DOCUMENTO DEI TEST

Per effettuare i test mi sono servito di JUnit versione 4.12.

CollectionAdapterTest.java

PRECONDIZIONE: prima di ogni test, attraverso il metodo setUp e l'utilizzo di @Before, istanzio un nuovo oggetto vuoto di tipo CollectionAdapter, denominato "adapter".

TEST

Nome del test: testSize()

Test del metodo: size()

Descrizione del test: controllo che la dimensione iniziale di adapter sia 0. Aggiungo un elemento e controllo che la dimensione finale sia 1.

Post-Condizioni: dimensione di adapter uguale a 1.

Nome del test: testIsEmpty()

Test del metodo: isEmpty()

Descrizione del test: controllo che adapter sia vuoto. Aggiungo un elemento e controllo che la adapter non sia più vuoto.

Post-Condizioni: adapter non è più vuoto

Nome del test: testClear()

Test del metodo: clear()

Descrizione del test: aggiungo degli elementi a adapter. Invoco il metodo e controllo che la dimensione sia 0, che adapter sia vuoto.

Post-Condizioni: dimensione 0, adapter vuoto.

Nome del test: testAdd()

Test del metodo: add(E e)

Descrizione del test: invoco il metodo e aggiungo un elemento. Invoco il metodo e aggiungo lo stesso elemento. Invoco il metodo e aggiungo un elemento diverso.

Post-Condizioni: adapter ha dimensione 2, gli elementi contenuti sono diversi e corrispondono a quelli inseriti. Il metodo ritorna true la prime volta, false la seconda, true la terza.

Nome del test: testAddException()

Test del metodo: add(E e)

Descrizione del test: invoco il metodo con e = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testAddAll()

Test del metodo: addAll(Collection<? Extends E> c)

Descrizione del test: creo una collezione e aggiungo degli elementi. Inserisco uno degli elementi ora contenuti dalla collezione in adapter. Invoco il metodo. Invoco nuovamente il metodo.

Post-Condizioni: adapter contiene il corretto numero di elementi. Il metodo ritorna true la prima volta, false la seconda.

Nome del test: testAddAllVoid()

Test del metodo: addAll(Collection<? Extends E> c)

Descrizione del test: creo una collezione e la lascio vuota. Invoco il metodo.

Post-Condizioni: adapter contiene 0 elementi. Il metodo ritorna false.

Nome del test: testAddAllException()

Test del metodo: addAll(Collection<? Extends E> c)

Descrizione del test: invoco il metodo con c = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testRemove()

Test del metodo: remove(Object o)

Descrizione del test: aggiungo 3 elementi a adapter. Invoco il metodo usando come parametro uno degli elementi contenuti e controllo che ritorni false, che la dimensione sia 2 e che adapter non contenga più l'elemento. Invoco il metodo usando come parametro un elemento non contenuto da adapter e controllo che ritorni false e che la dimensione di adapter non sia cambiata.

Post-Condizioni: la dimensione di adapter è 2.

Nome del test: testRemoveException()

Test del metodo: remove(Object o)

Descrizione del test: invoco il metodo con o = null.

Post-Condizioni: viene lanciato NullPointerException

Nome del test: testRemoveAll()

Test del metodo: removeAll(Collection<?> c)

Descrizione del test: creo una Collection e aggiungo degli elementi. Aggiungo degli elementi ora contenuti nella Collection a adapter. Invoco il metodo. Aggiungo ad adapter un elemento non contenuto nella collezione. Invoco il metodo.

Post-Condizioni: dopo la prima chiamata adapter ha dimensione 0 e il metodo ritorna false. Dopo la seconda chiamata adapter ha dimensione 1 e il metodo ritorna false.

Nome del test: testRemoveAllVoid()

Test del metodo: removeAll(Collection<?> c)

Descrizione del test: crea una Collection e la lascio vuota. Aggiungo 2 elementi ad adapter. Invoco il metodo.

Post-Condizioni: adapter ha dimensione 2, il metodo ritorna false.

Nome del test: testRemoveAllException()

Test del metodo: removeAll(Collection<?> c)

Descrizione del test: invoco il metodo usando come parametro null.

Post-Condizioni: viene lanciata NullPointerException

Nome del test: testContains()

Test del metodo: contains(Object o)

Descrizione del test: aggiungo un elemento a adapter. Invoco il metodo usando come parametro l'elemento. Invoco il metodo usando come parametro un elemento diverso da quello inserito.

Post-Condizioni: il metodo ritorna prima true e dopo false.

Nome del test: testContainsException()

Test del metodo: contains(Object o)

Descrizione del test: invoco il metodo usando come parametro null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testContainsAll()

Test del metodo: containsAll(Collection<?> c)

Descrizione del test: creo una collezione e aggiungo degli elementi. Aggiungo gli stessi elementi ed un elemento diverso ad adapter. Invoco il metodo. Aggiungo alla collezione un elemento non contenuto da adapter. Invoco il metodo.

Post-Condizioni: il metodo ritorna prima true e poi false.

Nome del test: testContainsAllVoid()

Test del metodo: containsAll(Collection<?> c)

Descrizione del test: creo una collezione e la lascio vuota. Invoco il metodo. Aggiungo degli elementi ad adapter. Invoco il metodo nuovamente.

Post-Condizioni: il metodo ritorna prima true e poi false.

