TEST SUITE MAPADAPTERTEST

La seguente test suit ha lo scopo di testare tutti i metodi presenti nella classe MapAdapter.java con lo scopo di dimostrarne il loro corretto funzionamento.

I metodi (quando previsto il possibile problema) sono stati testati in maniera duale, sia con valori validi sia con valori non validi con due test distinti all'interno della stessa pagina. La differenza tra i due è subito chiarita dall'aggiunta dell'attributo Exception al nome del metodo di testing. I test con i valori errati o non ammessi verificano il corretto lancio dell'eccezione

Nei test sarà sempre disponibile un oggetto di tipo MapAdapter che sarà usato in tutti i test . Tale oggetto viene sempre creato prima dell'esecuzione di ogni test. La mappa conterrà 20 entry, con chiavi intere che vanno da 0 a 19 e valori interi che vanno da 0 a 19 (Pre-condizione)

Terminato ogni test l'oggetto di tipo MapAdapter tornerà a contenere solo le 20 entry "di base" in modo da rendere ogni test indipendente dall'altro

Tutti i test seguono la struttura con cui è stato pensato il contenitore, ossia che può accettare qualunque tipo/oggetto, compresi tipi differenti es: String, Integer… contemporaneamente ad eccezione dei valori null che non sono ammessi

La descrizione dei test segue l'ordine del file MapAdapterTest.java

clearTest()

- test metodo void clear()
- Descrizione: verifica dimensione della mappa che dovrà essere pari a 20. Svuotamento della mappa e verifica del cambiamento della dimensione che passa da 20 a 0
- Sommario e design: eliminazioni di tutto il contenuto della mappa
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: la mappa dovrà svuotarsi e la sua dimensione essere pari a zero

sizeTest()

- Test del metodo int size()
- Descrizione: verifica dimensione della mappa che deve essere pari a 20, aggiunta di una entry(21,21) e verifica aumento dimensione mappa di 1. Svuotamento della mappa e verifica del cambiamento della dimensione, aggiunta di una nuova entry (1,1) alla mappa vuota e verifica aumento dimensione di 1
- Sommario e design: controllo dimensione mappa con rimozione e aggiunta di oggetti
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà la dimensione della mappa

containsKeyTest()

- test metodo boolean containsKey(object o)
- Descrizione: verifica presenza di una chiave(2) basata sulla conoscenza delle entry della mappa, verifica della non presenza di una chiave. Aggiunta di una nuova entry alla mappa(21,21) e verifica della presenza della chiave. Azzeramento della mappa e verifica della non presenza di una qualunque chiave.
- Sommario e design: verifica presenza di una chiave
- Pre-condizioni: mappa adatta contenente 20 elementi e chiave valida
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà true solo se la chiave è presente, altrimenti false

containsKeyException()

- Test lancio eccezione di boolean containsKey(Object o)
- Descrizione: ricerca di una chiave non valida(null) all'interno della mappa sia nel caso con 20 entry sia quando azzerata
- Sommario e design: lancio eccezione se passata chiave non valida
- Pre-condizioni: mappa adatta contenente 20 elementi e chiave non valida
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà l'eccezione NullPointerException

containsValueTest()

- test metodo boolean containsValue(object o)
- Descrizione: verifica presenza del valore(2) basata sulla conoscenza delle entry della mappa, verifica della non presenza del valore. Aggiunta di una nuova entry alla mappa(21,21) e verifica della presenza del valore. Azzeramento della mappa e verifica della non presenza di un qualunque valore.
- Sommario e design: verifica presenza di una cvalore
- Pre-condizioni: mappa adatta contenente 20 elementi e valore valido
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà true solo se il valore è presente, altrimenti false

containsKeyException()

- Test lancio eccezione di boolean containsValue(Object o)
- Descrizione: ricerca di un valore non valido(null) all'interno della mappa sia nel caso con 20 entry sia quando azzerata
- Sommario e design: lancio eccezione se passato valore non valido
- Pre-condizioni: mappa adatta contenente 20 elementi e valore non valido
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà l'eccezione NullPointerException

entrySetTest()

- Test metodo HSet entrySet()
- Descrizione: creazione di un set contenente le entry della mappa, verifica uguaglianza dimensione del set con la mappa, aggiunta di una entry nella mappa e verifica aumento di 1 sia della mappa che del set. Rimozione di due elementi dalla mappa (20,21) e verifica dell'assenza nel set
- Sommario e design: creazione di un set contenente le entry della mappa
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà un set contenente tutte le entry della mappa. Ogni cambiamento della mappa si rifletterà sul set e viceversa, ad eccezione delle aggiunte di elementi del set che è una operazione non prevista

equalsTest()

