All Packages

Package Summary

Package	Description
Unnamed Package	
MyInterface	
MySource	

JavaDoc Help

- Navigation:
  - Search
- Kinds of Pages:
  - Overview • Package • Class or Interface • Other Files • Tree (Class Hierarchy)
  - Constant Field Values • All Packages • All Classes and Interfaces • Index

---

Navigation

Starting from the Overview page, you can browse the documentation using the links in each page, and in the navigation bar at the top of each page. The Index and Search box allow you to navigate to specific declarations and summary pages, including: All Packages, All Classes and Interfaces

Search

You can search for definitions of modules, packages, types, fields, methods, system properties and other terms defined in the API. These items can be searched using part or all of the name, optionally using "camelCase" abbreviations, or multiple search terms separated by whitespace. Some examples:

- `"j.l.obj"` matches "java.lang.Object"
- `"InpStr"` matches "java.io.InputStream"
- `"math exact long"` matches "java.lang.Math.absExact(long)"

Refer to the Javadoc Search Specification for a full description of search features.

Kinds of Pages

The following sections describe the different kinds of pages in this collection.

Overview

The Overview page is the front page of this API document and provides a list of all packages with a summary for each. This page can also contain an overall description of the set of packages.

Package

Each package has a page that contains a list of its classes and interfaces, with a summary for each. These pages may contain the following categories:

- Interfaces

- Classes
- Enum Classes
- Exception Classes
- Annotation Interfaces

Class or Interface

Each class, interface, nested class and nested interface has its own separate page. Each of these pages has three sections consisting of a declaration and description, member summary tables, and detailed member descriptions. Entries in each of these sections are omitted if they are empty or not applicable.

- Class Inheritance Diagram
- Direct Subclasses
- All Known Subinterfaces
- All Known Implementing Classes
- Class or Interface Declaration
- Class or Interface Description

- Nested Class Summary
- Enum Constant Summary
- Field Summary
- Property Summary
- Constructor Summary
- Method Summary
- Required Element Summary
- Optional Element Summary

- Enum Constant Details
- Field Details
- Property Details
- Constructor Details
- Method Details
- Element Details

Note: Annotation interfaces have required and optional elements, but not methods. Only enum classes have enum constants. The components of a record class are displayed as part of the declaration of the record class. Properties are a feature of JavaFX.

The summary entries are alphabetical, while the detailed descriptions are in the order they appear in the source code. This preserves the logical groupings established by the programmer.

Other Files

Packages and modules may contain pages with additional information related to the declarations nearby.

Tree (Class Hierarchy)

There is a Class Hierarchy page for all packages, plus a hierarchy for each package. Each hierarchy page contains a list of classes and a list of interfaces. Classes are organized by inheritance structure starting with `java.lang.Object`. Interfaces do not inherit from `java.lang.Object`.

- When viewing the Overview page, clicking on TREE displays the hierarchy for all packages.
- When viewing a particular package, class or interface page, clicking on TREE displays the hierarchy for only that package.

Constant Field Values

The Constant Field Values page lists the static final fields and their values.

All Packages

The All Packages page contains an alphabetic index of all packages contained in the documentation.

All Classes and Interfaces

The All Classes and Interfaces page contains an alphabetic index of all classes and interfaces contained in the documentation, including annotation interfaces, enum classes, and record classes.

Index

The Index contains an alphabetic index of all classes, interfaces, constructors, methods, and fields in the documentation, as well as summary pages such as All Packages, All Classes and Interfaces.

This help file applies to API documentation generated by the standard doclet.

Hierarchy For Unnamed Package

Package Hierarchies:
All Packages

Class Hierarchy

- java.lang.**Object**
  - **Test**

Constant Field Values

Contents

Unnamed Package

Test

Modifier and Type	Constant Field	Value
`public static final int`	`default_input_capacity`	`10`
`public static final int`	`number_of_test`	`5`

MyInterface.*

MyInterface.Javacontainer<E>

Modifier and Type	Constant Field	Value
`public static final int`	`initial_capacity`	`10`

Index

A C D E F G H I J L M N R S T
All Classes and Interfaces | All Packages | Constant Field Values

A

C

D

E

F

G

H

I

J

L

M

N

R

S

set_capacity(int) - Method in class MySource.JavaSet
Setter for capacityi

set_capacity(int) - Method in class MySource.JavaVector
Setter for capacity

size() - Method in interface MyInterface.Javacontainer
Returns the current size (number of elements) of the container / Getter.

size() - Method in class MySource.JavaSet
Returns the current size of the set.

size() - Method in class MySource.JavaVector
Returns the size of the vector.

