statice, utile lucrului cu colectii (de ex: sort, binarySearch)

Interfata List si implementari

- Din punctul de vedere al structurii de date, reprezinta implementarea unui vector sau a unei liste
- Poate contine elemente duplicate
- Contine operatii bazate pe pozitie: get, add, remove, set
- Implementari uzuale:
    1. ArrayList
        - ✦ Implementeaza o structura de date de tip vector care poate fi redimensionata dinamic
        - ✦ Accesul la un element se face in timp constant (O(1))
        - ✦ Se recomanda a se folosi atunci cand predomina operatiile de accesare
        - ✦ Poate fi sortata
    2. LinkedList
        - ✦ Implementeaza o structura de date de tip lista dublu inlantuita
        - ✦ Pastreaza ordinea inserarii si poate fi si sortata
        - ✦ Accesul la un element nu se face in timp constant (O(n)), e necesara o parcurgere
        - ✦ Operatiile de add, remove sunt mai rapide decat la ArrayList
        - ✦ Se recomanda a se folosi atunci cand predomina operatiile de actualizare

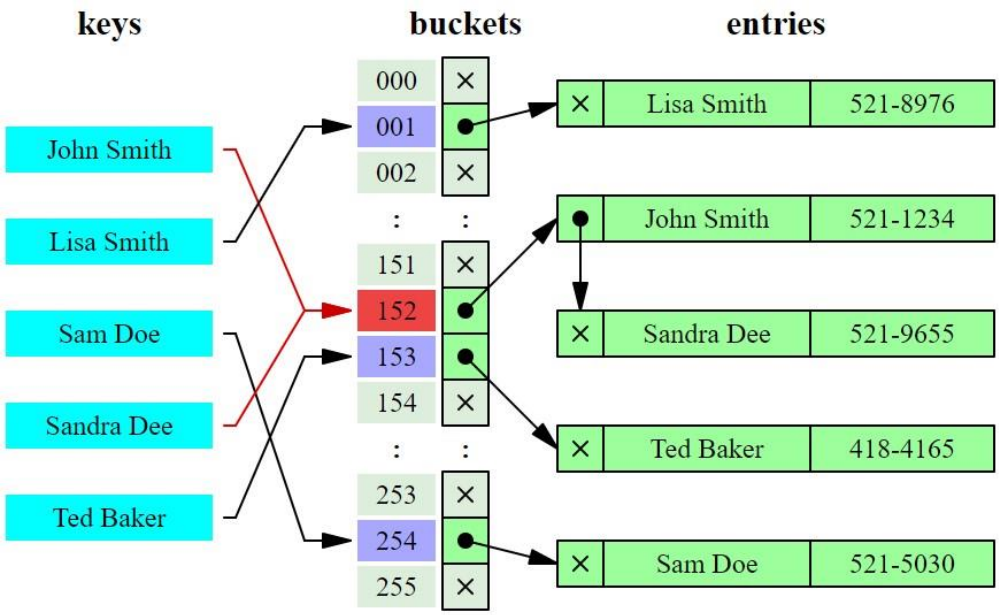
https://github.com/alina-puscasu/pao_lab_2022

Interfata Set si implementari

- Nu contine elemente duplicate
- Implementari uzuale:
    1. HashSet
        - ✦ Neordonat, nesortat
        - ✦ Elementele sunt memorate intr-un hashtable (tabela de dispersie) -> codul de dispersie este calculat pe baza metodei hashCode
        - ✦ Permite si valoarea null, o singura data
    2. TreeSet
        - ✦ Elementele sunt sortate pe baza valorilor, daca folosim constructorul fara parametri, insa putem specifica noi criteriul prin comparatorul pasat ca parametru daca alegem sa folosim constructorul cu parametri
        - ✦ Elementele sunt memorate intr-un arbore binar de tip Red-Black
        - Nu permite null
        - ✦ Complexitate: O(log(n))
    3. LinkedHashSet
        - ✦ Pastreaza ordinea inserarii prin folosirea unei liste dublu inlantuita

Equals si hashcode

Doua obiecte sunt egale daca au acelasi hashcode; viceversa nu este valabil. Nu uitati mereu cand suprascrieti equals sa suprascrieti si hashCode. Mai multe despre acest contract si aici.



Pentru a adauga un obiect in tabela de dispersie se efectueaza pasii:

https://github.com/alina-puscasu/pao_lab_2022

1. Se calculeaza hashCode care ne da indexul bucketului
2. Daca bucketul e gol, se adauga si operatia se incheie
3. Daca bucketul nu e gol, se parcurge si folosind metoda equals se verifica daca obiectul e deja adaugat, daca da, nu se mai adauga, daca nu, il adauga la finalul listei

Cu cat numarul generat de functia de hashCode e mai unic, cu atat dispersia si drept urmare performanta, va fi mai buna: O(1). In caz contrar, vom avea coliziuni si complexitatea va creste prin parcurgerea listei: O(n).

Interfata Map si implementari

- Modeleaza comportamentul colectiilor de tipul cheie-valoare
- Cheile nu pot fi duplicate
- Implementari uzuale:

1. HashMap
   - Intern utilizeaza o tabela de dispersie
   - Neordonat, nesortat
   - Daca dispersia e una buna, complexitatea operatiilor get, put, containsKey, remove poate fi O(1); in caz contrar poate ajunge si la O(n)
   - Putem folosi null atat ca valoare cat si cheie
   - Putem asocia mai multe valori pentru aceeasi cheie -> Map<String, List<String>>

2. LinkedHashMap
   - Pastreaza ordinea inserarii

3. TreeMap
   - Intern utilizeaza un arbore binar de tip Red-Black
   - Sortarea e in ordinea naturala a cheilor daca nu folosim constructorul cu parametri, sau conform comparatorului pasat ca parametru in constructor, daca alegem sa folosim constructorul cu parametri (atentie! aceasta sorteaza doar cheile)
   - Complexitate: O(log(n))

Parcurgerea colectiilor

1. Interfata Iterator<Tip>
   a. next() -> urmatorul element
   b. hasNext() – verifica daca exista un element urmator

https://github.com/alina-puscasu/pao_lab_2022

Interfata nu permite modificarea valorii elementului curent sau adaugarea altor elemente. E utila de folosit cand dorim stergerea unui element.

2. Enhanced for sau for each, care in spate se bazeaza tot pe un iterator