KeyTraining

Curs Java pentru testare automata

JAVA & SELENIUM

Cuprins curs:

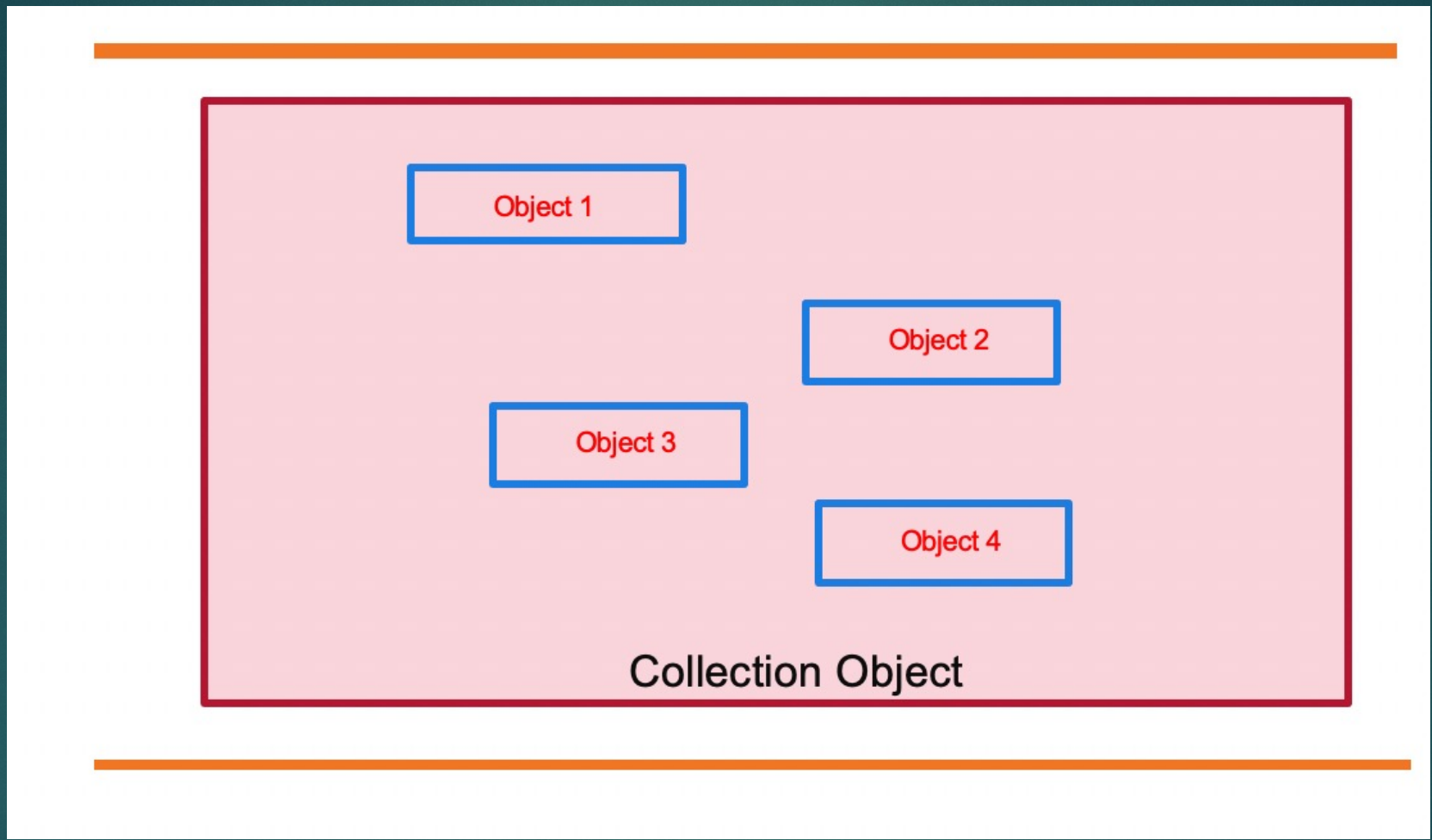
- Collections framework

Collection

- O colecție este un grup de obiecte. În Java, aceste obiecte sunt numite elemente ale colecției.
- Din punct de vedere tehnic, o colecție este un obiect sau container care stochează un grup de alte obiecte ca o singură unitate sau o singură entitate.
- Prin urmare, este cunoscut și ca obiect container sau obiect de colectare în java.
- Un obiect container înseamnă că el conține alte obiecte. Simplu spus, o colecție este un container care stochează mai multe elemente împreună.