Nome del test: testContainsAllException()

Test del metodo: containsAll(Collection<?> c)

Descrizione del test: invoco il metodo usando come parametro null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testRetainAll()

Test del metodo: retainAll(Collection<?> c)

Descrizione del test: creo c e aggiungo degli elementi. Inserisco in adapter elementi contenuti e non contenuti da c. Invoco il metodo. Invoco nuovamente il metodo e controllo.

Post-Condizioni: adapter contiene solo gli elementi contenuti da c. Il metodo ritorna true la prima volta, false la seconda

Nome del test: testRetainAllVoid()

Test del metodo: retainAll(Collection<?> c)

Descrizione del test: creo c e la lascio vuota. Aggiungo degli elementi a adapter. Invoco il metodo.

Post-Condizioni: adapter ha dimensione 0, il metodo ritorna true.

Nome del test: testRetainAllException()

Test del metodo: retainAll(Collection<?> c)

Descrizione del test: invoco il metodo usando come parametro null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testEquals()

Test del metodo: equals(Object o)

Descrizione del test: creo una Collection e aggiungo degli elementi. Aggiungo gli stessi elementi a adapter. Invoco il metodo. Aggiungo un elemento alla collezione. Invoco il metodo nuovamente.

Post-Condizioni: il metodo ritorna true la prima volta, false la seconda.

Nome del test: testEqualsVoid()

Test del metodo: equals(Object o)

Descrizione del test: creo una Collection e la lascio vuota. Invoco il metodo.

Post-Condizioni: il metodo ritorna true.

Nome del test: testEqualsInstanceOf()

Test del metodo: equals(Object o)

Descrizione del test: creo un Set. Aggiungo gli stessi elementi al Set e a adapter. Invoco il metodo.

Post-Condizioni: il metodo ritorna false.

Nome del test: testHashCode()

Test del metodo: hashCode()

Descrizione del test: creo una Collection. Aggiungo alla Collection e a adapter gli stessi elementi. Invoco il metodo. Aggiungo un elemento a adapter. Invoco il metodo nuovamente.

Post-Condizioni: il metodo ritorna valori uguali la prima volta, diversi la seconda.

Nome del test: testIteratorHasNext()

Test del metodo: hasNext()

Descrizione del test: aggiungo degli elementi a adapter. Invoco il metodo.

Post-Condizioni: l’iteratore scorre completamente tutti gli elementi di adapter dal primo all’ultimo.

Nome del test: testIteratorNext()

Test del metodo: next()

Descrizione del test: aggiungo degli elementi a adapter. Creo l’iteratore e invoco consecutivamente il metodo.

Post-Condizioni: i valori ritornati da next() sono corretti.

Nome del test: testIteratorNextException()

Test del metodo: next()

Descrizione del test: aggiungo degli elementi a adapter. Creo l’iteratore e invoco il metodo un numero di volte = adapter.size() + 1.

Post-Condizioni: viene lanciato NoSuchElementException.

Nome del test: testIteratorRemove()

Test del metodo: remove()

Descrizione del test: aggiungo degli elementi a adapter. Creo l’iteratore. Attraverso next() scorro l’iteratore. Invoco il metodo.

Post-Condizioni: viene rimosso l’ultimo elemento ritornato da next(). La dimensione di adapter diminuisce di 1.

Nome del test: testIteratorRemoveException1()

Test del metodo: remove()

Descrizione del test: aggiungo degli elementi a adapter. Creo l’iteratore. Attraverso next() scorro l’iteratore. Invoco il metodo due volte consecutivamente.

Post-Condizioni: viene lanciata IllegalStateException.

Nome del test: testIteratorRemoveException2()

Test del metodo: remove()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco il metodo.

Post-Condizioni: viene lanciata IllegalStateException.

Nome del test: testToArray()

Test del metodo: toArray()

Descrizione del test: aggiungo degli elementi ad adapter. Invoco il metodo e controllo attraverso un iteratore che quanto ritornato da toArray() sia corretto.

Post-Condizioni: //

SetAdapterTest.java

PRECONDIZIONE: prima di ogni test, attraverso il metodo setUp e l'utilizzo di @Before, istanzio un nuovo oggetto vuoto di tipo SetAdapter, denominato "adapter".

TEST

Nome del test: testSize()

Test del metodo: size()

Descrizione del test: controllo che la dimensione iniziale di adapter sia 0. Aggiungo un elemento e controllo che la dimensione finale sia 1.

Post-Condizioni: dimensione di adapter uguale a 1.

Nome del test: testIsEmpty()

Test del metodo: isEmpty()

Descrizione del test: controllo che adapter sia vuoto. Aggiungo un elemento e controllo che la adapter non sia più vuoto.

Post-Condizioni: adapter non è più vuoto

Nome del test: testClear()

Test del metodo: clear()

Descrizione del test: aggiungo degli elementi a adapter. Invoco il metodo e controllo che la dimensione sia 0, che adapter sia vuoto.

Post-Condizioni: dimensione 0, adapter vuoto.

Nome del test: testAdd()

Test del metodo: add(E e)

Descrizione del test: invoco il metodo e aggiungo un elemento. Invoco il metodo e aggiungo lo stesso elemento. Invoco il metodo e aggiungo un elemento diverso.

Post-Condizioni: adapter ha dimensione 2, gli elementi contenuti sono diversi e corrispondono a quelli inseriti. Il metodo ritorna true la prime volta, false la seconda, true la terza.

