All Packages

Package Summary

Package	Description
myContainerPackage	

All Classes

All Classes Interface Summary Class Summary

Class	Description
JavaContainer<T>	JavaContainer interface
JavaSet<T extends java.lang.Comparable<T>>	JavaSet class
JavaVector<T>	JavaVector class

Package myContainerPackage

Interface Summary

Interface	Description
JavaContainer<T>	JavaContainer interface

Class Summary

Class	Description
JavaSet<T extends java.lang.Comparable<T>>	JavaSet class
JavaVector<T>	JavaVector class

Hierarchy For Package myContainerPackage

Class Hierarchy

- java.lang.Object
  - myContainerPackage.**JavaSet**<T> (implements myContainerPackage.JavaContainer<T>)
  - myContainerPackage.**JavaVector**<T> (implements myContainerPackage.JavaContainer<T>)

Interface Hierarchy

- myContainerPackage.**JavaContainer**<T>

Hierarchy For All Packages

Package Hierarchies:
myContainerPackage

Class Hierarchy

- java.lang.Object
  - myContainerPackage.**JavaSet**<T> (implements myContainerPackage.JavaContainer<T>)
  - myContainerPackage.**JavaVector**<T> (implements myContainerPackage.JavaContainer<T>)

Interface Hierarchy

- myContainerPackage.**JavaContainer**<T>

Package myContainerPackage

Interface JavaContainer<T>

Type Parameters:

T - type of data It is a generic interface for JavaSet and JavaVector It has 4 methods: 1. Add(T n) : add an element to the container 2. Remove(T n) : remove an element from the container 3. Size() : return the size of the container 4. getIterator() : return an iterator of the container

All Known Implementing Classes:

JavaSet, JavaVector

```
public interface JavaContainer<T>
```

JavaContainer interface

Method Summary

All Methods Instance Methods Abstract Methods

Modifier and Type	Method	Description
boolean	Add(T n)	Add method
java.util.Iterator<T>	getIterator()	getIterator method
boolean	Remove(T n)	Remove method
int	Size()	Size method

Method Detail

Add

```
boolean Add(T n)
```

Add method

Parameters:

n - element to be added It adds the given element to the container

Parameters:

n - element to be removed It removes the given element from the container

Size

```
int Size()
```

Size method

Returns:

size of the container It returns the size of the container

getIterator

```
java.util.Iterator<T> getIterator()
```

getIterator method

Returns:

iterator of the container It returns an iterator of the container

Package myContainerPackage

Class JavaSet<T extends java.lang.Comparable<T>>

java.lang.Object
    myContainerPackage.JavaSet<T>

Type Parameters:

T - type of data It is a generic class for JavaSet. It has 15 methods: 1. Add(T n) : add an element to the container 2. Remove(T n) : remove an element from the container 3. Size() : return the size of the container 4. getCapacity() : return the capacity of the container 5. getData(int index) : return the data at the given index 6. isIn(T element) : return true if the element is in the container 7. getIterator() : return an iterator of the container 8. toString() : return a string representation of the container 9. equals(Object obj) : return true if the given object is equal to the container 10. JavaSet() : default constructor 11. JavaSet(int n) : constructor with capacity 12. JavaSet(JavaSet other) : copy constructor 13. IteratorImpl : private class for iterator 14. hasNext() : return true if the iterator has next element 15. next() : return the next element of the iterator

All Implemented Interfaces:

JavaContainer<T>

```
public class JavaSet<T extends java.lang.Comparable<T>>
extends java.lang.Object
implements JavaContainer<T>
```

JavaSet class

Constructor Summary

Constructors

Constructor	Description
JavaSet()	JavaSet constructor It creates a JavaSet object with default capacity 2
JavaSet(int n)	JavaSet constructor
JavaSet (JavaSet<T> other)	JavaSet constructor

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description

T	getData(int index)	getData() method
java.util.Iterator<T>	getIterator()	getIterator() method
boolean	isIn(T element)	isIn(T element) method
boolean	Remove(T n)	Remove(T n) method
int	Size()	Size() method
java.lang.String	toString()	toString() method

Methods inherited from class java.lang.Object

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

Constructor Detail

JavaSet

```
public JavaSet()
```

JavaSet constructor It creates a JavaSet object with default capacity 2

JavaSet

```
public JavaSet(int n)
```

JavaSet constructor

Parameters:

n - capacity of the JavaSet

Throws:

java.security.InvalidParameterException - if the given capacity is invalid It creates a JavaSet object with given capacity