Collection

- JVM (Java Virtual Machine) stochează referința altor obiecte într-un obiect de colecție.
- Nu stochează niciodată copii fizice ale altor obiecte, deoarece alte obiecte sunt deja disponibile în memorie și stocarea unei alte copii a obiectelor într-un obiect de colecție ar fi o pierdere de memorie.
- Un grup de obiecte stocate într-un obiect de colecție este prezentat în figura de mai jos. În figură, stocăm 4 obiecte într-un obiect de colecție.


Object 1
Object 2
Object 3
Object 4
Collection Object

Collection

- Un obiect de colectare are o clasă cunoscută sub numele de clasă de colectare sau clasă container.
- Toate clasele de colectare sunt prezente în pachetul java.util.
- Un grup de clase de colecție se numește collection framework în java.

List

- O listă în Java este o colecție pentru stocarea elementelor în ordine secvențială. Ordinea secvențială înseamnă primul element, urmat de al doilea element, urmat de al treilea element și așa mai departe.
- Este folosita pentru a stoca o colecție de elemente în care sunt permise elemente duplicate.
- Interfața List în java are patru subclase concrete:
  - ArrayList
  - LinkedList
  - Vector
  - Stack.

List

- Aceste patru subclase implementează interfața List. ArrayList și LinkedList sunt utilizate pe scară largă în Java pentru a crea o listă. Clasa Vector este depreciată de la JDK 5.

List – Caracteristici generale

- Permite stocarea elementelor duplicate
- Putem adăuga un element în orice poziție
- Menține ordinea de inserare
- Permite stocarea multor elemente nule.
- Utilizează o matrice redimensionabilă pentru implementarea sa. Redimensionabil înseamnă că putem crește sau micșora dimensiunea matricei.
- Oferă un Iterator special numit ListIterator care permite accesarea elementelor în direcția înainte folosind metodele hasNext () și next ();