Nome del test: testAddException()

Test del metodo: add(E e)

Descrizione del test: invoco il metodo con e = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testAddAll()

Test del metodo: addAll(Collection<? Extends E> c)

Descrizione del test: creo una collezione e aggiungo degli elementi. Inserisco uno degli elementi ora contenuti dalla collezione in adapter. Invoco il metodo. Invoco nuovamente il metodo.

Post-Condizioni: adapter contiene il corretto numero di elementi. Il metodo ritorna true la prima volta, false la seconda.

Nome del test: testAddAllVoid()

Test del metodo: addAll(Collection<? Extends E> c)

Descrizione del test: creo una collezione e la lascio vuota. Invoco il metodo.

Post-Condizioni: adapter contiene 0 elementi. Il metodo ritorna false.

Nome del test: testAddAllException()

Test del metodo: addAll(Collection<? Extends E> c)

Descrizione del test: invoco il metodo con c = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testRemove()

Test del metodo: remove(Object o)

Descrizione del test: aggiungo 3 elementi a adapter. Invoco il metodo usando come parametro uno degli elementi contenuti e controllo che ritorni false, che la dimensione sia 2 e che adapter non contenga più l'elemento. Invoco il metodo usando come parametro un elemento non contenuto da adapter e controllo che ritorni false e che la dimensione di adapter non sia cambiata.

Post-Condizioni: la dimensione di adapter è 2.

Nome del test: testRemoveException()

Test del metodo: remove(Object o)

Descrizione del test: invoco il metodo con o = null.

Post-Condizioni: viene lanciato NullPointerException

Nome del test: testRemoveAll()

Test del metodo: removeAll(Collection<?> c)

Descrizione del test: creo una Collection e aggiungo degli elementi. Aggiungo degli elementi ora contenuti nella Collection a adapter. Invoco il metodo. Aggiungo ad adapter un elemento non contenuto nella collezione. Invoco il metodo.

Post-Condizioni: dopo la prima chiamata adapter ha dimensione 0 e il metodo ritorna false. Dopo la seconda chiamata adapter ha dimensione 1 e il metodo ritorna false.

Nome del test: testRemoveAllVoid()

Test del metodo: removeAll(Collection<?> c)

Descrizione del test: crea una Collection e la lascio vuota. Aggiungo 2 elementi ad adapter. Invoco il metodo.

Post-Condizioni: adapter ha dimensione 2, il metodo ritorna false.

Nome del test: testRemoveAllException()

Test del metodo: removeAll(Collection<?> c)

Descrizione del test: invoco il metodo usando come parametro null.

Post-Condizioni: viene lanciata NullPointerException

Nome del test: testContains()

Test del metodo: contains(Object o)

Descrizione del test: aggiungo un elemento a adapter. Invoco il metodo usando come parametro l'elemento. Invoco il metodo usando come parametro un elemento diverso da quello inserito.

Post-Condizioni: il metodo ritorna prima true e dopo false.

Nome del test: testContainsException()

Test del metodo: contains(Object o)

Descrizione del test: invoco il metodo usando come parametro null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testContainsAll()

Test del metodo: containsAll(Collection<?> c)

Descrizione del test: creo una collezione e aggiungo degli elementi. Aggiungo gli stessi elementi ed un elemento diverso ad adapter. Invoco il metodo. Aggiungo alla collezione un elemento non contenuto da adapter. Invoco il metodo.

Post-Condizioni: il metodo ritorna prima true e poi false.

Nome del test: testContainsAllVoid()

Test del metodo: containsAll(Collection<?> c)

Descrizione del test: creo una collezione e la lascio vuota. Invoco il metodo. Aggiungo degli elementi ad adapter. Invoco il metodo nuovamente.

Post-Condizioni: il metodo ritorna prima true e poi false.

Nome del test: testContainsAllException()

Test del metodo: containsAll(Collection<?> c)

Descrizione del test: invoco il metodo usando come parametro null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testRetainAll()

Test del metodo: retainAll(Collection<?> c)

Descrizione del test: creo c e aggiungo degli elementi. Inserisco in adapter elementi contenuti e non contenuti da c. Invoco il metodo. Invoco nuovamente il metodo e controllo.

Post-Condizioni: adapter contiene solo gli elementi contenuti da c. Il metodo ritorna true la prima volta, false la seconda

Nome del test: testRetainAllVoid()

Test del metodo: retainAll(Collection<?> c)

Descrizione del test: creo c e la lascio vuota. Aggiungo degli elementi a adapter. Invoco il metodo.

Post-Condizioni: adapter ha dimensione 0, il metodo ritorna true.

Nome del test: testRetainAllException()

Test del metodo: retainAll(Collection<?> c)

Descrizione del test: invoco il metodo usando come parametro null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testEquals()

Test del metodo: equals(Object o)

Descrizione del test: creo un Set e aggiungo degli elementi. Aggiungo gli stessi elementi a adapter. Invoco il metodo. Aggiungo un elemento alla collezione. Invoco il metodo nuovamente.

Post-Condizioni: il metodo ritorna true la prima volta, false la seconda.

Nome del test: testEqualsVoid()

Test del metodo: equals(Object o)

Descrizione del test: creo un Set e lo lascio vuoto. Invoco il metodo.

Post-Condizioni: il metodo ritorna true.

Nome del test: testEqualsInstanceOf()

Test del metodo: equals(Object o)

Descrizione del test: creo una collezione. Aggiungo gli stessi elementi alla collezione e a adapter. Invoco il metodo.