T

Test - Class in Unnamed Package
This is the test code of the containers that I implemented for JavaSet and JavaVector , Note that not every extra implemented functions are used in testing , but you can try them whichever you want.

Test() - Constructor for class Test

toString() - Method in interface MyInterface.Javacontainer
Returns a string representation of the container.

toString() - Method in class MySource.JavaSet
It converts set to String format.

toString() - Method in class MySource.JavaVector
Returns a string representation of the vector.

Package MyInterface

Interface Javacontainer<E>

Type Parameters:

E - the type of elements in the container

All Known Implementing Classes:

JavaSet, JavaVector

```
public interface Javacontainer<E>
```

The Javacontainer interface defines common methods shared by both set and vector implementations in Java. * I have implement extra methods to make an improved and better homework , I hope it is not a problem. King regards.

Since:

January 18, 2024

Field Summary

Fields

Modifier and Type	Field	Description
`static final int`	**`initial_capacity`**	The initial capacity for containers

Method Summary

All Methods | **Instance Methods** | **Abstract Methods**

Modifier and Type	Method	Description
`void`	**`add`**`(E element)`	Adds the specified element to the container.
`int`	**`capacity`**`()`	Returns the current capacity of the container // Getter

		// Getter.
void	clear()	Clears the container by removing all elements.
boolean	equals(Object obj)	Checks if the container is equal to another object.
E	firstElement()	Returns the first element in the container.
E	get(int index)	Returns the element at the specified index in the container.
Myiterator<E>	getIterator()	Returns an iterator for sequential access to the elements in the container.
int	index_of(E element)	Returns the index of the specified element in the container.
boolean	is_find(E element)	Checks if the specified element is present in the container.
boolean	isEmpty()	Checks if the container is empty.
E	lastElement()	Returns the last element in the container.
void	load_file(String fileName)	Loads the container from a file with the specified file name.
void	remove(E element)	Removes the specified element from the container.
void	removeElementAt(int index)	Removes the element at the specified index from the container.
void	save_file()	Saves the current

		container to a file.
void	set_capacity (int _capacity)	Setter for capacity
int	size()	Returns the current size (number of elements) of the container / Getter.
String	toString()	Returns a string representation of the container.

Field Details

initial_capacity

```
static final int initial_capacity
```

The initial capacity for containers

See Also:

Constant Field Values

Method Details

add

```
void add(E element)
```

Adds the specified element to the container.

Parameters:

`element` - the element to be added to the container

remove

```
void remove(E element)
```

Removes the specified element from the container.

Parameters:

`element` - the element which will be removed from container.

size

```
int size()
```

Returns the current size (number of elements) of the container / Getter.

Returns:

the size of the container

getIterator

```
Myiterator<E> getIterator()
```

Returns an iterator for sequential access to the elements in the container. The iterator follows the Iterator design pattern, patterns were explained before the implementation which are : (`hasNext()`) and (`next()`).

Returns:

an iterator for the elements in the container

equals

```
boolean equals(Object obj)
```

Checks if the container is equal to another object.

Overrides:

`equals` in class `Object`

Parameters:

`obj` - the object to compare

Returns:

true if the containers are equal, false otherwise

toString

`String` `toString()`

Returns a string representation of the container.

Overrides:

`toString` in class `Object`

Returns:

a string representation of the container

removeElementAt

`void removeElementAt(int index)`

Removes the element at the specified index from the container.

Parameters:

`index` - the index of the element to be removed

clear

`void clear()`

Clears the container by removing all elements.

capacity

`int capacity()`

Returns the current capacity of the container // Getter.

Returns:

the capacity of the container

index_of

`int index_of(E element)`

Returns the index of the specified element in the container.

Parameters:

`element` - the element to find the index of

Returns:

the index of the element, or -1 if not found

is_find

```
boolean is_find(E element)
```

Checks if the specified element is present in the container.

Parameters:

`element` - the element to search for

Returns:

true if the element is found, false otherwise

isEmpty

```
boolean isEmpty()
```

Checks if the container is empty.