List – Methods

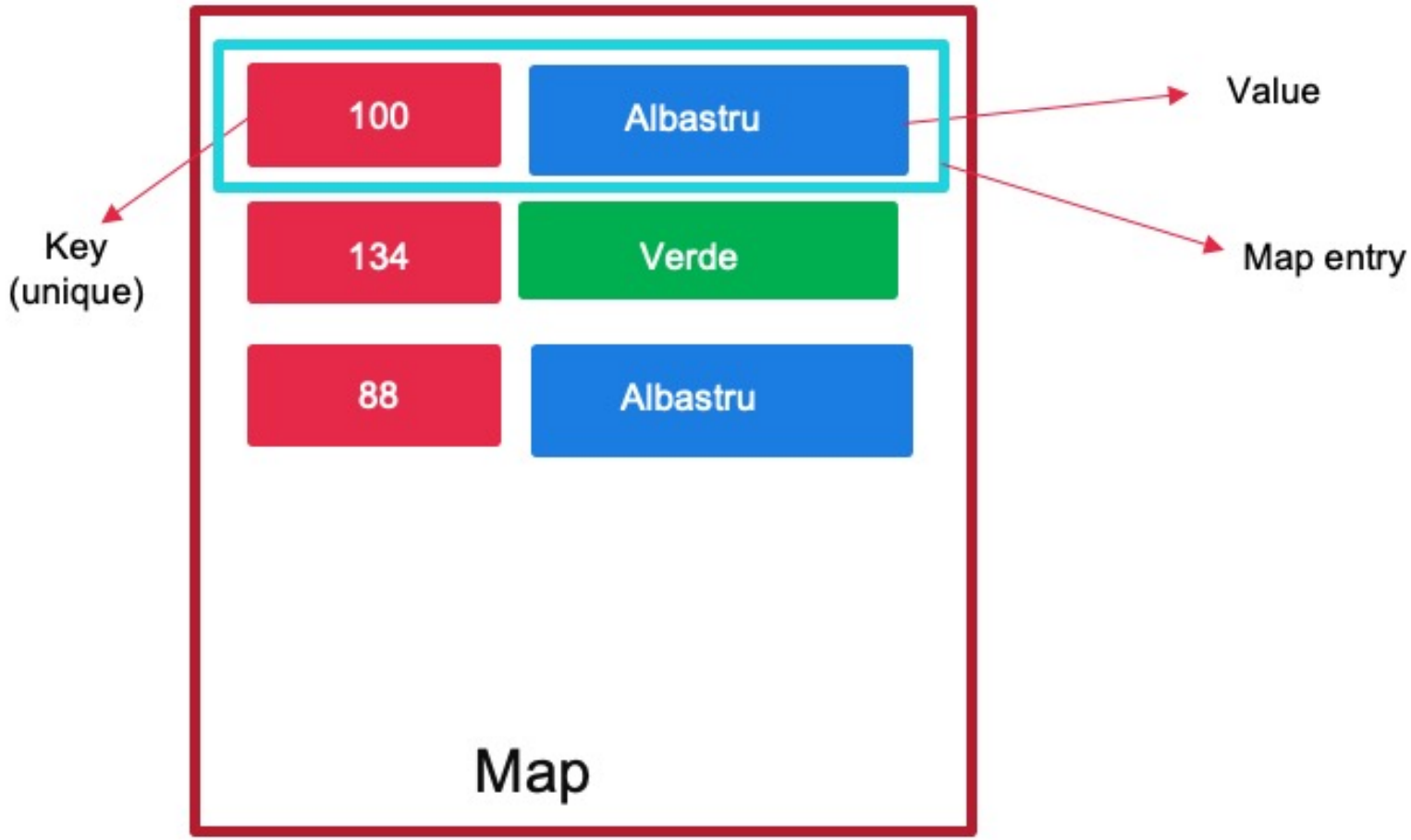
Metoda	descriere
add(Object obj)	începe să adauge elementul specificat de la locația zero
add(int index, Object obj)	inserează elementul specificat într-o anumită poziție din listă specificata de index
addAll(Collection c)	este utilizată pentru a adăuga un grup de elemente la sfârșitul ultimului element
addAll(int index, Collection c)	este utilizată pentru a adăuga un grup de elemente într-o anumită poziție din listă
remove(int index)	este folosit pentru a elimina un element într-o poziție specificată din listă
get(int index)	este utilizată pentru a returna elementul / obiectul stocat la o poziție specificată din listă. Tipul de returnare a metodei get () este Object și tipul de intrare este int [indexul Listei]

List – Methods

Metoda	descriere
indexOf(Object obj)	este utilizata pentru a returna indexul unui anumit element
lastIndexOf(Object obj)	returnează indexul ultimei apariții a unui element specificat din listă
set(int index, Object o)	înlocuiește elementul existent în poziția specificată din listă cu un element specificat nou

Map

- Map în Java este un obiect container care stochează elemente sub formă de perechi de chei și valori.
- O cheie este un element unic (obiect) care servește ca „index” in Map.
- Elementul care este asociat cu o cheie se numește valoare. Într-un Map, atât cheile, cât și valorile trebuie să fie obiecte și nu pot fi tipuri primitive.
- Map nu poate avea chei duplicate. Fiecare cheie mapează o singură valoare. Acest tip de mapare se numește mapare unu-la-unu în java.