- Test del metodo boolean equals(Object o)
- Descrizione: creazione di una nuova mappa contenente le stesse entry della mappa di base, verifica dell'uguaglianza delle dimensioni tra le due mappe. In fine verifica della presenza delle stesse chiavi e valori in entrambe le mappe
- Sommario e design: confronto uguaglianza tra due mappe
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà true se le due mappe combaciano perfettamente, stessa dimensione e stesse entry, altrimenti false

getTest(Object key)

- Test metodo object get(Object key)
- Descrizione: verifico che ad una data chiave corrisponda un determinato oggetto(2 ,2). Come controprova verifico che ad una data chiave non corrisponda un determinato oggetto(12,1). In fine rimuovo una entry e verifico che comunque l'ultima entry sia ancora in ultima posizione e che data la chive in size() corrisponda l'oggetto presente in tale entry
- Sommario e design: ritorno del valore data la chiave corrispondente
- Pre-condizioni: mappa adatta contenente 20 elementi e valore valido
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà il valore corrispondente a tale chiave

getExceptionTest()

- Test eccezione del metodo object get(object key)
- Descrizione: ricerca di una chiave non valida(null) all'interno della mappa sia nel caso con 20 entry sia quando azzerata
- Sommario e design: lancio eccezione se passata chiave non valida
- Pre-condizioni: mappa adatta contenente 20 elementi e chiave non valida
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà l'eccezione NullPointerException

hashCodeTest()

- Test del metodo int hashCode()
- Descrizione: creazione di una nuova mappa identica a quella di base, verifica che i due hashCode siano identici. Aggiunta di un elemento nella nuova mappa e verifica che i due hashCode siano differenti
- Sommario e design: calcolo valore hashCode della mappa
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà il valore dell'hashCode della mappa

isEmptyTest()

- Test metodo boolean isEmpty()
- Descrizione: verifica che la mappa abbia dimensione pari a 20, verifica che la mappa non sia vuota. Azzeramento della mappa e verifica che la mappa ora sia vuota. Aggiunta di una nuova entry nella mappa (0,0), verifica che la mappa non sia più vuota e in fine rimozione dell' unica entry presente e verifica della dimensione pari a 0
- Sommario e design: ritorna True se la mappa è vuota
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà true solo se la mappa sarà vuota altrimenti false

keySetTest()

- Test metodo HSet keySet()
- Descrizione: creazione di un set contenente le chiavi della mappa, verifica dell'uguaglianza delle dimensioni tra il set e la mappa. Verifica che tutte le chiavi contenenti nel set siano le stesse della mappa. Aggiunta di un elemento alla mappa e verifica della presenza nel set di quest'ultimo
- Sommario e design: creazione di un set contenente le chiavi della mappa
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà un set contenente tutte le chiavi della mappa. Ogni cambiamento della mappa si rifletterà sul set e viceversa, ad eccezione delle aggiunte di elementi del set che è una operazione non prevista

putTest()

- Test metodo object put(object key, object value)
- Descrizione: verifica che la mappa abbia dimensione 20. Inserimento di un entry(21, “ciao”). Verifica aumento di 1 dimensione mappa, verifica che la mappa contenga la chiave 21 e il valore “ciao”
- Sommario e design: inserimento di un entry nella mappa
- Pre-condizioni: mappa adatta contenente 20 elementi, chiavi e valori non nulli
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo aggiungerà alla mappa la entry e ritornerà il precedente valore associato a quella chiave se presente

putExceptionTest()

- Test lancio eccezione di object put(Object key, object value)
- Descrizione: verifica del lancio dell' eccezione in seguito all'inserimento di un entry con chiave valida e valore nullo, chiave nulla e valore nullo, chiave nulla e valore valido
- Sommario e design : lancio eccezione con chiavi e valori nulli
- Pre-condizioni: mappa adatta contenente 20 elementi, chiavi e valori nulli
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: lancio di NullPointerException

putAllTest()

- Test metodo void putAll(HMap m)
- Descrizione: creazione di una nuova mappa avente dimensione zero, verifica che la nuova mappa e la mappa di base abbiano dimensioni differenti. Inserimento di tutte le entry della mappa nella nuova mappa, verifica che la nuova mappa sia uguale a quella di base
- Sommario e design: inserimento di tutte le entry di una mappa in un'altra
- Pre-condizioni: mappa adatta contenente 20 elementi e mappa non nulla
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: tutte le entry della mappa saranno inserite(come copia) nella mappa scelta come destinazione

putAllExceptionTest()

- Test lancio eccezione di void putAll(HMap m)
- Descrizione: verifica del lancio dell' eccezione in seguito al passaggio come argomento di una mappa nulla
- Sommario e design: lancio eccezioni avendo mappa nulla
- Pre-condizioni: mappa adatta contenente 20 elementi e mappa nulla
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà l'eccezione NullPointerException

removeTest()

