Documentazione MapAdapterTest

Questa suite ha l'obiettivo di testare tutti i metodi di MapAdapter, alla ricerca di errori per verificarne l'assenza. I metodi vengono testati con variabili di diverso tipo e parametri validi e non, allo scopo di controllare il funzionamento delle eccezioni.

Pre condizioni generali:

Prima di ogni test, viene creata una mappa vuota map1 con il costruttore di default, a cui sono stati aggiunte coppie chiavi-valori di vario tipo, {0, (double) 3.3333333333333335}, {1, (int) -5}, {2,(char) L}, {3, (string) Test}, {4,(int) 7}, {5, (double) 3.3333333333333335}, {6, (boolean) true}.
Versione JUnit utilizzata per eseguire tutti i test: junit-4.13
Componenti del frame work utilizzati: assertEquals, assertNotEquals, assertFalse, assertNotNull, assertNull, assertTrue.

0. b4()

- Riassunto: @Before tramite il quale viene riempita la mappa map1, per essere poi utilizzata.
- Design: vengono aggiunti elementi alla mappa coppie di chiavi-valori di vario tipo.
- Descrizione: vengono aggiunte coppie chiavi-valori di vario tipo, {0, (double) 3.3333333333333335}, {1, (int) -5}, {2,(char) L}, {3, (string) Test}, {4,(int) 7}, {5, (double) 3.3333333333333335}, {6, (boolean) true}.
- ÏPre condizioni: mappa vuota map1.
- Post condizioni: mappa map1 piena.
- Risultato atteso: mappa map1 piena.

1. ConstructorNotNull()

- Riassunto: test del costruttore mapAdapter() per verificare che l'oggetto mapAdapter creato non sia nullo.
- Design: viene creata la mappa e verificato che non sia nulla.
- Descrizione: viene creata una nuova mappa map2 con il costruttore di default e viene verificato che non sia nulla.
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: map2 non nulla.

2. Constructor2NotNull()

- Riassunto: test del costruttore MapAdapter(MapAdapter map) per verificare che l'oggetto mapAdapter creato non sia nullo.
- Design: viene creata la mappa e verificato che non sia nulla.
- Descrizione: viene creata una nuova mappa map2 con il costruttore MapAdapter(MapAdapter map) e map1 come parametro e viene verificato che non sia nulla.
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: map2 non nulla.

3. clear()

- Riassunto: test del metodo clear().
- Design: viene svuotata la mappa map1 e viene verificato che la sua dimensione sia 0.

- Descrizione: viene svuotata la lista lst1 e viene verificato che la sua dimensione sia 0.
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: lista lst1 vuota, quindi con dimensione uguale a 0.

4. containsKeyNullPointer()

- Riassunto: test dell'eccezione NullPointer del metodo containsKey(Object key).
- Design: viene passato un null come parametro al metodo containsKey(Object key).
- Descrizione: viene passato un null come parametro al metodo containsKey(Object key)
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: eccezione NullPointer.

5. containsKey()

- Riassunto: test del metodo containsKey(Object key).
- Design: vengono passate delle chiavi a containsKey(Object key).
- Descrizione: vengono passate delle chiavi al metodo containsKey(Object key) applicato su map1, e viene verificato che il metodo ritorni true solo per le chiavi presenti nella mappa.
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: il metodo containsKey() ritorna true con le chiavi effettivamente presenti e false con le chiavi non facenti parte della mappa.

6. containsValueNullPointer()

- Riassunto: test dell'eccezione NullPointer del metodo containsValue(Object value).
- Design: viene passato un null come parametro al metodo containsValue(Object value).
- Descrizione: viene passato un null come parametro al metodo containsValue(Object value)
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: eccezione NullPointer.

7. containsValue()

- Riassunto: test del metodo containsValue(Object value).
- Design: vengono passate dei valori a containsValue(Object value).
- Descrizione: vengono passate dei valori al metodo containsValue(Object value) applicato su map1, e viene verificato che il metodo ritorni true solo per i valori presenti nella mappa.
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: il metodo containsValue() ritorna true con i valori effettivamente presenti e false con i valori non facenti parte della mappa.

8. entrySet()

- Riassunto: test del metodo entrySet().
- Design: viene creata una lista di EntrySet, che poi vengono confrontati.
- Descrizione: viene creata una lista che viene riempita con i MapEntry di map1. Poi vengono creati un array e un iteratore della lista che vengono usati per confrontare i MapEntry.
- Pre condizioni: lista piena map1.

- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: le MapEntry restituite sono uguali a due a due.

9. equals()

- Riassunto: test del metodo equals().
- Design: viene testato il metodo equals() confrontando la mappa map1 con una mappa uguale.
- Descrizione: viene creata la mappa map2 a cui vengono aggiunte le stesse coppie chiavi-valori presenti in map1. Le due mappe vengono poi confrontate con il metodo equals().
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: il metodo restituisce true solo quando le mappe confrontate sono effettivamente identiche.

10. getNullPointer()

- Riassunto: test dell'eccezione NullPointer del metodo get(Object key).
- Design: viene passato un null come parametro al metodo get(Object key).
- Descrizione: viene passato un null come parametro al metodo get(Object key)
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: eccezione NullPointer.

11. get()

- Riassunto: test del metodo get(Object key).
- Design: viene verificato il contenuto della mappa con il metodo get(Object key).
- Descrizione: vengono passate tutte le chiavi di map1 al metodo get(Object key) applicato su map1, e viene verificato che il metodo ritorni tutti i valori. Vengono anche passate chiavi non appartenenti a map1 per verificare che il metodo restituisca null.
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: il metodo restituisce i valori corretti.