Returns:

true if the container is empty, false otherwise

firstElement

```
E firstElement()
```

Returns the first element in the container.

Returns:

the first element

lastElement

```
E lastElement()
```

Returns the last element in the container.

Returns:

the last element

get

```
E get(int index)
```

Returns the element at the specified index in the container.

Parameters:

`index` - the index of the element to be retrieved

Returns:

the element at the specified index

load_file

```
void load_file(String fileName)
```

Loads the container from a file with the specified file name.

Parameters:

`fileName` - the name of the file to load the container from

save_file

```
void save_file()
```

Saves the current container to a file.

set_capacity

```
void set_capacity(int _capacity)
```

Setter for capacity

Parameters:

`_capacity` - the specified capacity that is determined by user.

Package MySource

Class JavaSet<E>

java.lang.Object
MySource.JavaSet<E>

Type Parameters:

E - the type of elements in the set

All Implemented Interfaces:

Cloneable, Javacontainer<E>

```
public class JavaSet<E>
extends Object
implements Javacontainer<E>, Cloneable
```

My JavaSet generic class represents a Set collection of Java with various helper methods. Each method is documented with a Javadoc comment describing its functionality. Please read the Javadoc comments for details of any function. Generic type parameter:

Since:

Java 10.0, January 18, 2024

Field Summary

Fields inherited from interface MyInterface.Javacontainer

initial_capacity

Constructor Summary

Constructors

Constructor	Description
JavaSet()	Constructs an empty JavaSet with the default initial capacity which is defined in

	JavaContainer inteface.
`JavaSet(int _capacity)`	Constructs an empty JavaSet with the specified initial capacity.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
`void`	`add(E element)`	Add method adds element to the Setcontainer if it is not used before.
`int`	`capacity()`	Returns the current capacity of the set.
`void`	`clear()`	Clears the set , assignes size to 0.
`protected Object`	`clone()`	Implement deep copy of the JavaSet Using Cloneable interface.
`boolean`	`equals(Object obj)`	Checks if the current set is equal to another JavaSet.
`E`	`firstElement()`	Returns the first element in the set.
`E`	`get(int index)`	Returns the element at the specified index in the set.
`Myiterator<E>`	`getIterator()`	Return the iterator over the set elements.
`int`	`index_of(E element)`	Returns the index of the specified element in the set.

boolean	is_find(E element)	Checks if the specified element is exist in the set.
boolean	isEmpty()	Checks if the set is empty.
E	lastElement()	Returns the last element in the set.
void	load_file(String fileName)	Loads the set from the file with the specified file name as parameter..
void	remove(E element)	It removes the element which is send as parameter of the function.
void	removeElementAt (int index)	Removes the element at the specified index from the set.
void	save_file()	Saves set to the file.
void	set_capacity (int _capacity)	Setter for capacityi
int	size()	Returns the current size of the set.
String	toString()	It converts set to String format.

Methods inherited from class java.lang.Object

finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Details

JavaSet

```
public JavaSet()
```

Constructs an empty JavaSet with the default initial capacity which is defined in JavaContainer inteface.

JavaSet

```
public JavaSet(int _capacity)
```

Constructs an empty JavaSet with the specified initial capacity.

Parameters:

`_capacity` - the initial capacity of the set that user determines.

Throws:

`IllegalArgumentException` - if the specified initial capacity is less than 1 which is invalid.

Method Details

getIterator

```
public Myiterator<E> getIterator()
```

Return the iterator over the set elements.

Specified by:

`getIterator` in interface `Javacontainer<E>`

Returns:

an iterator for javaset.

set_capacity

```
public void set_capacity(int _capacity)
```

Setter for capacityi

Specified by:

`set_capacity` in interface `Javacontainer<E>`

Parameters:

`_capacity` - the specified value for capacity

add

```
public void add(E element)
```

Add method adds element to the Setcontainer if it is not used before. Using isfind method before add to check if it is already valid.

Specified by:

`add` in interface `Javacontainer<E>`

Parameters:

`element` - the element to be added to the set

remove

```
public void remove(E element)
```

It removes the element which is send as parameter of the function.

