

StorageResource

Summary

StorageResource in edu.duke

- You have a library of Iterable classes to support programming and problem-solving
 - Looping over data to solve problems



StorageResource in edu.duke

- You have a library of Iterable classes to support programming and problem-solving
 - Looping over data to solve problems
- Often storing intermediate data helps
 - Create an Iterable that you can use!
 - Dynamic, altered as a result of program control
- Results of information finding processed
 - After finding information, using methods

Reviewing class StorageResource

- Good review often includes documentation

Constructor Summary		
Constructors		
Constructor and Description		
StorageResource () Create an empty StorageResource object		
Method Summary		
All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method and Description	
void	add (java.lang.String s) Adds a string to this storage object	
void	clear () Remove all strings from this object so that .size() == 0	
boolean	contains (java.lang.String s) Determines if a string is stored in this object	
java.lang.Iterable<java.lang.String>	data () Create and return an iterable for all strings in this object	
int	size () Returns the number of string added/stored in this object	

Reviewing class StorageResource

- Good review often includes documentation
 - Ideally more than method summary, but usage

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

edu.duke

Class StorageResource

java.lang.Object
edu.duke.StorageResource

public class **StorageResource**
extends java.lang.Object

A class for storing and accessing String objects that mirrors an ArrayList in some functionality, but with a different model of creating and using iterables.

A StorageResource object supports adding strings using the .add() method, accessing via an iterator, and accessor methods .size() and .contains() whose functionality is outlined in this documentation.

As used in the Duke/Coursera course typical usage is

```
FileResource fr = new FileResource();  
StorageResource store = new StorageResource();  
for(String s : fr.words()){  
    store.add(s);  
}  
// can process store here, e.g.,  
int x = store.size(); // number of strings in store  
for(String s : store.data()){  
    // print or process s  
}
```

Reviewing class StorageResource

- Good review often includes documentation
 - Ideally more than method summary, but usage
 - Constructor

```
FileResource fr = new FileResource("/data/confucius.txt");
StorageResource unique = new StorageResource();
for(String w : fr.words()){
    if (! unique.contains(w)){
        unique.add(w);
    }
}
System.out.println("number of unique words: "+unique.size());
```

Reviewing class StorageResource

- Good review often includes documentation
 - Ideally more than method summary, but usage
 - Constructor
 - .add() and .contains()

```
FileResource fr = new FileResource("/data/confucius.txt");
StorageResource unique = new StorageResource();
for(String w : fr.words()){
    if (! unique.contains(w)){
        unique.add(w);
    }
}
System.out.println("number of unique words: "+unique.size());
```

Reviewing class StorageResource

- Good review often includes documentation
 - Ideally more than method summary, but usage
 - Constructor
 - .add() and .contains()
 - .size()

```
FileResource fr = new FileResource("/data/confucius.txt");
StorageResource unique = new StorageResource();
for(String w : fr.words()){
    if (! unique.contains(w)){
        unique.add(w);
    }
}
System.out.println("number of unique words: "+unique.size());
```