12. hashCodeTest()

- Riassunto: test del metodo hashCode().
- Design: vengono confrontati i valori dell'hashcode restituito dall'omonimo metodo con dei valori calcolati esternamente.
- Descrizione: viene svuotata la mappa map1. Viene aggiunto un elemento e viene confrontato l'hashCode() restituito.
- Pre condizioni: lista piena lst1.
- Post condizioni: lista lst1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: i valori restituiti dal metodo coincidono con quelli previsti.

13. isEmpty()

- Riassunto: test del metodo isEmpty().
- Design: viene verificato con il metodo isEmpty() che la mappa sia vuota dopo averla svuotata e dopo aver aggiunto una coppia chiave-valore.
- Descrizione: dopo aver verificato che map1 non sia vuota, viene svuotata e viene controllato di nuovo se sia vuota o meno. Dopodiché viene aggiunto un elemento e verificato di
- Pre condizioni: lista piena map1.

- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: il metodo restituisce true se la mappa è effettivamente vuota, e false altrimenti.

14. keyset()

- Riassunto: test del metodo keySet().
- Design: viene verificato che il metodo keySet() restituisca un set contenente tutte le chiavi della mappa.
- Descrizione: viene creato un set set1 contenente le chiavi di map1. Dopodiché viene creato un secondo set set2 su in cui vengono salvate le chiavi di map1 tramite il metodo keySet(). Viene poi verificato che set2 contenga tutti gli elementi di set1 e viceversa, e che abbiano la stessa dimensione.
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: il metodo restituisce tutte e sole le chiavi della mappa.

15. putNullPointer()

- Riassunto: test dell'eccezione NullPointer del metodo put(Object key, Object value).
- Design: viene passato un null come parametro al metodo put(Object key, Object value).
- Descrizione: viene passato un valore null come value al metodo put(Object key, Object value).
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: eccezione NullPointer.

16. putNullPointer2()

- Riassunto: test dell'eccezione NullPointer del metodo put(Object key, Object value).
- Design: viene passato un null come parametro al metodo put(Object key, Object value).
- Descrizione: viene passato un valore null come key al metodo put(Object key, Object value).
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: eccezione NullPointer.

17. put()

- Riassunto: test del metodo put(Object key, Object value).
- Design: viene sostituito un valore della mappa con il metodo put(Object key, Object value).
- Descrizione: viene sostituito un valore di map1 e ne viene verificata l'avvenuta sostituzione.
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: il metodo sostituisce il valore corretto.

18. putAllNullPointer()

- Riassunto: test dell'eccezione NullPointer del metodo putAll(HMap t).
- Design: viene passato un null come parametro al metodo putAll(HMap t).
- Descrizione: viene passata un valore null come parametro al metodo putAll(HMap t).
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: eccezione NullPointer.

19. putAll()

- Riassunto: test del metodo putAll(HMap t).
- Design: vengono aggiunti tutti i valori di una mappa ad una mappa vuota. Successivamente vengono confrontate.
- Descrizione: dopo aver creato una nuova mappa map2, le vengono aggiunte tutte le coppie chiavi-valori di map1. Il contenuto delle due mappe viene poi confrontato con un ciclo for.
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: il metodo aggiunge tutte le coppie chiavi-valori della mappa nell'ordine corretto.

20. removeNullPointer()

- Riassunto: test dell'eccezione NullPointer del metodo remove(Object key).
- Design: viene passato un null come parametro al metodo remove(Object key).
- Descrizione: viene passata un valore null come parametro al metodo remove(Object key).
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: eccezione NullPointer.

21. remove()

- Riassunto: test del metodo remove(Object key).
- Design: vengono rimossi tutti i valori di una mappa con il metodo remove(Object key).
- Descrizione: vengono rimossi tutti i valori di map1 con il metodo remove(Object key). Man mano che vengono tolti i valori viene verificato che siano stati tolti i valori corretti. Viene infine verificata la dimensione della mappa.
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: il metodo rimuove tutte le coppie chiavi-valori della mappa correttamente.

22. size()

- Riassunto: test del metodo size().
- Design: viene verificata la dimensione con il metodo size() di una mappa dopo aver rimosso e aggiunto valori.
- Descrizione: viene verificata la dimensione di map1 con il metodo size() di una mappa dopo aver rimosso una coppia di chiave-valore e dopo averne aggiunte due.
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: il misura la dimensione della mappa correttamente.

23. values()

- Riassunto: test del metodo values().
- Design: viene verificato che il metodo velues() restituisca una collection contenente tutte i valori della mappa.
- Descrizione: viene creato un ListAdapter contenente tutti i valori della mappa utilizzando il metodo values(). Poi viene creato un altro ListAdapter lst1 in cui vengono aggiunti tutti i valori della mappa.
- Pre condizioni: lista piena map1. Viene poi verificato che lst2 contenga tutti gli elementi di lst1 e viceversa, e che abbiano la stessa dimensione.
- Pre condizioni: lista piena map1.
-

- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: il metodo restituisce tutti e soli i valori della mappa.

24. mapEntryConstructorNotNull()

- Riassunto: test del costruttore mapEntry() per verificare che l'oggetto MapEntry creato non sia nullo.
- Design: viene creato il MapEntry e verificato che non sia nullo.
- Descrizione: viene creato una nuovo set di MapEntry set con il costruttore metodo mapEntry() e viene verificato che non sia nullo.
- Pre condizioni: lista piena map1.
- Post condizioni: lista map1 torna alla situazione iniziale (vedasi pre condizioni).
- Risultato atteso: set non nullo.