Specified by:

`remove` in interface `Javacontainer<E>`

Parameters:

`element` - element to be removed from set.

removeElementAt

```
public void removeElementAt(int index)
```

Removes the element at the specified index from the set.

Specified by:

`removeElementAt` in interface `Javacontainer<E>`

Parameters:

`index` - the index of the element which will be removed

Throws:

`IndexOutOfBoundsException` - if the index is out of range

`NoSuchElementException` - if the set is already empty

firstElement

```
public E firstElement()
```

Returns the first element in the set.

Specified by:

`firstElement` in interface `Javacontainer<E>`

Returns:

the first element

Throws:

`NoSuchElementException` - if the set is empty

lastElement

```
public E lastElement()
```

Returns the last element in the set.

Specified by:

`lastElement` in interface `Javacontainer<E>`

Returns:

the last element

Throws:

`NoSuchElementException` - if the set is empty

get

```
public E get(int index)
```

Returns the element at the specified index in the set.

Specified by:

`get` in interface `Javacontainer<E>`

Parameters:

`index` - the index of the element to be retrieved

Returns:

the element at the specified index

Throws:

`IndexOutOfBoundsException` - if the index is invalid value.

capacity

```
public int capacity()
```

Returns the current capacity of the set.

Specified by:

`capacity` in interface `Javacontainer<E>`

Returns:

current capacity of the set.

size

```
public int size()
```

Returns the current size of the set.

Specified by:

`size` in interface `Javacontainer<E>`

Returns:

current size of the set.

clear

```
public void clear()
```

Clears the set , assignes size to 0.

Specified by:

`clear` in interface `Javacontainer<E>`

Throws:

`NoSuchElementException` - if the set is already empty.

is_find

```
public boolean is_find(E element)
```

Checks if the specified element is exist in the set.

Specified by:

`is_find` in interface `Javacontainer<E>`

Parameters:

`element` - the element to be cheched if it is already exist.

Returns:

true if the element is found, false if it is not found.

index_of

```
public int index_of(E element)
```

Returns the index of the specified element in the set.

Specified by:

`index_of` in interface `Javacontainer<E>`

Parameters:

`element` - the element to find the index of

Returns:

the index of the element, or -1 if not found

isEmpty

```
public boolean isEmpty()
```

Checks if the set is empty.

Specified by:

`isEmpty` in interface `Javacontainer<E>`

Returns:

true if the set is empty, false otherwise

save_file

```
public void save_file()
```

Saves set to the file.

Specified by:

`save_file` in interface `Javacontainer<E>`

load_file

```
public void load_file(String fileName)
```

Loads the set from the file with the specified file name as parameter..

Specified by:

`load_file` in interface `Javacontainer<E>`

Parameters:

`fileName` - the name of the file to load the set from

equals

```
public boolean equals(Object obj)
```

Checks if the current set is equal to another JavaSet.

Specified by:

`equals` in interface `Javacontainer<E>`

Overrides:

`equals` in class `Object`

Parameters:

`obj` - the object to compare

Returns:

true if the sets are equal, false otherwise

clone

```
protected Object clone()
                throws CloneNotSupportedException
```

Implement deep copy of the JavaSet Using Cloneable interface.

Overrides:

`clone` in class `Object`

Returns:

the cloned JavaSet object.

Throws:

`CloneNotSupportedException` - if cloning is not supported

toString

```
public String toString()
```

It converts set to String format.

Specified by:

`toString` in interface `Javacontainer<E>`

Overrides:

`toString` in class `Object`

Returns:

a string representation of the set

Package MySource

Class JavaVector<E>

java.lang.Object
MySource.JavaVector<E>

Type Parameters:

E - the type of elements in the set

All Implemented Interfaces:

Cloneable, Javacontainer<E>

```
public class JavaVector<E>
extends Object
implements Javacontainer<E>, Cloneable
```

My JavaVector generic class represents a vector collectionof Java with various helper methods. Each method is documented with a Javadoc comment describing its functionality Please read the javadoc comments for details of any function. * Generic type parameter:

Since:

Java 10.0 , January 18, 2024

Field Summary

Fields inherited from interface MyInterface.Javacontainer
initial_capacity

Constructor Summary

Constructors

Constructor	Description
JavaVector()	Default constructor for JavaVector.
JavaVector(int capacity)	Constructor for JavaVector with a