- Test del metodo object remove(Object key)
- Descrizione: verifica che la mappa dimensione uguale a 20, rimozione di una chiave(19) dalla mappa. Verifica che la dimensione sia calata di 1 e che la mappa non contenga la chiave rimossa
- Sommario e design: rimozione di una chiave e del relativo valore associato dalla mappa
- Pre-condizioni: mappa adatta contenente 20 elementi e chiave non nulla
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo rimuoverà la coppia chiave valore associata alla chiave e ritornerà il valore eliminato o null se la chiave non era presente

removeExceptionTest()

- Test del lancio dell'eccezione di object remove(Object key)
- Descrizione: verifica lancio eccezione in corrispondenza della richiesta di rimuovere una chiave con valore null
- Sommario e design: lancio dell'eccezione avendo chiave nulla
- Pre-condizioni: mappa adatta contenente 20 elementi e chiave nulla
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: lancio di NullPointerException

valuesTest()

- Test del metodo HCollection values()
- Descrizione: creazione di una collection contenente i valori della mappa, verifica dell' uguaglianza della dimensione tra la mappa e la collection. Aggiunta di un elemento alla mappa(21,21) e verifica aggiornamento delle dimensioni che aumenteranno di 1. In fine verifica della presenza dell' elemento aggiunto alla mappa nella collection
- Sommario e design: creazione di una collection contenente i valori della mappa
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà una collection contenente tutti i valori presenti nella mappa

TEST SUITE ENTRYADAPTERTEST.JAVA

La seguente test suit ha lo scopo di testare tutti i metodi presenti nella classe entryAdapterTest.java con lo scopo di dimostrarne il loro corretto funzionamento. Tale classe è presente all'interno della classe MapAdapter.java

La struttura dei test segue quella di MapAdapterTest.java

Nei test sarà sempre disponibile un oggetto di tipo MapAdapter che sarà usato in tutti i test . Tale oggetto viene sempre creato prima dell'esecuzione di ogni test. La mappa conterrà 20 entry, con chiavi intere che vanno da 0 a 19 e valori interi che vanno da 0 a 19 (Pre-condizione)

Terminato ogni test l'oggetto di tipo MapAdapter tornerà a contenere solo le 20 entry "di base" in modo da rendere ogni test indipendente dall'altro

Tutti i test seguono la struttura con cui è stato pensato il contenitore, ossia che può accettare qualunque tipo/oggetto, compresi tipi differenti es: String, Integer… contemporaneamente ad eccezione dei valori null che non sono ammessi

La descrizione dei test segue l'ordine del file EntryAdapterTest.java

getKeyEntryTest()

- Test del metodo object getKey()
- Descrizione: creazione di un set contenente le entry della mappa, verifica uguaglianza dimensione set con la mappa. Creazione di un interatore per il set, fino a che l'iteratore non ha attraversato tutto il set salvo una copia dell'entry ritornata da next() e controllo che la mappa contenga tale entry, in particolare la chiave
- Sommario e design: ritorna la chiave corrispondente alla entry
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà la chiave corrispondente alla entry

getValueEntryTest()

- Test del metodo object getValue()
- Descrizione: creazione di un set contenente le entry della mappa, verifica uguaglianza dimensione set con la mappa. Creazione di un interatore per il set, fino a che l'iteratore non ha attraversato tutto il set salvo una copia dell'entry ritornata da next() e controllo che la mappa contenga una tale entry, in particolare in valore
- Sommario e design: ritorna il valore corrispondente alla entry
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà il valore corrispondente alla entry

Test setValue()

- Test object setValue(Object o)
- Descrizione: creazioni di un set con tutte le entry della mappa, verifica uguaglianza dimensione set e mappa. Creazione di un iteratore del set, salvataggio dell'elemento ritornato da next() e modifica del valore in tale entry. Verifica che la mappa contenga tale valore.
- Sommario e design: sostituzione del valore dell'entry con uno non nullo
- Pre-condizioni: mappa adatta contenente 20 elementi e valore non nulla
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà il valore dell' entry sostituita

Test setValueEntryExceptionTest()

- Test eccezione lanciata da setValue(Object o)
- Descrizione: creazioni di un set con tutte le entry della mappa. Creazione di un iteratore del set, salvataggio dell'elemento ritornato da next(). Verifica del lancio dell'eccezione a seguito dell'inserimento del valore null
- Sommario e design: lancio eccezione a seguito dell'inserimento del valore null
- Pre-condizioni: mappa adatta contenente 20 elementi e valore nulla
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà NullPointerException

euqualsEntryTest()

- Test del metodo boolean equals(Object o)
- Descrizione: creazione di un set contenente le entry della lista, verifica dell'uguaglianza della dimensione tra il set e la mappa. Creazione di un iteratore del set e di un altro set contente le entry della mappa. Verifico uguaglianza delle dimensioni tra i due set e se in generale i due set contengono le stesse entry
- Sommario e design: controlla se le entry sono uguali
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà true se le entry sono uguali

hashCodeEntryTest()

- Test metodo int hashCode()
- Descrizione: creazione di un set contenente tutte le entry della mappa, verifica uguaglianza delle dimensioni. Creazione di una nuova mappa contenente le stesse entry della mappa di base e di un set contenente le entry di tale mappa. Verifica che l' hashCode del primo set sia uguale a quella del secondo e che in seguito all'aggiunta di un'entry in uno dei due set(21,"rompo tutto") l'hashCode cambi
- Sommario e design: valore dell'hashCode di tutte le entry
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà il valore dell'hashCode

TEST SUITE SETADAPTERTEST

La seguente test suit ha lo scopo di testare tutti i metodi presenti nella classe SetAdapter.java con lo scopo di dimostrarne il loro corretto funzionamento.

