JavaScript is disabled on your browser.

[Skip navigation links](#1fob9te)

* [Package](http://docs.google.com/package-summary.html)
* Class
* [Use](http://docs.google.com/class-use/DenseBag.html)
* [Tree](http://docs.google.com/package-tree.html)
* [Deprecated](http://docs.google.com/deprecated-list.html)
* [Index](http://docs.google.com/index-files/index-1.html)
* [Help](http://docs.google.com/help-doc.html)
* Prev Class
* Next Class
* [Frames](http://docs.google.com/index.html?DenseBag.html)
* [No Frames](http://docs.google.com/DenseBag.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* Field |
* [Constr](#3znysh7) |
* [Method](#2et92p0)
* Detail:
* Field |
* [Constr](#2s8eyo1) |
* [Method](#3rdcrjn)

Class DenseBag<T>

* java.lang.Object
  + java.util.AbstractCollection<T>
    - DenseBag<T>
* All Implemented Interfaces: java.io.Serializable, java.lang.Iterable<T>, java.util.Collection<T>  
    
  public class DenseBag<T>  
  extends java.util.AbstractCollection<T>  
  implements java.io.Serializable  
  The DenseBag class implements a Set-like collection that allows duplicates (a lot of them).  
  The DenseBag class provides Bag semantics: it represents a collection with duplicates. The "Dense" part of the class name comes from the fact that the class needs to efficiently handle the case where the bag contains 100,000,000 copies of a particular item (e.g., don't store 100,000,000 references to the item).  
  In a Bag, removing an item removes a single instance of the item. For example, a Bag b could contain additional instances of the String "a" even after calling b.remove("a").  
  The iterator for a dense bag must iterate over all instances, including duplicates. In other words, if a bag contains 5 instances of the String "a", an iterator will generate the String "a" 5 times.  
  In addition to the methods defined in the Collection interface, the DenseBag class supports several additional methods: uniqueElements, getCount, and choose.  
  The class extends AbstractCollection in order to get implementations of addAll, removeAll, retainAll and containsAll. (We will not be over-riding those). All other methods defined in the Collection interface will be implemented here.See Also: [Serialized Form](http://docs.google.com/serialized-form.html#DenseBag)

### Constructor SummaryConstructors

|  |
| --- |
| * + Constructor and Description |
| * + [DenseBag](http://docs.google.com/DenseBag.html#DenseBag--)() Initialize a new, empty DenseBag |

### Method SummaryAll Methods Instance Methods Concrete Methods

|  |  |
| --- | --- |
| * + Modifier and Type | * + Method and Description |
| * + boolean | * + [add](http://docs.google.com/DenseBag.html#add-T-)([T](http://docs.google.com/DenseBag.html) o) Adds an instance of o to the Bag |
| * + boolean | * + [addMany](http://docs.google.com/DenseBag.html#addMany-T-int-)([T](http://docs.google.com/DenseBag.html) o, int count) Adds multiple instances of o to the Bag. |
| * + [T](http://docs.google.com/DenseBag.html) | * + [choose](http://docs.google.com/DenseBag.html#choose-java.util.Random-)(java.util.Random r) Given a random number generator, randomly choose an element from the Bag according to the distribution of objects in the Bag (e.g., if a Bag contains 7 a's and 3 b's, then 70% of the time choose should return an a, and 30% of the time it should return a b. |
| * + boolean | * + [contains](http://docs.google.com/DenseBag.html#contains-java.lang.Object-)(java.lang.Object o) Returns true if the Bag contains one or more instances of o |
| * + boolean | * + [equals](http://docs.google.com/DenseBag.html#equals-java.lang.Object-)(java.lang.Object o) Tests if two DenseBags are equal. |
| * + int | * + [getCount](http://docs.google.com/DenseBag.html#getCount-java.lang.Object-)(java.lang.Object o) Return the number of instances of a particular object in the bag. |
| * + int | * + [hashCode](http://docs.google.com/DenseBag.html#hashCode--)() Return a hashCode that fulfills the requirements for hashCode (such as any two equal DenseBags must have the same hashCode) as well as desired properties (two unequal DenseBags will generally, but not always, have unequal hashCodes). |
| * + java.util.Iterator<[T](http://docs.google.com/DenseBag.html)> | * + [iterator](http://docs.google.com/DenseBag.html#iterator--)() Returns an iterator over the elements in a dense bag. |
| * + boolean | * + [remove](http://docs.google.com/DenseBag.html#remove-java.lang.Object-)(java.lang.Object o) Decrements the number of instances of o in the Bag. |
| * + int | * + [size](http://docs.google.com/DenseBag.html#size--)() Total number of instances of any object in the Bag (counting duplicates) |
| * + java.lang.String | * + [toString](http://docs.google.com/DenseBag.html#toString--)() Generate a String representation of the DenseBag. |
| * + java.util.Set<[T](http://docs.google.com/DenseBag.html)> | * + [uniqueElements](http://docs.google.com/DenseBag.html#uniqueElements--)() return a Set of the elements in the Bag (since the returned value is a set, it will contain one value for each UNIQUE value in the Bag). |

### Methods inherited from class java.util.AbstractCollectionaddAll, clear, containsAll, isEmpty, removeAll, retainAll, toArray, toArray