Post-Condizioni: il metodo ritorna false.

Nome del test: testHashCode()

Test del metodo: hashCode()

Descrizione del test: creo un Set. Aggiungo al Set e a adapter gli stessi elementi. Invoco il metodo. Aggiungo un elemento a adapter. Invoco il metodo nuovamente.

Post-Condizioni: il metodo ritorna valori uguali la prima volta, diversi la seconda.

Nome del test: testIteratorHasNext()

Test del metodo: hasNext()

Descrizione del test: aggiungo degli elementi a adapter. Invoco il metodo.

Post-Condizioni: l'iteratore scorre completamente tutti gli elementi di adapter dal primo all'ultimo.

Nome del test: testIteratorNext()

Test del metodo: next()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco consecutivamente il metodo.

Post-Condizioni: i valori ritornati da next() sono corretti.

Nome del test: testIteratorNextException()

Test del metodo: next()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco il metodo un numero di volte = adapter.size() + 1.

Post-Condizioni: viene lanciato NoSuchElementException.

Nome del test: testIteratorRemove()

Test del metodo: remove()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore. Attraverso next() scorro l'iteratore. Invoco il metodo.

Post-Condizioni: viene rimosso l'ultimo elemento ritornato da next(). La dimensione di adapter diminuisce di 1.

Nome del test: testIteratorRemoveException1()

Test del metodo: remove()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore. Attraverso next() scorro l'iteratore. Invoco il metodo due volte consecutivamente.

Post-Condizioni: viene lanciata IllegalStateException.

Nome del test: testIteratorRemoveException2()

Test del metodo: remove()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco il metodo.

Post-Condizioni: viene lanciata IllegalStateException.

Nome del test: testToArray()

Test del metodo: toArray()

Descrizione del test: aggiungo degli elementi ad adapter. Invoco il metodo e controllo attraverso un iteratore che quanto ritornato da toArray() sia corretto.

Post-Condizioni: //

ListAdapterTest.java

***A causa della mancanza di tempo, non sono presenti le tabelle riguardanti la classe interna Sublist e i relativi iteratori. Mi scuso per l'inconveniente.**

Ad ogni modo, i test di SubList rispecchiano in larga parte quelli effettuati per List (viene testata ogni eccezione, si testano tutti i "casi particolari" testati su List (ad esempio: addAll(Collection<? extends E> c) con c vuoto)) tenendo inoltre conto di verificare che ogni modifica effettuata alla sottolista risulti al tempo stesso visibile nella lista "madre".

PRECONDIZIONE: prima di ogni test, attraverso il metodo setUp e l'utilizzo di @Before, istanzio un nuovo oggetto vuoto di tipo ListAdapter, denominato "adapter".

TEST

Nome del test: testSize()

Test del metodo: size()

Descrizione del test: controllo che la dimensione iniziale di adapter sia 0. Aggiungo un elemento e controllo che la dimensione finale sia 1.

Post-Condizioni: dimensione di adapter uguale a 1.

Nome del test: testIsEmpty()

Test del metodo: isEmpty()

Descrizione del test: controllo che adapter sia vuoto. Aggiungo un elemento e controllo che la adapter non sia più vuoto.

Post-Condizioni: adapter non è più vuoto

Nome del test: testClear()

Test del metodo: clear()

Descrizione del test: aggiungo degli elementi a adapter. Invoco il metodo e controllo che la dimensione sia 0, che adapter sia vuoto.

Post-Condizioni: dimensione 0, adapter vuoto.

Nome del test: testGet()

Test del metodo: get()

Descrizione del test: inserisco in adapter degli elementi. Invoco il metodo e controllo che il valore ritornato sia corretto.

Post-Condizioni: //

Nome del test: testGetException1()

Test del metodo: get(int index)

Descrizione del test: invoco il metedo get con un qualsiasi indice >= 0. Controllo che lanci IndexOutOfBoundException.

Post-Condizioni: //

Nome del test: testGetException2()

Test del metodo: get(int index)

Descrizione del test: invoco il metedo get con un qualsiasi indice < 0. Controllo che lanci IndexOutOfBoundException.

Post-Condizioni: //

Nome del test: testGetException()

Test del metodo: get(int index)

Descrizione del test: aggiungo un elemento a adapter. Invoco il metodo con indice = 1 (= dimensione di adapter) e controllo che lanci IndexOutOfBoundsException

Post-Condizioni: //

Nome del test: testSet()

Test del metodo: set(int index,E e)

Descrizione del test: aggiungo degli elementi a adapter. Invoco il metodo e controllo che l'elemento ritornato sia quello precedentemente contenuto al dato indice. Controllo che l'elemento inserito sia nella posizione corretta in adapter. Controllo che la dimensione di adapter sia congrua.

Post-Condizioni: adapter contiene un elemento diverso rispetto a quelli inseriti all'inizio del test.

Nome del test: testSetException1()

Test del metodo: set(int index,E e)

Descrizione del test: invoco il metodo con un qualsiasi indice >= 0 e controllo che lanci IndexOutOfBoundsException.

Post-Condizioni: //

Nome del test: testSetException2()

Test del metodo: set(int index,E e)

Descrizione del test: invoco il metodo con un qualsiasi indice < 0 e controllo che lanci IndexOutOfBoundsException.

Post-Condizioni: //

Nome del test: testSetException3()

Test del metodo: set(int index,E e)

Descrizione del test: aggiungo ad adapter un elemento. Invoco il metodo con indice uguale a 1 (= dimensione di adapter) e controllo che lanci IndexOutOfBoundsException.