Tale classe è presente all'interno della classe MapAdapter.java.
La struttura dei test segue quella di MapAdapterTest.java

Nei test sarà sempre disponibile un oggetto di tipo MapAdapter che sarà usato in tutti i test . Tale oggetto viene sempre creato prima dell'esecuzione di ogni test. La mappa conterrà 20 entry, con chiavi intere che vanno da 0 a 19 e valori interi che vanno da 0 a 19 (Pre-condizione)

Terminato ogni test l'oggetto di tipo MapAdapter tornerà a contenere solo le 20 entry "di base" in modo da rendere ogni test indipendente dall'altro.

Il test va a dimostrare inoltre che ogni azione, modifica del set si rifletterà sulla mappa e viceversa. Non sono mai previste e ammesse operazione di aggiunta di elementi direttamente nel set, bisognerà sempre passare per la mappa correlata

Nella seguente classe verrano testati 2 tipo di set e una collection (implementata come set): il primo contenente le entry della mappa, il secondo le chiavi il terzo i valori. Alcuni metodi del set di chiavi e di valori sono identici per tanto verrà usato un tipo di set qualsiasi tra i due. Sarà indicato quando un test è applicabile solo ad un determinato set. Alcuni metodi del set di entry saranno ereditati dai set di valori e chiavi, pertanto ci si limita al solo test applicato al set di enty

Tutti i test seguono la struttura con cui è stato pensato il contenitore, ossia che può accettare qualunque tipo/oggetto, compresi tipi differenti es: String, Integer… contemporaneamente ad eccezione dei valori null che non sono ammessi

La descrizione dei test segue l'ordine del file SetAdapterTest.java

clearSetEntryTest()

- Test metodo void clear()
- Descrizione: creazione di un set contenente tutte le entry della mappa, verifica uguaglianza della dimensione tra la mappa e il set. Azzeramento del set e verifica che la mappa e il set abbiano la stessa dimensione pari a 0
- Sommario e design: azzeramento del set
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo renderà la dimensione del set pari a zero (anche quella della mappa)

isEmptySetEntryTest()

- Test metodo boolean isEmpty()
- Descrizione: creazione di un set di entry dalla mappa, verifica dell'uguaglianza della dimensione tra il set e la mappa, azzeramento della mappa e verifica che la dimensione della mappa e del set sia pari a zero
- Sommario e design: ritorna true se il set è vuoto
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il ritornerà true se il set ha dimensione pari a zero, altrimenti false.

retainAllSetEntryTest()

- Test metodo boolean retainAll(HCollection c)
- Descrizione: creazione di un set di entry contenente le entry della mappa e di una nuova mappa contenente gli stesse elementi della mappa di base. Creazione di un altro set di entry della nuova mappa, inserimento di due entry(20,20)(21,21) nella nuova mappa e verifica che le dimensioni dei due set siano differenti. Eliminazione di tutti gli elementi del set creato dalla mappa di base che non sono presenti nel set della mappa nuova. verifica che la mappa di base e il set correlato non contengano le due entry aggiunte precedentemente
- Sommario e design: eliminazione di tutte le entry non presenti nella collection
- Pre-condizioni: mappa adatta contenente 20 elementi e collection valida
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il ritornerà true se l'eliminazione è avvenuta data la presenza di elementi differenti, altrimenti false

retainAllSetEntryException()

- test eccezione lanciata da boolean retainAll(HCollection c)
- Descrizione: creazione di un set contenente le entry della mappa, verifica del lancio dell' eccezione dovuta alla collection nulla. Azzeramento del set Ehi verifica del lancio dell' eccezione dovuta alla collection nulla
- Sommario e design: lancio eccezione dovuta a collection nulla
- Pre-condizioni: mappa adatta contenente 20 elementi e collection non valida
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: lancio di NullPointerException

toArrayEntrySetTest()

- Test del metodo object[] toArray()
- Descrizione: creazione di un set contenente le entry della mappa e di un array contenente i valori del set. Verifica che la dimensione del set sia la stessa di quella dell' array e che il set contenga tutti gli elementi presenti nell' array
- Sommario e design: copia in un array di tutte le entry
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà un array contenente tutte le entry

toArrayAEntrySetTest()