100
Albastru
134
Verde
88
Albastru
Value
Map entry
Key
(unique)
Map

Map

- Toate cheile trebuie să fie unice, dar valorile pot fi duplicate (adică aceeași valoare poate fi stocată pe mai multe chei diferite).
- După ce intrarea (perechi cheie / valoare) este stocată într-un Map, îi putem prelua (obține) valoarea folosind cheia sa.
- Map în Java este definită în pachetul java.util.Map. Face parte din cadrul collection framework, dar nu extinde interfața colecției.
- Definierea in pachetul java.util.Map:
  public interface Map <K, V>
- Exemplu de folosire:
- Map<String, String> ; Map<Integer, String>

Clase care implementeaza Map

- AbstractMap: Este o clasă abstractă care implementează interfața Map.
- Este clasa părinte a tuturor claselor concrete de implementare Map, cum ar fi HashMap, TreeMap și LinkedHashMap.
- Implementează toate metodele din interfața Map, cu excepția metodei entrySet ().
- EnumMap: Clasa EnumMap extinde AbstractMap și implementează interfața Map. Este special utilizat pentru obiectele de tip Enum.

Clase care implementeaza Map

- HashMap: Este o clasă concretă care extinde AbstractMap.
- Folosește un hash table pentru a stoca elemente. Este utilizat în principal pentru localizarea unei valori, inserarea și ștergerea unei intrări.
- TreeMap: Este o clasă concretă care extinde AbstractMap și implementează interfața NavigableMap.
- Este folosita pentru trecerea cheilor în ordine sortată.
- Cheile pot fi sortate folosind interfața Comparable sau interfața Comparator.

Clase care implementeaza Map

- LinkedHashMap: Extinde clasa HashMap cu o implementare de listă legată (linked-list)care acceptă ordinea de inserare a intrărilor In Map.
- Intrările într-un HashMap nu sunt ordonate, dar intrările într-un LinkedHashMap pot fi recuperate în ordinea în care au fost inserate in Map.
- IdentityHashMap: Această clasă extinde AbstractMap și folosește egalitatea de referință pentru compararea intrărilor. Această clasă nu este utilizată în scopuri generale

Map -Methdos

Metoda	descriere
put(key K, value V)	adăuga o intrare cu cheia și valoarea specificate
putAll (Map m)	adăuga toate intrările din Map –ul specificat
putIfAbsent (key K, value V)	adăuga valoarea specificată cu cheia specificată în hartă numai dacă nu este deja
remove (Object key)	șterge o intrare pentru cheia specificată
remove(Object key, Object value)	elimina din Map valoarea specificată asociată cu cheia specificată
Set<K> keySet()	returnează un set format din cheile din Map
clear()	elimina toate intrările din Map