Post-Condizioni: //

Nome del test: testSetException4()

Test del metodo: set(int index,E e)

Descrizione del test: provo ad inserire attraverso il metodo null. Controllo che lanci NullPointerException.

Post-Condizioni: //

Nome del test: testIndexOf ()

Test del metodo: indexOf(Object o)

Descrizione del test: aggiungo elementi a adapter. Testo il metodo con elemento contenuto e elemento non contenuto da adapter.

Post-Condizioni: adapter ritorna l'indice per l'elemento contenuto, -1 per l'altro.

Nome del test: testIndexOfException()

Test del metodo: indexOf(Object o)

Descrizione del test: testo il metodo con null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testLastIndexOf()

Test del metodo: lastIndexOf(Object o)

Descrizione del test: aggiungo elementi a adapter. Testo il metodo con elemento contenuto e elemento non contenuto da adapter.

Post-Condizioni: adapter ritorna l'indice per l'elemento contenuto, -1 per l'altro.

Nome del test: testLastIndexOfException()

Test del metodo: indexOf(Object o)

Descrizione del test: testo il metodo con null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testHashCode()

Test del metodo: hashCode()

Descrizione del test: creo una lista. Aggiungo gli stessi elementi alla lista e a adapter. Testo il metodo. Aggiungo un altro elemento alla lista. Testo il metodo.

Post-Condizioni: valori uguali per il primo test, diversi per il secondo.

Nome del test: testAddWithIndex()

Test del metodo: add(int index,E element)

Descrizione del test: aggiungo degli elementi a adapter. Testo il metodo.

Post-Condizioni: la dimensione di adapter aumenta di 1, gli elementi assumono gli indici corretti.

Nome del test: testAddWithIndexException1()

Test del metodo: add(int index,E element)

Descrizione del test: testo il metodo con element = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testAddWithIndexException2()

Test del metodo: add(int index,E element)

Descrizione del test: testo il metodo con index < 0.

Post-Condizioni: viene lanciata IndexOutOfBoundsException

Nome del test: testAddWithIndexException3()

Test del metodo: add(int index,E element)

Descrizione del test: testo il metodo con index = adapter.size() + 1.

Post-Condizioni: viene lanciata IndexOutOfBoundsException

Nome del test: testAdd()

Test del metodo: add(E e)

Descrizione del test: aggiungo un elemento a adapter. Testo il metodo. Test il metodo nuovamente con lo stesso valore.

Post-Condizioni: il metodo ritorna prima true e poi false, la dimensione aumenta di 1 ogni volta, gli elementi sono nella posizione corretta.

Nome del test: testAddException()

Test del metodo: add(E e)

Descrizione del test: testo il metodo con e = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: addAllWithIndex()

Test del metodo: addAll(int index,Collection<? Extends E> c)

Descrizione del test: creo c e aggiungo elementi. Aggiungo elementi a adapter. Testo il metodo con un indice valido. Testo nuovamente il metodo con gli stessi parametri.

Post-Condizioni: dimensione di adapter aumentata correttamente, tutti gli elementi inseriti si trovano agli indici corretti. Il metodo ritorna true la prima volta, false la seconda.

Nome del test: addAllWithIndexException1()

Test del metodo: addAll(int index,Collection<? Extends E> c)

Descrizione del test: testo il metodo con indice < 0

Post-Condizioni: viene lanciata IndexOutOfBoundsException

Nome del test: addAllWithIndexException2()

Test del metodo: addAll(int index,Collection<? Extends E> c)

Descrizione del test: testo il metodo con indice = adapter.size() + 1.

Post-Condizioni: viene lanciata IndexOutOfBoundsException

Nome del test: addAllWithIndexException3()

Test del metodo: addAll(int index,Collection<? Extends E> c)

Descrizione del test: testo il metodo c = null.

Post-Condizioni: viene lanciata NullPointerException

Nome del test: testAddAll()

Test del metodo: addAll(Collection<? extends E> c)

Descrizione del test: creo c e aggiungo degli elementi. Aggiungo degli elementi a adapter. Invoco il metodo due volte consecutivamente.

Post-Condizioni: la dimensione di size aumenta correttamente, gli elementi inseriti si trovano nelle posizioni corrette. Il metodo ritorna true la prima volta, false la seconda.

Nome del test: testAddAllVoid()

Test del metodo: addAll(Collection<? extends E> c)

Descrizione del test: creo c e la lascio vuota. Aggiungo degli elementi a adapter. Testo il metodo.

Post-Condizioni: il metodo ritorna false.

Nome del test: testAddAllException()

Test del metodo: addAll(Collection<? extends E> c)

Descrizione del test: testo il metodo con c = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testRemoveWithIndex()

Test del metodo: remove(int index)

Descrizione del test: aggiungo degli elementi a adapter. Testo il metodo.

Post-Condizioni: dimensione di size diminuita di 1, il metodo rimuove l'elemento contenuto al dato indice, gli altri elementi assumono gli indici corretti.

Nome del test: testRemoveWithIndex1()

Test del metodo: remove(int index)

Descrizione del test: aggiungo un elemento a adapter, testo il metodo con indice < 0.

Post-Condizioni: viene lanciata IndexOutOfBoundsException.

Nome del test: testRemoveWithIndex2()

Test del metodo: remove(int index)

Descrizione del test: aggiungo un elemento a adapter, testo il metodo con indice = adapter.size().