- Test del metodo object[] toArray(object[]a)
- Descrizione: creazione di un set contenente le entry della mappa e di un array delle stesse dimensioni del set contenente dei valori qualunque. Verifica che la dimensione del set sia la stessa di quella dell' array e che il set contenga tutti gli elementi presenti nell' array
- Sommario e design: copia in un specifico array di tutte le entry
- Pre-condizioni: mappa adatta contenente 20 elementi e array non nullo
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà l' array contenente tutte le entry

toArrayExceptionEntrySetTest()

- Test lancio eccezione di toArray(Object[]a)
- Descrizione: creazione di un set contenente tutte le entry della mappa e di un array avente la stessa dimensione del set ma con un tipo specifico. Verifica lancio dell' eccezione a causa di un inserimento impossibile e dell' eccezione dovuta a un inserimento in un array null
- Sommario e design: lancio delle eccezioni relative
- Pre-condizioni: mappa adatta contenente 20 elementi e array nullo
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà l'eccezione ArrayStoreException e NullPointerException

containsEntrySetTest()

- Test del metodo boolean contains(Object o)
- Descrizione: creazione di un set contenente tutte le entry della mappa e di un iteratore del set. Scorrimento di tutto il set tramite l' iteratore e verifica della presenza degli elementi ritornati da next() nel set
- Sommario e design: verifica della presenza dell'entry
- Pre-condizioni: mappa adatta contenente 20 elementi ed entry non nullo
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo true se l'entry è presente, false altrimenti

containsEntrySetExceptionTest()

- Test lancio eccezione di contains(object o)
- Descrizione: creazione di set contenente le entry della mappa e verifica del lancio dell' eccezione relativo alla ricerca di un valore null
- Sommario e design: lancio eccezione dovuta alla ricerca di un valore non valido
- Pre-condizioni: mappa adatta contenente 20 elementi ed entry nulla
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà l'eccezione NullPointerException

containsAllEntrySetTest()

- Test metodo boolean containsAll(HCollection c)
- Descrizione: creazione di un set contenente tutte le entry della mappa e di una mappa vuota con il rispettivo set di entry. Inserimento nella mappa nuova degli stessi valori della mappa di base e verifica che il set della mappa di base contenga tutti gli elementi del nuovo set. Aggiunta di un elemento(10,"combino guai") alla mappa di base e verifica che il set base non contenga tutto il set della mappa nuova
- Sommario e design: verifica della presenza di tutti gli elementi di una collection non nulla in un set di entry
- Pre-condizioni: mappa adatta contenente 20 elementi e collection non nulla
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà true se il set contiene tutti gli elementi della collection, false altrimenti

containsAllEntrySetExceptionTest()

- Test lancio eccezione di boolean containsAll(Hcollection c)
- Descrizione: creazione di un set contenente tutte le entry della mappa e verifica lancio dell' eccezione dovuta alla ricerca di una collection null
- Sommario e design: lancio eccezione dovuta alla ricerca oggetto null
- Pre-condizioni: mappa adatta contenente 20 elementi e collection nulla
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà NullPointerException

hashCodeEntrySetTest()

- Test metodo int hashCode()
- Descrizione: creazione di un set contenente tutte le entry della mappa e di una nuova mappa che sarà riempita con gli stessi elementi della mappa di base. verifica che la hashCode del set di entry sia uguale all' hashCode restituito dal set di entry della nuova mappa
- Sommario e design: calcolo dell'hashCode dell'insieme di entry
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà l'hashCode dell'insieme di entry

removeEntrySetTest()

- Test metodo boolean remove(object o)
- Descrizione: creazione di un set contenente le entry della mappa, salvataggio dell'elemento ritornato dall'iteratore del set seguito dalla rimozione di tale elemento. Verifica che il set non contenga più l' elemento e che la dimensione sia calata di 1
- Sommario e design: rimozione di un elemento non nullo
- Pre-condizioni: mappa adatta contenente 20 elementi ed elemento non nullo
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà true in seguito alla rimozione dell'entry, altrimenti false

removeEntrySetException()

- Test lancio eccezione di boolean remove(Object o)
- Descrizione: creazione di un set contenente le entry della mappa e verifica del lancio dell' eccezione in corrispondenza dell'eliminazione di un oggetto null. Azzeramento del set e verifica del lancio dell' eccezione nel tentativo di rimozione di un oggetto null
- Sommario e design: lancio dell'eccezione dovuta alla rimozione ad un oggetto null
- Pre-condizioni: mappa adatta contenente 20 elementi e oggetto nullo
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà NullPointerException

removeAllEntrySetTest()