Map -Methdos

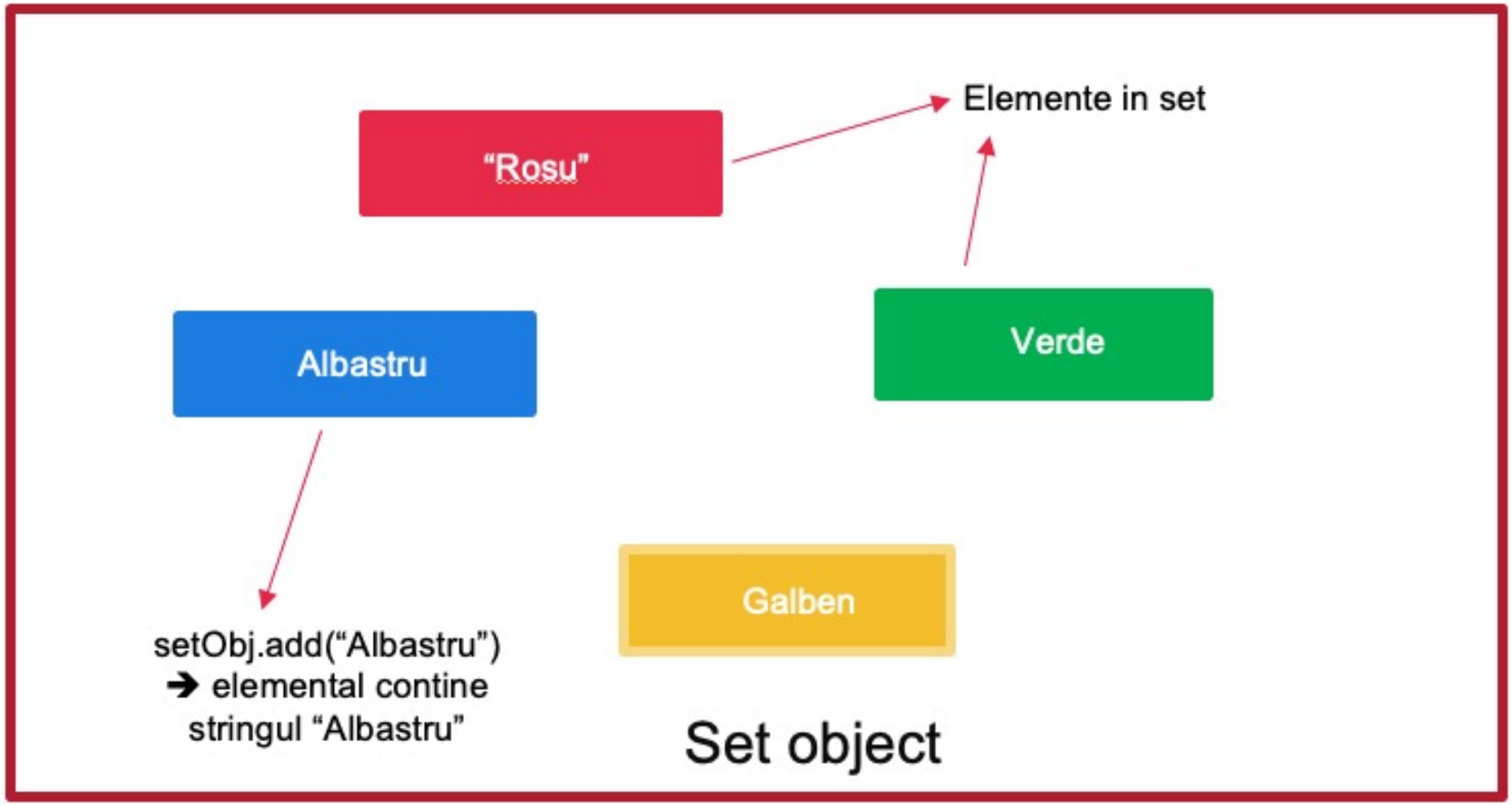
Metoda	descriere
get(Object key)	returnează valoarea pentru cheia specificată
hashCode()	returnează valoarea hash code
isEmpty()	verifica dacă Map-ul conține intrări
size()	returnează numărul de intrări (numărul de perechi cheie / valoare)
replace(K key, V value)	este utilizată pentru a înlocui valoarea veche cu o valoare nouă pentru o cheie specificată
containsKey(Object key)	este utilizată pentru a verifica dacă Map-ul conține o intrare pentru cheia specificată
containsValue(Object value)	este utilizată pentru a verifica dacă Map-ul conține o intrare pentru valoarea specificată
equals(Object obj)	este utilizată pentru a compara obiectul specificat

Set

- Un set în Java este o colecție neordonată de elemente sau obiecte unice. Cu alte cuvinte, un set este o colecție de elemente care nu sunt stocate într-o anumită ordine.
- Elementele sunt pur și simplu adăugate setului fără niciun control asupra locului în care se îndreaptă. De exemplu, în multe aplicații, nu trebuie să ne preocupăm de ordinea elementelor dintr-o colecție.
- Putem adăuga elemente într-un set și itera peste toate elementele dintr-un set. Va crește dinamic atunci când elementele sunt stocate în el. De asemenea, putem verifica dacă un anumit element este prezent în set.

Set

- Un set nu va permite elemente duplicate. Asta înseamnă că un element poate exista o singură dată în set. Dacă încercăm să adăugăm același element care este deja prezent în set, atunci acesta nu este stocat în set. De aceea, fiecare element dintr-un set este unic.
- Set în Java poate fi folosit pentru a elimina elementele duplicate din colecție.
- Setul deține o singură referință la un obiect. Nu oferă două referințe la același obiect, două referințe catre null sau referințe la două obiecte a și b astfel încât a.equals (b).