Post-Condizioni: viene lanciata IndexOutOfBoundsException.

Nome del test: testRemove()

Test del metodo: remove(Object o)

Descrizione del test: aggiungo degli elementi a adapter. Test il metodo con un elemento contenuto. Testo il metodo con un elemento non contenuto.

Post-Condizioni: la dimensione di adapter diminuisce di 1, viene eliminata solo la prima occorrenza dell'elemento contenuto, i restanti elementi assumono gli indici corretti. Il metodo ritorna true la prima volta, false la seconda.

Nome del test: testRemoveException()

Test del metodo: remove(Objedct o)

Descrizione del test: testo il metodo con o = null.

Post-Condizioni: viene lanciata NullPointerException

Nome del test: testRemoveAll()

Test del metodo: removeAll(Collection<?> c)

Descrizione del test: creo c e aggiungo elementi. Aggiungo elementi diversi da quelli di c a adapter. Invoco il metodo. Aggiungo a c elementi contenuti in adapter. Invoco il metodo.

Post-Condizioni: dimensione di adapter diminuita correttamente, eliminate da adapter tutte le occorenze degli elementi contenuti in c. Il metodo ritorna false la prima volta, true la seconda.

Nome del test: testRemoveAllVoid()

Test del metodo: removeAll(Collection<?> c)

Descrizione del test: creo c e la lascio vuota. Aggiungo elementi a adapter. Testo il metodo.

Post-Condizioni: il metodo ritorna false.

Nome del test: testRemoveAllException()

Test del metodo: removeAll(Collection<?> c)

Descrizione del test: testo il metodo con c = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testContains()

Test del metodo: contains(Object o)

Descrizione del test: aggiungo un elemento a adapter. Testo il metodo con elemento contenuto. Testo il metodo con elemento non conetunoto.

Post-Condizioni: il metodo ritorna true la prima volta, false la seconda.

Nome del test: testContainsException()

Test del metodo: contains(Object o)

Descrizione del test: testo il metodo con o = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testContainsAll()

Test del metodo: containsAll(Collection<?> c)

Descrizione del test: creo c e aggiungo degli elementi. Aggiungo gli stessi elementi a adapter. Invoco il metodo. Aggiungo un nuovo elemento a c. Invoco il metodo.

Post-Condizioni: il metodo ritorna true la prima volta, false la seconda.

Nome del test: testContainsAllVoid()

Test del metodo: containsAll(Collection<?> c)

Descrizione del test: creo c e la lascio vuota. Invoco il metodo.

Post-Condizioni: il metodo ritorna true.

Nome del test: testContainsAllException()

Test del metodo: containsAll(Collection<?> c)

Descrizione del test: testo il metodo con c = null.

Post-Condizioni: viene lanciata NullPointerException

Nome del test: testRetainAll()

Test del metodo: retainAll(Collection<?> c)

Descrizione del test: creo c e aggiungo degli elementi. Aggiungo a adapter gli stessi elementi di c e altri elementi diversi. Invoco il metodo due volte consecutivamente.

Post-Condizioni: la dimensione di adapter è corretta, tutte le occorrenze degli elementi non contenuti in c vengono eliminate. Il metodo la prima volta ritorna true, la seconda false.

Nome del test: testRetainAllVoid()

Test del metodo: retainAll(Collection<?> c)

Descrizione del test: creo c e la lascio vuota. Invoco il metodo.

Post-Condizioni: la dimensione di adapter è 0, tutti gli elementi vengono eliminati.

Nome del test: testRetainAllException()

Test del metodo: retainAll(Collection<?> c)

Descrizione del test: testo il metodo con c = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testEquals()

Test del metodo: equals(Object o)

Descrizione del test: creo una lista e aggiungo gli stessi elementi che aggiungo a adapter. Invoco il metodo. Aggiungo alla lista un nuovo elemento. Invoco il metodo.

Post-Condizioni: il metodo la prima volta ritorna true, la seconda false.

Nome del test: testEqualsVoid()

Test del metodo: equals(Object o)

Descrizione del test: creo una lista e la lascio vuota. Invoco il metodo.

Post-Condizioni: il metodo ritorna true.

Nome del test: testInstanceOfEquals()

Test del metodo: equals(Object o)

Descrizione del test: creo un Set e aggiungo gli stessi elementi che aggiungo a adapter. Invoco il metodo.

Post-Condizioni: il metodo ritorna false.

Nome del test: testToArray()

Test del metodo: toArray()

Descrizione del test: aggiungo degli elementi a adapter. Creo un interatore, invoco il metodo.

Post-Condizioni: gli elementi dell'array e gli elementi ritornati dall'iteratore si trovano nello stesso ordine.

Nome del test: testIteratorHasNext()

Test del metodo: hasNext()

Descrizione del test: aggiungo degli elementi a adapter. Invoco il metodo.

Post-Condizioni: l'iteratore scorre completamente tutti gli elementi di adapter dal primo all'ultimo.

Nome del test: testIteratorNext()

Test del metodo: next()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco consecutivamente il metodo.

Post-Condizioni: i valori ritornati da next() sono corretti.

Nome del test: testIteratorNextException()

Test del metodo: next()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco il metodo un numero di volte = adapter.size() + 1.

Post-Condizioni: viene lanciato NoSuchElementException.

Nome del test: testIteratorRemove()

Test del metodo: remove()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore. Attraverso next() scorro l'iteratore. Invoco il metodo.