- Test del metodo booelan removeAll(HCollection)
- Descrizione: creazione di un set contenente le entry della mappa e di una nuova mappa contenente metà valori della mappa di base. Creazione di un altro set contenente l' entry della seconda mappa e rimozione dalla dal set di base di tutti gli elementi contenuti nel set della nuova mappa seguito dalla verifica sulla dimensione del set di base che deve essere uguale a 10 e dalla presenza di tutti gli elementi da 0 a 10
- Sommario e design: rimozione di una specifica collection di entry
- Pre-condizioni: mappa adatta contenente 20 elementi e collection non nulla
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà true a seguito dell'eliminazione della collection, false altrimenti

removeAllEntrySetExceptionTest()

- Test lancio eccezione del metodo removeAll(HCollection c)
- Descrizione: creazione di un set contenente le entry della mappa e verifica del lancio dell' eccezione nel tentativo di rimuovere una collection null. Azzeramento del set e verifica del lancio dell' eccezione nel tentativo di rimuovere una collection null
- Sommario e design: lancio eccezione dovuta alla rimozione di una collection null
- Pre-condizioni: mappa adatta contenente 20 elementi e collection nulla
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà NullPointerException

sizeEntrySetTest()

- Test del metodo int size()
- Descrizione: creazione di un set contenente le entry della mappa verifica dell'uguaglianza tra la dimensione della mappa e del set. Aggiunta di un elemento(20,20) alla mappa e verifica dell'aumento della dimensione nel set
- Sommario e design: ritorno della dimensione del set
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà la dimensione del set

entrySetIteratorTest()

- Test metodo HIterator iterator()
- Descrizione: creazione di un set contenente le entry della mappa e di un iteratore del set. Creazione di una variabile che conterrà il numero degli elementi scansionati dell' iteratore che dovrà essere uguale alla dimensione del set
- Sommario e design: creazione e uso di un iteratore per scorrere il set
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: creazione di un iteratore che è in grado di scansionare il set

equalsEntrySetTest()

- Test del metodo boolean equals()
- Descrizione: creazione di un set contenente lenti della mappa e di un'altra mappa contenente gli stessi valori della mappa di base. Creazione di un altro set contenente le entry della seconda mappa E verifica che le dimensioni dei set sono uguali. aggiunta di un elemento(20, “ti renderò falso”) alla mappa in modo tale da verificare che i due set non siano più uguali
- Sommario e design: verifica uguaglianza tra due set di entry
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà true se i due set sono uguali, altrimenti false

addAndAddAllEntrySetExceptionTest()

- Test lancio eccezione di add(object o) e addAll(HCollection c)
- Descrizione: creazione di un set contenente l' entry della mappa e verifica del lancio dell' eccezione al tentativo di aggiungere elementi al set. Creazione di una nuova mappa ed di un nuovo set ad esso correlato, aggiunti due elementi(20,20)(21,21) alla mappa e verifica del lancio dell' eccezione al tentativo di aggiungere nel primo set gli elementi del secondo set
- Sommario e design: lancio eccezione per operazione impossibile
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà UnsupportedOperationException

nextAndHasNextEntrySetIteratorTest()

- Test metodo object next() e boolean hasNext() di HIterator
- Descrizione: creazione di un set contenente le entry della mappa e di un iteratore del set. Che il set abbia la stessa dimensione della mappa. Scansione di tutto il set tramite l'iteratore e verifica che il set contenga ogni elemento ritornato da next() seguito dalla verifica che l' interatore abbia scorso tutto il set.
- Sommario e design: creazione di un iteratore che percorre tutta la lista e ritorna gli elementi scansionati
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà un iteratore per il set

nextEntrySetIteratorTest()

- Test lancio eccezione di object next() di HIterator
- Descrizione: creazione di un set contenente le entry della mappa e di un iteratore del set.
- Scorrimento di tutto il set tramite l'iteratore e verifica del lancio dell' eccezione in corrispondenza della richiesta di next() una volta finita la scansione del set
- Sommario e design: lancio eccezione dovuta alla mancanza dell'elemento da ritornare da parte dell'iteratore
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà NoSuchElementException

removeEntrySetIteratorTest()

- Test del metodo void remove() di HIterator
- Descrizione: creazione di un set contenente le entry della mappa e di un iteratore del set. Verifica della dimensione del set che deve essere pari a quella della mappa cioè 20. Salvataggio dell'oggetto ritornato da next() e successiva eliminazione di quest'ultimo. Verifica dell'assenza di tale oggetto dal set
- Sommario e design: rimozione dell'oggetto puntato dall'iteratore
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: eliminazione dell'oggetto puntato dall'iteratore nel set

removeEntrySetIteratorExceptionTest()

- Descrizione: creazione di un set contenente le entry della mappa e di un iteratore del set. Verifica lancio dell' eccezione in seguito alla rimozione in una posizione non valida da parte dell'iteratore
- Sommario e design: lancio eccezione in seguito alla rimozione di un oggetto tramite iteratore in una posizione non valida
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà l'eccezione IllegalStateException

containsSetKeyValueTest()