`JavaVector(int capacity_)`	Constructor for JavaVector with a specified initial capacity.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
`void`	`add(E element)`	Adds the specified element to the vector.
`int`	`capacity()`	Returns the capacity of the vector.
`void`	`clear()`	Clears the vector by removing all elements.
`protected Object`	`clone()`	Implements the clone method for JavaVector.
`boolean`	`equals(Object obj)`	Checks if the current vector is equal to another JavaVector.
`E`	`firstElement()`	Returns the first element in the vector.
`E`	`get(int index)`	Returns the element at the specified index in the vector.
`Myiterator<E>`	`getIterator()`	Returns an iterator over the elements in the vector.
`int`	`index_of(E element)`	Returns the index of the specified element in the vector.
`boolean`	`is_find(E element)`	Checks if the specified element is present in the vector.

boolean	isEmpty()	Checks if the vector is empty.
E	lastElement()	Returns the last element in the vector.
void	load_file(String fileName)	Loads the vector from a file specified by the given file name.
void	remove(E element)	Removes the specified element from the vector.
void	removeElementAt(int index)	Removes the element at the specified index from the vector.
void	save_file()	Saves the vector to a file.
void	set_capacity(int _capacity)	Setter for capacity
int	size()	Returns the size of the vector.
String	toString()	Returns a string representation of the vector.

Methods inherited from class java.lang.Object

finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Details

JavaVector

```
public JavaVector()
```

Default constructor for JavaVector. Initializes an empty vector with the

default capacity.

JavaVector

```
public JavaVector(int capacity_)
```

Constructor for JavaVector with a specified initial capacity.

Parameters:

`capacity_` - the initial capacity of the vector

Throws:

`IllegalArgumentException` - if the specified initial capacity is less than 1

Method Details

getIterator

```
public Myiterator<E> getIterator()
```

Returns an iterator over the elements in the vector.

Specified by:

`getIterator` in interface `Javacontainer<E>`

Returns:

an iterator for JavaVector

set_capacity

```
public void set_capacity(int _capacity)
```

Setter for capacity

Specified by:

`set_capacity` in interface `Javacontainer<E>`

Parameters:

`_capacity` - the specified value for capacity

add

```
public void add(E element)
```

Adds the specified element to the vector.

Specified by:

`add` in interface `Javacontainer<E>`

Parameters:

`element` - the element to be added

remove

```
public void remove(E element)
```

Removes the specified element from the vector.

Specified by:

`remove` in interface `Javacontainer<E>`

Parameters:

`element` - the element to be removed

Throws:

`NoSuchElementException` - with proper warning message if there is no such an element.

removeElementAt

```
public void removeElementAt(int index)
```

Removes the element at the specified index from the vector.

Specified by:

`removeElementAt` in interface `Javacontainer<E>`

Parameters:

`index` - the index of the element to be removed

Throws:

`IllegalArgumentException` - if the index is invalid

clear

```
public void clear()
```

Clears the vector by removing all elements.

Specified by:

`clear` in interface `Javacontainer<E>`

firstElement

```
public E firstElement()
```

Returns the first element in the vector.

Specified by:

`firstElement` in interface `Javacontainer<E>`

Returns:

the first element

Throws:

`NoSuchElementException` - if the vector is empty

lastElement

```
public E lastElement()
```

Returns the last element in the vector.

Specified by:

`lastElement` in interface `Javacontainer<E>`

Returns:

the last element

Throws:

`NoSuchElementException` - if the vector is empty

get

```
public E get(int index)
```

Returns the element at the specified index in the vector.

Specified by:

`get` in interface `Javacontainer<E>`

Parameters:

`index` - the index of the element to be retrieved

Returns:

the element at the specified index

Throws:

`IndexOutOfBoundsException` - if the index is invalid

size

```
public int size()
```

Returns the size of the vector.

Specified by:

`size` in interface `Javacontainer<E>`

Returns:

the size of the vector

capacity

```
public int capacity()
```

Returns the capacity of the vector.

Specified by:

`capacity` in interface `Javacontainer<E>`

Returns:

the capacity of the vector

is_find

```
public boolean is_find(E element)
```

Checks if the specified element is present in the vector.

Specified by:

`is_find` in interface `Javacontainer<E>`

Parameters:

`element` - the element to search for

Returns:

true if the element is found, false otherwise

isEmpty

```
public boolean isEmpty()
```

Checks if the vector is empty.