### Methods inherited from class java.lang.ObjectgetClass, notify, notifyAll, wait, wait, wait

### Methods inherited from interface java.util.CollectionparallelStream, removeIf, spliterator, stream

### Methods inherited from interface java.lang.IterableforEach

### Constructor Detail

#### DenseBag public DenseBag() Initialize a new, empty DenseBag

### Method Detail

#### toString public java.lang.String toString() Generate a String representation of the DenseBag. This will be useful for your own debugging purposes, but will not be tested other than to ensure that it does return a String and that two different DenseBags return two different Strings.Overrides: toString in class java.util.AbstractCollection<[T](http://docs.google.com/DenseBag.html)>

#### equals public boolean equals(java.lang.Object o) Tests if two DenseBags are equal. Two DenseBags are considered equal if they contain the same number of copies of the same elements. Comparing a DenseBag to an instance of any other class should return false;Specified by: equals in interface java.util.Collection<[T](http://docs.google.com/DenseBag.html)> Overrides: equals in class java.lang.Object

#### hashCode public int hashCode() Return a hashCode that fulfills the requirements for hashCode (such as any two equal DenseBags must have the same hashCode) as well as desired properties (two unequal DenseBags will generally, but not always, have unequal hashCodes).Specified by: hashCode in interface java.util.Collection<[T](http://docs.google.com/DenseBag.html)> Overrides: hashCode in class java.lang.Object

#### iterator public java.util.Iterator<[T](http://docs.google.com/DenseBag.html)> iterator() Returns an iterator over the elements in a dense bag. Note that if a Dense bag contains 3 a's, then the iterator must iterate over 3 a's individually.Specified by: iterator in interface java.lang.Iterable<[T](http://docs.google.com/DenseBag.html)> Specified by: iterator in interface java.util.Collection<[T](http://docs.google.com/DenseBag.html)> Specified by: iterator in class java.util.AbstractCollection<[T](http://docs.google.com/DenseBag.html)>

#### uniqueElements public java.util.Set<[T](http://docs.google.com/DenseBag.html)> uniqueElements() return a Set of the elements in the Bag (since the returned value is a set, it will contain one value for each UNIQUE value in the Bag).Returns: A set of elements in the Bag

#### getCount public int getCount(java.lang.Object o) Return the number of instances of a particular object in the bag. Return 0 if it doesn't exist at all.Parameters: o - object of interest Returns: number of times that object occurs in the Bag

#### choose public [T](http://docs.google.com/DenseBag.html) choose(java.util.Random r) Given a random number generator, randomly choose an element from the Bag according to the distribution of objects in the Bag (e.g., if a Bag contains 7 a's and 3 b's, then 70% of the time choose should return an a, and 30% of the time it should return a b. This operation can take time proportional to the number of unique objects in the Bag, but no more. This operation should not affect the Bag.Parameters: r - Random number generator Returns: randomly chosen element

#### contains public boolean contains(java.lang.Object o) Returns true if the Bag contains one or more instances of oSpecified by: contains in interface java.util.Collection<[T](http://docs.google.com/DenseBag.html)> Overrides: contains in class java.util.AbstractCollection<[T](http://docs.google.com/DenseBag.html)>

#### add public boolean add([T](http://docs.google.com/DenseBag.html) o) Adds an instance of o to the BagSpecified by: add in interface java.util.Collection<[T](http://docs.google.com/DenseBag.html)> Overrides: add in class java.util.AbstractCollection<[T](http://docs.google.com/DenseBag.html)> Returns: always returns true, since added an element to a bag always changes it

#### addMany public boolean addMany([T](http://docs.google.com/DenseBag.html) o, int count) Adds multiple instances of o to the Bag. If count is less than 0 or count is greater than 1 billion, throws an IllegalArgumentException.Parameters: o - the element to add count - the number of instances of o to add Returns: true, since addMany always modifies the DenseBag.

#### remove public boolean remove(java.lang.Object o) Decrements the number of instances of o in the Bag.Specified by: remove in interface java.util.Collection<[T](http://docs.google.com/DenseBag.html)> Overrides: remove in class java.util.AbstractCollection<[T](http://docs.google.com/DenseBag.html)> Returns: return true if and only if at least one instance of o exists in the Bag and was removed.

#### size public int size() Total number of instances of any object in the Bag (counting duplicates)Specified by: size in interface java.util.Collection<[T](http://docs.google.com/DenseBag.html)> Specified by: size in class java.util.AbstractCollection<[T](http://docs.google.com/DenseBag.html)>

[Skip navigation links](#1pxezwc)

* [Package](http://docs.google.com/package-summary.html)
* Class
* [Use](http://docs.google.com/class-use/DenseBag.html)
* [Tree](http://docs.google.com/package-tree.html)
* [Deprecated](http://docs.google.com/deprecated-list.html)
* [Index](http://docs.google.com/index-files/index-1.html)
* [Help](http://docs.google.com/help-doc.html)
* Prev Class
* Next Class
* [Frames](http://docs.google.com/index.html?DenseBag.html)
* [No Frames](http://docs.google.com/DenseBag.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* Field |
* [Constr](#3znysh7) |
* [Method](#2et92p0)
* Detail:
* Field |
* [Constr](#2s8eyo1) |
* [Method](#3rdcrjn)

[Web Accessibility](https://www.umd.edu/web-accessibility)