- Test metodo boolean contains(Object o)
- Descrizione: un set contenente le chiavi della mappa e un iteratore del set. Scansione di tutto il set e verifica della presenza delle chiavi tramite next(). Creazione di un nuovo set contenente i valori della mappa e un iteratore del set. Scansione di tutto il set e verifica della presenza dei valori tramite next()
- Sommario e design: controllo presenza valore in un set
- Pre-condizioni: mappa adatta contenente 20 elementi e valore valido
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo tornerà true se il set contiene il valore, false altrimenti

containsSetKeyValueExceptionTest()

- Test lancio eccezione del metodo boolean contains(object o)
- Descrizione: creazione di un set contenente le chiavi della mappa e di uno contenente i valori della mappa. Verifica del lancio dell' eccezione nel tentativo di cercare un valore o una chiave non valida
- Sommario e design: lancio eccezione data la ricerca di un valore non valido
- Pre-condizioni: mappa adatta contenente 20 elementi e valore non valido
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà NullPointerException

containsAllSeKeyValueTest()

- Test del metodo boolean containsAll(HCollection c)
- Descrizione: creazione di un set contenente i valori della mappa e di una nuova mappa contenente gli stessi valori della mappa di base. Creazione di un set di valori per la nuova mappa e verifica che il set di valori contenga tutti gli elementi presenti nel set della mappa nuova
- Sommario e design: controllo presenza collection di valori in un set
- Pre-condizioni: mappa adatta contenente 20 elementi e collection valida
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo tornerà true se il set contiene la collection, altrimenti false

containsAllSetKeyValueExceptionTest()

- Test lancio eccezione del metodo boolean containsAll(HCollection o)
- Descrizione: creazione di un set contenente i valori della mappa e di uno contenente le chiavi della mappa. Verifica del lancio dell' eccezione nel tentativo di cercare un valore o una chiave non valida
- Sommario e design: lancio eccezione data la ricerca di una collection non valida
- Pre-condizioni: mappa adatta contenente 20 elementi e collection non valida
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà NullPointerException

hashCodeSetKeyValueTest()

- Test metodo int hashCode()
- Descrizione: creazione di un set contenente le chiavi della mappa ed una nuova mappa con gli stessi valori di quella di base. Creazione di un nuovo set funzionante le chiavi della nuova mappa e verifica dell'uguaglianza degli hashCode dei due set. Aggiunta di un elemento(21,”rompo”) alla nuova mappa e verifica modifica dell’hashCode
- Sommario e design: calcolo valore dell’hashCode
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo tornerà in valore dell’hashCode

removeAllSetKeyValueTest()

- Test del metodo boolean removeAll(HCollection c)
- Descrizione: creazione di un set contenente i valori della mappa contenente metà valori della mappa di base(0,10). Creazione di un altro set contenente i valori della nuova mappa. Rimozione dal set di valore iniziale dei valori contenuti nel nuovo set e verifica dell'eliminazione e della diminuzione della dimensione di 10
- Sommario e design: eliminazione dal set della colletion
- Pre-condizioni: mappa adatta contenente 20 elementi e collection non null
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà true se l'eliminazione degli elementi presenti nella collection è avvenuta. Altrimenti false

removeAllSetKeyValueExceptionTest()

- Test lancio eccezione del metodo boolean removeAll(HCollection c)
- Descrizione: creazione di un set contenente i valori della mappa e di un set contenente le chiavi della mappa e verifica in entrambi i set del lancio delle eccezioni in corrispondenza di una collection null, anche nel caso di azzeramento dei due set
- Sommario e design: lancio dell'eccezione in corrispondenza di una collection null
- Pre-condizioni: mappa adatta contenente 20 elementi e collection null
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà NullPointerException

toArraySetKeyValueTest()

- Test metodo object[] toArray()
- Descrizione: creazione di 1 7 contenente i valori della mappa e di un array contenente i valori presenti nel set. Verifica uguaglianza tra la lunghezza dell' array e del set. infine verifica che il 7 contenga gli elementi presenti nell'array
- Sommario e design: copia degli elementi del set in un array
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà un array contenente i valori presenti nel set

toArrayASetKeyValueTest()

- Test metodo object[] toArray(object[]a)
- Descrizione: creazione di un set contenente i valori della mappa e di un array della stessa dimensione del set. Riempimento dell'array con valori qualsiasi e verifica dimensione dell' array pari a 20. Copia nell' array dei valori contenuti nel set, verifica della dimensione dell' array e della presenza nel set dei valori presenti nell'array
- Sommario e design: copia elementi del set in un array valido stabilito
- Pre-condizioni: mappa adatta contenente 20 elementi e array valido
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà l'array passato come argomento con all'interno i valori presenti nel set

toArraySetKeyValueExceptionTest()