JavaSet

`other` - JavaSet object It creates a JavaSet object with given JavaSet object

Method Detail

getData

```
public T getData(int index)
```

getData() method

Parameters:

`index` - index of the data

Returns:

data at the given index

Throws:

`java.security.InvalidParameterException` - if the given index is invalid It returns the data at the given index

Size

```
public int Size()
```

Size() method

Specified by:

`Size` in interface `JavaContainer<T extends java.lang.Comparable<T>>`

Returns:

size of the container It returns the size of the container

getCapacity

```
public int getCapacity()
```

getCapacity() method

Returns:

capacity of the container It returns the capacity of the container

getIterator() method

Specified by:

`getIterator` in interface `JavaContainer<T extends java.lang.Comparable<T>>`

Returns:

iterator of the container It returns the iterator of the container

Add

```
public boolean Add(T n)
```

Add(T n) method

Specified by:

`Add` in interface `JavaContainer<T extends java.lang.Comparable<T>>`

Parameters:

`n` - element to be added It adds the given element to the container It doubles the capacity if the size is equal to the capacity It adds the element to the container in order It shifts the elements after the given element It adds the element to the end if the element is the largest It increases the size by 1

Returns:

true if the element is added to the container

Throws:

`java.security.InvalidParameterException` - if the element is already in the container

Remove

```
public boolean Remove(T n)
```

Remove(T n) method

Specified by:

`Remove` in interface `JavaContainer<T extends java.lang.Comparable<T>>`

Parameters:

`n` - element to be removed It removes the given element from the container It halves the capacity if the size is equal to half of the capacity It removes the element from the container It shifts the elements after the given element It decreases the size by 1

Returns:

true if the element is removed from the container

Throws:

isIn

```
public boolean isIn(T element)
```

isIn(T element) method

Parameters:

`element` - element to be checked

Returns:

true if the element is in the container It returns true if the element is in the container

toString

```
public java.lang.String toString()
```

toString() method

Overrides:

`toString` in class `java.lang.Object`

Returns:

string representation of the container It returns a string representation of the container

equals

```
public boolean equals(java.lang.Object obj)
```

equals(Object obj) method

Overrides:

`equals` in class `java.lang.Object`

Parameters:

`obj` - object to be compared

Returns:

true if the given object is equal to the container It returns false if the given object is null or the given object is not a JavaSet object It returns false if the size of the given object is not equal to the size of the container It returns false if the elements of the given object are not equal to the elements of the container

Package myContainerPackage

Class JavaVector<T>

java.lang.Object
    myContainerPackage.JavaVector<T>

Type Parameters:

T - type of data It is a generic class for JavaVector. It has 15 methods: 1. Add(T n) : add an element to the container 2. Remove(T n) : remove an element from the container 3. Size() : return the size of the container 4. getCapacity() : return the capacity of the container 5. getData(int index) : return the data at the given index 6. isIn(T element) : return true if the element is in the container 7. getIterator() : return an iterator of the container 8. toString() : return a string representation of the container 9. equals(Object obj) : return true if the given object is equal to the container 10. JavaVector() : default constructor 11. JavaVector(int n) : constructor with capacity 12. JavaVector(JavaVector other) : copy constructor 13. IteratorImpl : private class for iterator 14. hasNext() : return true if the iterator has next element 15. next() : return the next element of the iterator

All Implemented Interfaces:

JavaContainer<T>

```
public class JavaVector<T>
extends java.lang.Object
implements JavaContainer<T>
```

JavaVector class

Constructor Summary

Constructors

Constructor	Description
JavaVector()	JavaVector constructor It creates a JavaVector object with default capacity 2
JavaVector(int n)	JavaVector constructor
JavaVector (JavaVector<T> other)	JavaVector constructor

Method Summary

All Methods Instance Methods Concrete Methods

int	**getCapacity**()	getCapacity method
T	**getData**(int index)	getData method
java.util.Iterator<**T**>	**getIterator**()	getIterator method
boolean	**isIn**(**T** element)	isIn method.
boolean	**Remove**(**T** element)	Remove method.
void	**setExactData**(int index, **T** newData)	getExactData method
int	**Size**()	Size method.
java.lang.String	**toString**()	next method