Elemente in set
“Rosu”
Albastru
Verde
Galben
setObj.add(“Albastru”)
➔ elemental contine
stringul “Albastru”
Set object

Set

- Set este o interfață introdusă în versiunea Java 1.2. Este o interfață generică care este declarată ca:
- Interface Set<data type>
- Interfața Java Set nu oferă metode proprii suplimentare. Folosește metode definite de collection framework, dar impune restricții suplimentare acestor metode.
- De exemplu, atunci când setul folosește metodele add () sau addAll () definite de collection framework, acesta nu adaugă un element setului dacă setul conține deja acel element.

Set

- Interfața Java Set nu oferă nicio metodă get (), pentru a prelua elementele.
- Prin urmare, singura modalitate de a scoate elemente din set este de a face folosind metoda Iterator (). Dar atenție, această metodă nu returnează elemente din set într-o anumită ordine.
- Deasemenea, folosind Iteratorul, putem parcurge doar în direcția înainte de la primul la ultimul element.
- Nu putem parcurge elemente în direcția inversă folosind metoda iteratorului.

Set –notiuni generale

- 1. Set - este o colecție neordonată de elemente. Asta înseamnă că ordinea nu este menținută în timpul stocării elementelor. În timp ce recuperăm, este posibil să nu primim aceeași ordine cu care am pus elemente.
- 2. Set - este folosit pentru a stoca o colecție de elemente fără duplicate. Asta înseamnă că poate conține doar elemente unice.
- 3. Poate fi iterat folosind Iterator, dar nu poate fi iterat folosind ListIterator.

Set –notiuni generale

- 4. Majoritatea implementărilor Set -ului permit adăugarea unui singur element null. TreeSet nu permite adăugarea niciunui element null.
- 5. Set nu este o structură bazată pe indexare, ca o listă în Java. Prin urmare, nu putem accesa elemente după poziția lor de index.

Set

- Java Collections oferă trei implementări generale de seturi:
  - HashSet
  - TreeSet
  - LinkedHashSet

Set -HashSet

- HashSet este o clasă concretă care implementează interfața setată. Folosește un mecanism de tip hash table pentru a-și stoca elementele. Este cea mai performantă implementare.

Set -TreeSet

- TreeSet este o clasă concretă care implementează interfața SortedSet (o subinterfață a setului).
- Folosește un mecanism binar de căutare pentru a-și stoca elementele.
- Își ordonează elementele pe baza valorilor lor. Este considerabil mai lent decât HashSet.

Set -LinkedHashSet

- LinkedHashSet este o clasă concretă care extinde clasa HashSet. Acesta își stochează elementele folosind mecanismul hash table cu o implementare a linked-list.
- Își ordonează elementele pe baza ordinii în care au fost inserate în set. Adică, elementele din HashSet nu sunt ordonate, dar elementele din LinkedHashSet pot fi recuperate în ordinea în care au fost inserate în set.

Set -Methods

Method name	Description	type
add(Object obj)	adăuga elementul specificat în set	boolean
addAll(Collection c)	adaugă toate elementele din colecția dată	boolean
size()	obține numărul de elemente din set	int
isEmpty()	verifică dacă setul este gol sau nu	boolean
clear()	elimina toate elementele din set	void
contains(Object o)	returnează adevărat dacă acest set conține elementul dat	boolean

Set -Methods

Method name	Description	type
containsAll(Collection c)	returnează adevărat dacă acest set conține toate elementele colecției date	boolean
remove(Object o)	elimina elementul specificat din set	boolean
removeAll(Collection c)	elimină din set toate elementele din colecția dată	boolean
equals(Object o)	se folosește pentru a compara elementul dat	boolean
iterator()	returnează un Iterator peste elementele din acest set	Iterator
hashCode()	folosita pentru a obține valoarea hashCode pentru acest set	int