- Test lancio eccezione del metodo toArray(Object[]a)
- Descrizione: creazione di un set contenente i valori della mappa di uno contenenti le chiavi della mappa. Creazione di un array con un tipo particolare e dimensione pari a quella della mappa. Verifica lancia eccezione rispettivamente dal set di chiavi e del set di valori a causa di valori non validi o non congruenti.
- Sommario e design: lancio eccezione in seguito ad array non valido
- Pre-condizioni: mappa adatta contenente 20 elementi e array null
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà NullPointerException e ArrayStoreException

removeSetValueTest()

- Test del metodo boolean remove(object o) di setValue
- Descrizione: creazione di un set contenente i valori della mappa, verifica che il set abbia dimensione pari a 20 che contenga valori 2 e 10. Rimozione del valore 2 e verifica che il set non contenga più tale elemento e che la dimensione si è calata di uno
- Sommario e design: eliminazione di un elemento del set di valori
- Pre-condizioni: mappa adatta contenente 20 elementi e valore valido
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà true a seguito della rimozione dell'elemento, se non è presente false

removeSetValueExceptionTest()

- Test lancio eccezione del metodo boolean remove(object o) di setValue
- Descrizione: creazione di un set contenente i valori della mappa e verifica lancio dell' eccezione al tentativo di rimuovere un valore null
- Sommario e design: lancio eccezione in seguito al tentativo di rimozione di un valore non valido
- Pre-condizioni: mappa adatta contenente 20 elementi e valore non valido
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà NullPointerException

removeSetKeyTest()

- Test del metodo boolean remove(object o) di setKey
- Descrizione: creazione di un set contenente le chiavi della mappa, verifica che il set abbia dimensione pari a 20 che contenga le chiavi 2 e 10. Rimozione della chiave 2 e verifica che il set non contenga più tale chiave e che la dimensione si è calata di uno
- Sommario e design: eliminazione di un elemento del set di valori
- Pre-condizioni: mappa adatta contenente 20 elementi e chiave non valida
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà true a seguito della rimozione della chiave, se non è presente false

removeSetKeyExceptionTest()

- Test lancio eccezione del metodo boolean remove(object o) di setKey
- Descrizione: creazione di un set contenente le chiavi della mappa e verifica lancio dell' eccezione al tentativo di rimuovere una chiave null
- Sommario e design: lancio eccezione in seguito al tentativo di rimozione di una chiave non valida
- Pre-condizioni: mappa adatta contenente 20 elementi e chiave non valida
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà NullPointerException

iteratorSetValueTest()

- Test metodo HIterator iterator() di setValue
- Descrizione: creazione di un set contenente i valori della mappa e di un iteratore del set. Verifica che il set abbia dimensione pari a 20 cioè uguale la mappa. Scorrimento del set tramite le l'iteratore fino alla fine con verifica dell' effettivo scorrimento
- Sommario e design: creazione di un iteratore in grado di scorrere il set
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà un iteratore in grado di scorrere il set e restituire gli elementi

iteratorSetKeyTest()

- Test metodo HIterator iterator() di setKey
- Descrizione: creazione di un set contenente le chiavi della mappa e di un iteratore del set. Verifica che il set abbia dimensione pari a 20 cioè uguale la mappa. Scorrimento del set tramite le l'iteratore fino alla fine con verifica dell' effettivo scorrimento
- Sommario e design: creazione di un iteratore in grado di scorrere il set
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà un iteratore in grado di scorrere il set e restituire gli elementi

nextSetKeyIteratorTest()

- Test del metodo object next() di setKey (HIterator)
- Descrizione: creazione di un set contenente le chiavi della mappa e di un iteratore del set. Scorrimento di tutto il set tramite l' iteratore e verifica che il set contenga quanto ritornato da next()
- Sommario e design: restituzione dell'elemento puntato dall'iteratore
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo ritornerà l'elemento puntato dall'iteratore

removeSetKeyIteratorTest()

- Test metodo void remove() di setKey (HIterator)
- Descrizione: creazione di un set contenente le chiavi della mappa e di un integratore del set. Verifica che la dimensione del set sia pari a 20 cioè a quella mappa. Scorrimento di tutta la lista Ehi rimozione graduale di tutti gli elementi puntati dall' iteratore. verifica che il set si sia azzerato
- Sommario e design: rimozione dell'elemento puntato dall'iteratore
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo eliminerà l'elemento puntato dall'iteratore

removeSetKeyExceptionIteratorTest()

- Test eccezione lanciata da void remove() di setKey (HIterator)
- Descrizione: creazione di un set contenente le chiavi della mappa e di un iteratore del set. Azzeramento del set e verifica del lancio dell' eccezione
- Sommario e design: lancio dell'eccezione a seguito del tentativo dell'eliminazione in una posizione non valida
- Pre-condizioni: mappa adatta contenente 20 elementi
- Post-condizioni: una volta finito il test la mappa dovrà contenere solamente 20 entry(chiavi e valori che vanno entrambi da 0 a 19) , condizione per il corretto test dei successivi metodi
- Risultato atteso: il metodo lancerà IllegalStateException
-