Methods inherited from class java.lang.Object

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

Constructor Detail

JavaVector

```
public JavaVector()
```

JavaVector constructor It creates a JavaVector object with default capacity 2

JavaVector

```
public JavaVector(int n)
```

JavaVector constructor

Parameters:

n - capacity of the JavaVector

Throws:

`java.security.InvalidParameterException` - if the given capacity is invalid It creates a JavaVector object with given capacity

JavaVector constructor

Parameters:

`other` - JavaVector object It creates a JavaVector object with given JavaVector object

Method Detail

getData

```
public T getData(int index)
```

getData method

Parameters:

`index` - index of the data

Returns:

data at the given index

Throws:

`java.security.InvalidParameterException` - if the given index is invalid It returns the data at the given index

setExactData

```
public void setExactData(int index, T newData)
```

getExactData method

Parameters:

`index` - index of the data

`newData` - new data of the given index

Throws:

`java.lang.RuntimeException` - if the given index is invalid It returns the data at the given index

getCapacity

capacity of the container It returns the capacity of the container

getIterator

```
public java.util.Iterator<T> getIterator()
```

getIterator method

Specified by:

`getIterator` in interface `JavaContainer<T>`

Returns:

iterator of the container It returns an iterator of the container

Add

```
public boolean Add(T element)
```

Add method.

Specified by:

`Add` in interface `JavaContainer<T>`

Parameters:

`element` - element to be added. It adds the given element to the container. If the size is equal to capacity, it doubles the capacity. If the size is equal to 0, it adds the element to the first index. Otherwise, it adds the element to the end of the container. It increases the size by 1. It gives a warning if the element is already in the container.

Returns:

true if the element is added to the container.

Remove

```
public boolean Remove(T element)
```

Remove method.

Specified by:

`Remove` in interface `JavaContainer<T>`

Parameters:

Returns:

true if the element is removed from the container.

Throws:

`java.security.InvalidParameterException` - if the element is not in the container.

`java.lang.ArithmeticException` - if the container is empty.

isIn

```
public boolean isIn(T element)
```

isIn method.

Parameters:

`element` - element to be checked.

Returns:

true if the element is in the container. It returns true if the element is in the container. Otherwise, it returns false.

Size

```
public int Size()
```

Size method.

Specified by:

`Size` in interface `JavaContainer<T>`

Returns:

size of the container. It returns the size of the container.

equals

```
public boolean equals(java.lang.Object obj)
```

hasNext method

Overrides:

`equals` in class `java.lang.Object`

Returns:

toString

```
public java.lang.String toString()
```

next method

Overrides:

`toString` in class `java.lang.Object`

Returns:

the next element of the iterator. It returns the next element of the iterator.

A

E

G

I

J

M

R

S

T

How This API Document Is Organized

This API (Application Programming Interface) document has pages corresponding to the items in the navigation bar, described as follows.

Package

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

- Interfaces
- Classes
- Enums
- Exceptions
- Errors
- Annotation Types

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 class/interface description, summary tables, and detailed member descriptions:

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

- Nested Class Summary
- Field Summary
- Property Summary
- Constructor Summary
- Method Summary

- Field Detail
- Property Detail
- Constructor Detail
- Method Detail

Each summary entry contains the first sentence from the detailed description for that item. 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.

Annotation Type

Each annotation type has its own separate page with the following sections:

- Annotation Type Declaration

Enum

Each enum has its own separate page with the following sections:

- Enum Declaration
- Enum Description
- Enum Constant Summary
- Enum Constant Detail

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.

Deprecated API

The Deprecated API page lists all of the API that have been deprecated. A deprecated API is not recommended for use, generally due to improvements, and a replacement API is usually given. Deprecated APIs may be removed in future implementations.

Index

The Index contains an alphabetic index of all classes, interfaces, constructors, methods, and fields, as well as lists of all packages and all classes.

All Classes

The All Classes link shows all classes and interfaces except non-static nested types.

Serialized Form

Each serializable or externalizable class has a description of its serialization fields and methods. This information is of interest to re-implementors, not to developers using the API. While there is no link in the navigation bar, you can get to this information by going to any serialized class and clicking "Serialized Form" in the "See also" section of the class description.

Constant Field Values

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

Search

You can search for definitions of modules, packages, types, fields, methods and other terms defined in the API, using some or all of the name. "Camel-case" abbreviations are supported: for example, "InpStr" will find "InputStream" and "InputStreamReader".

ALL CLASSES