Post-Condizioni: viene rimosso l'ultimo elemento ritornato da next(). La dimensione di adapter diminuisce di 1.

Nome del test: testIteratorRemoveException1()

Test del metodo: remove()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore. Attraverso next() scorro l'iteratore. Invoco il metodo due volte consecutivamente.

Post-Condizioni: viene lanciata IllegalStateException.

Nome del test: testIteratorRemoveException2()

Test del metodo: remove()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco il metodo.

Post-Condizioni: viene lanciata IllegalStateException.

Nome del test: testListIteratorHasNext()

Test del metodo: hasNext()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco il metodo.

Post-Condizioni: l'iteratore scorre tutti gli elementi di adapter dal primo all'ultimo.

Nome del test: testListIteratorHasPrevious()

Test del metodo: hasPrevious()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e scorro tutti gli elementi fino all'ultimo. Invoco il metodo.

Post-Condizioni: l'iteratore scorre tutti gli elementi dall'ultimo al primo.

Nome del test: testListIteratorNext()

Test del metodo: next()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco consecutivamente il metodo.

Post-Condizioni: i valori ritornati da next() sono corretti.

Nome del test: testListIteratorNextException()

Test del metodo: next()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco il metodo un numero di volte = adapter.size() + 1.

Post-Condizioni: viene lanciato NoSuchElementException.

Nome del test: testListIteratorPrevious()

Test del metodo: previous()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e scorro tutti gli elementi fino all'ultimo. Invoco consecutivamente il metodo.

Post-Condizioni: gli elementi ritornati dal metodo sono corretti.

Nome del test: testListIteratorPreviousException()

Test del metodo: previous()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e scorro tutti gli elementi fino all'ultimo. Invoco il metodo un numero di volte = adapter.size() + 1.

Post-Condizioni: viene lanciato NoSuchElementException.

Nome del test: testListIteratorNextIndex()

Test del metodo: nextIndex()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco consecutivamente il metodo scorrendo gli elementi dal primo all'ultimo.

Post-Condizioni: i valori ritornati sono corretti.

Nome del test: testListIteratorPreviousIndex()

Test del metodo: previousIndex()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e scorro tutti gli elementi fino all'ultimo. Invoco consecutivamente il metodo scorrendo gli elementi dall'ultimo al primo.

Post-Condizioni: i valori ritornati sono corretti.

Nome del test: testListIteratorAdd()

Test del metodo: add(E e)

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore, scorro gli elementi e invoco il metodo.

Post-Condizioni: la dimensione di adapter aumenta di 1, gli elementi all'interno di adapter assumono la posizione corretta.

Nome del test: testListIteratorAddException()

Test del metodo: add(E e)

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco il metodo con e = null.

Post-Condizioni: viene lanciata IllegalArgumentException

Nome del test: testListIteratorRemove()

Test del metodo: remove()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore. Scorro gli elementi di adapter con next() e previous(). Invoco il metodo più volte.

Post-Condizioni: la dimensione di adapter diminuisce del numero di volte che è stato invocato remove(). Il metodo rimuove l'ultimo elemento ritornato da next() o previous().

Nome del test: testListIteratorRemoveException1()

Test del metodo: remove()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco il metodo.

Post-Condizioni: viene lanciata IllegalStateException.

Nome del test: testListIteratorRemoveException2()

Test del metodo: remove()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore, scorro gli elementi con next() e invoco il metodo.

Post-Condizioni: viene lanciata IllegalStateException.

Nome del test: testListIteratorRemoveException3()

Test del metodo: remove()

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore, scorro gli elementi con next(). Scorro con previous() e invoco il metodo.

Post-Condizioni: viene lanciata IllegalStateException.

Nome del test: testListIteratorSet()

Test del metodo: set(E e)

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e testo due volte il metodo chiamando prima di esso next() e previous().

Post-Condizioni: la dimensione di adapter non varia, adapter contiene gli elementi correnti posizionati correttamente

Nome del test: testListIteratorSetException1()

Test del metodo: set(E e)

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore e invoco il metodo.

Post-Condizioni: viene lanciata IllegalStateException.

Nome del test: testListIteratorSetException2()

Test del metodo: set(E e)

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore, invoco next(), invoco remove() e invoco il metodo.

Post-Condizioni: viene lanciata IllegalStateException.

Nome del test: testListIteratorSetException3()

Test del metodo: set(E e)

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore, invoco next(), invoco add() e invoco il metodo.

Post-Condizioni: viene lanciata IllegalStateException.

Nome del test: testListIteratorSetException4()

Test del metodo: set(E e)

Descrizione del test: aggiungo degli elementi a adapter. Creo l'iteratore, invoco next() e di seguito il metodo con null come parametro.

Post-Condizioni: viene lanciata IllegalArgumentException.

Nome del test: testListIteratorWithIndexException1()

Test del metodo: listIterator(int index)

Descrizione del test: aggiungo un elemento a adapter. Invoco il metodo con index = adapter.size().

Post-Condizioni: viene lanciata IndexOutOfBoundsException.

Nome del test: testListIteratorWithIndexException2()

Test del metodo: listIterator(int index)

Descrizione del test: aggiungo un elemento a adapter. Invoco il metodo con index < 0.

Post-Condizioni: viene lanciata IndexOutOfBoundsException.