Specified by:

`isEmpty` in interface `Javacontainer<E>`

Returns:

true if the vector is empty, false otherwise

index_of

```
public int index_of(E element)
```

Returns the index of the specified element in the vector.

Specified by:

`index_of` in interface `Javacontainer<E>`

Parameters:

`element` - the element to find the index of

Returns:

the index of the element, or -1 if not found

save_file

```
public void save_file()
```

Saves the vector to a file.

Specified by:

save_file in interface Javacontainer<E>

load_file

```
public void load_file(String fileName)
```

Loads the vector from a file specified by the given file name.

Specified by:

load_file in interface Javacontainer<E>

Parameters:

fileName - the name of the file from which to load the vector

clone

```
protected Object clone()
                throws CloneNotSupportedException
```

Implements the clone method for JavaVector.

Overrides:

clone in class Object

Returns:

the cloned JavaVector object

Throws:

CloneNotSupportedException - if cloning is not supported

equals

```
public boolean equals(Object obj)
```

Checks if the current vector is equal to another JavaVector.

Specified by:

equals in interface Javacontainer<E>

Overrides:

equals in class Object

Parameters:

`obj` - the object to compare

Returns:

true if the vectors are equal, false otherwise

toString

```
public String toString()
```

Returns a string representation of the vector.

Specified by:

`toString` in interface `Javacontainer<E>`

Overrides:

`toString` in class `Object`

Returns:

a string representation of the vector

Package MySource

```
package MySource
```

Classes

Class	Description
JavaSet<E>	My JavaSet generic class represents a Set collection of Java with various helper methods.
JavaVector<E>	My JavaVector generic class represents a vector collectionof Java with various helper methods.
Myiterator<E>	This inner iterator class is utilized for iterating over elements within the JavaSet and JavaVector classes.

Class Test

java.lang.Object
Test

```
public class Test
extends Object
```

This is the test code of the containers that I implemented for JavaSet and JavaVector , Note that not every extra implemented functions are used in testing , but you can try them whichever you want. I used some helper functions to make my test implementation clearer and more readible.

Field Summary

Fields

Modifier and Type	Field	Description
`static final int`	`default_input_capaci`	My final variables that I used them in test code.
`static final int`	`number_of_test`	

Constructor Summary

Constructors

Constructor	Description
`Test()`	

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`static void`	`main(String[] args)`	The main method that

static void	main(String[] args)	The main method that initiates the testing of JavaVector and JavaSet classes.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Details

default_input_capacity

```
public static final int default_input_capacity
```

My final variables that I used them in test code. `default_input_capacity` In any case of invalid capacity input of user, default will be used. ▶ invalid @code

See Also:

Constant Field Values

number_of_test

```
public static final int number_of_test
```

See Also:

Constant Field Values

Constructor Details

Test

```
public Test()
```

Method Details

main

```
public static void main(String[] args)
```

The main method that initiates the testing of JavaVector and JavaSet classes.

Parameters:

`args` - Command-line arguments (not used).

Class Test

java.lang.Object
Test

```
public class Test
extends Object
```

This is the test code of the containers that I implemented for JavaSet and JavaVector , Note that not every extra implemented functions are used in testing , but you can try them whichever you want. I used some helper functions to make my test implementation clearer and more readible.

Field Summary

Fields

Modifier and Type	Field	Description
`static final int`	`default_input_capaci`	My final variables that I used them in test code.
`static final int`	`number_of_test`	

Constructor Summary

Constructors

Constructor	Description
`Test()`	

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`static void`	`main(String[] args)`	The main method that

static void	main(String[] args)	The main method that initiates the testing of JavaVector and JavaSet classes.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Details

default_input_capacity

```
public static final int default_input_capacity
```

My final variables that I used them in test code. `default_input_capacity` In any case of invalid capacity input of user, default will be used. ▶ invalid @code

See Also:

Constant Field Values

number_of_test

```
public static final int number_of_test
```

See Also:

Constant Field Values

Constructor Details

Test

```
public Test()
```

Method Details

main

```
public static void main(String[] args)
```

The main method that initiates the testing of JavaVector and JavaSet classes.

Parameters:

`args` - Command-line arguments (not used).