MapAdapterTest.java

***La classe contiene anche i test relativi alle classi EntrySet, KeySet e MapValues. Dato che le prime due implementano l'interfaccia Set e l'ultima l'interfaccia Collection, ho ritenuto ridondante scrivere le tabelle dei test di queste classi.**

Ad ogni modo, i test relativi a queste classi sono identici a quelli realizzati in SetAdapterTest.java e CollectionAdapterTest.java.

PRECONDIZIONE: prima di ogni test, attraverso il metodo setUp e l'utilizzo di @Before, istanzio un nuovo oggetto vuoto di tipo MapAdapter, denominato "adapter".

TEST

Nome del test: testSize()

Test del metodo: size()

Descrizione del test: controllo che la dimensione iniziale di adapter sia 0. Aggiungo un elemento e controllo che la dimensione finale sia 1.

Post-Condizioni: dimensione di adapter uguale a 1.

Nome del test: testIsEmpty()

Test del metodo: isEmpty()

Descrizione del test: controllo che adapter sia vuoto. Aggiungo un elemento e controllo che la adapter non sia più vuoto.

Post-Condizioni: adapter non è più vuoto

Nome del test: testClear()

Test del metodo: clear()

Descrizione del test: aggiungo degli elementi a adapter. Invoco il metodo e controllo che la dimensione sia 0, che adapter sia vuoto.

Post-Condizioni: dimensione 0, adapter vuoto.

Nome del test: testGet()

Test del metodo: get(Object key)

Descrizione del test: invoco il metodo con adapter vuoto. Inserisco un elemento in adapter. Invoco il metodo nuovamente.

Post-Condizioni: il metodo ritorna la prima volta null, la seconda volta il valore associato alla chiave inserita precedentemente.

Nome del test: testGetException()

Test del metodo: get(Object key)

Descrizione del test: invoco il metodo key = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testPut()

Test del metodo: put(K key, V value)

Descrizione del test: invoco il metodo e inserisco un elemento (chiave-valore). Invoco nuovamente il metodo e inserisco con la stessa chiave un altro valore.

Post-Condizioni: dimensione di adapter = 1. La prima volta il metodo ritorna null, la seconda volta il valore precedentemente mappato da quella chiave.

Nome del test: testPutException1()

Test del metodo: put(K key, V value)

Descrizione del test: invoco il metodo con key = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testPutException2()

Test del metodo: put(K key, V value)

Descrizione del test: invoco il metodo con value = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testContainsKey()

Test del metodo: containsKey(Object o)

Descrizione del test: invoco il metodo con adapter vuoto. Inserisco un elemento (chiave-valore) e invoco il metodo usando come parametro la chiave dell'elemento.

Post-Condizioni: il metodo ritorna false la prima volta, true la seconda.

Nome del test: testContainsKeyException()

Test del metodo: containsKey(Object o)

Descrizione del test: invoco il metodo con o = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testContainsValue()

Test del metodo: containsValue(Object o)

Descrizione del test: invoco il metodo con adapter vuoto. Inserisco un elemento (chiave-valore) e invoco il metodo usando come parametro il valore dell'elemento.

Post-Condizioni: il metodo ritorna false la prima volta, true la seconda.

Nome del test: testContainsValueException()

Test del metodo: containsValue(Object o)

Descrizione del test: invoco il metodo con o = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testRemove()

Test del metodo: remove(Object key)

Descrizione del test: invoco il metodo con adapter vuoto. Inserisco un elemento (chiave-valore) e invoco il metodo usando come parametro la chiave dell'elemento,

Post-Condizioni: dimensione di adapter = 0. Il metodo ritorna null la prima volta, il valore dell'elemento rimosso la seconda.

Nome del test: testRemoveException()

Test del metodo: remove(Object key)

Descrizione del test: invoco il metodo con key = null.

Post-Condizioni: viene lanciata NullPointerException.

Nome del test: testEquals()

Test del metodo: equals(Object o)

Descrizione del test: creo una mappa. Inserisco nella mappa gli stessi elementi che inserisco in adapter. Invoco il metodo. Aggiungo un elemento (chiave-valore) alla mappa. Invoco il metodo.

Post-Condizioni: il metodo ritorna true la prima volta, false la seconda.

Nome del test: testEqualsVoid()

Test del metodo: equals(Object o)

Descrizione del test: creo una mappa e la lascio vuota. Invoco il metodo.

Post-Condizioni: il metodo ritorna true.

Nome del test: testEqualsInstanceOf()

Test del metodo: equlas(Object o)

Descrizione del test: creo una collezione. Invoco il metodo.

Post-Condizioni: il metodo ritorna false.

Nome del test: testHashCode()

Test del metodo: hashCode()

Descrizione del test: creo una mappa. Inserisco nella mappa gli stessi elementi che inserisco in adapter. Invoco il metodo. Inserisco un nuovo elemento in adapter. Invoco il metodo.

Post-Condizioni: il metodo ritorna valori uguali la prima volta, diversi la seconda.

Nome del test: testPutAll()

Test del metodo: putAll(Map<? Extends K,? Extends V> m)

Descrizione del test: creo m e aggiungo dei valori. Invoco il metodo.

Post-Condizioni: dimensione di adapter = 3, adapter contiene tutti gli elementi (chiave-valore) presenti in m.

Nome del test: testPutAllException()

Test del metodo: putAll(Map<? Extends K,? Extends V> m)

Descrizione del test: invoco il metodo con m = null.

Post-Condizioni: viene lanciata NullPointerException.