Package

csu.csci325

csu.csci325
Class CircularArrayQueue

```
java.lang.Object
  |
  +-csu.csci325.CircularArrayQueue
```

All Implemented Interfaces:

Queue

public class **CircularArrayQueue**
extends java.lang.Object
implements Queue

Constructor Summary

public	CircularArrayQueue()

Method Summary

java.lang.Object	dequeue()
void	enqueue(java.lang.Object element)
java.lang.Object	first()
boolean	isEmpty()
int	size()

Methods inherited from class `java.lang.Object`

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

Methods inherited from interface `csu.csci325.Queue`

`dequeue, enqueue, first, isEmpty, size, toString`

Constructors

CircularArrayQueue

`public CircularArrayQueue()`

Methods

(continued on next page)

(continued from last page)

enqueue

```
public void enqueue(java.lang.Object element)
```

dequeue

```
public java.lang.Object dequeue()
```

first

```
public java.lang.Object first()
```

isEmpty

```
public boolean isEmpty()
```

size

```
public int size()
```

csu.csci325
Class LinkedListQueue

```
java.lang.Object
   |
   +-csu.csci325.LinkedListQueue
```

All Implemented Interfaces:

Queue

public class **LinkedListQueue**
extends java.lang.Object
implements Queue

Constructor Summary

`public`	`LinkedListQueue()`

Method Summary

`java.lang.Object`	`dequeue()`
`void`	`enqueue(java.lang.Object element)`
`java.lang.Object`	`first()`
`boolean`	`isEmpty()`
`int`	`size()`

Methods inherited from class `java.lang.Object`

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

Methods inherited from interface `csu.csci325.Queue`

`dequeue, enqueue, first, isEmpty, size, toString`

Constructors

LinkedListQueue

`public LinkedListQueue()`

Methods

(continued from last page)

enqueue

```
public void enqueue(java.lang.Object element)
```

dequeue

```
public java.lang.Object dequeue()
```

first

```
public java.lang.Object first()
```

isEmpty

```
public boolean isEmpty()
```

size

```
public int size()
```

csu.csci325
Class LinkedListStack

```
java.lang.Object
  |
  +-csu.csci325.LinkedListStack
```

All Implemented Interfaces:

Stack

public class **LinkedListStack**
extends java.lang.Object
implements Stack

Constructor Summary

public	`LinkedListStack()`

Method Summary

boolean	`isEmpty()`
static void	`main(java.lang.String[] args)`
java.lang.Object	`peek()`
java.lang.Object	`pop()`
void	`push(java.lang.Object element)`
int	`size()`
java.lang.String	`toString()`

Methods inherited from class `java.lang.Object`

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

Methods inherited from interface `csu.csci325.Stack`

`isEmpty, peek, pop, push, size, toString`

Constructors

(continued from last page)

LinkedListStack

`public` **`LinkedListStack`**`()`

Methods

push

`public void` **`push`**`(java.lang.Object element)`

pop

`public java.lang.Object` **`pop`**`()`

peek

`public java.lang.Object` **`peek`**`()`

isEmpty

`public boolean` **`isEmpty`**`()`

size

`public int` **`size`**`()`

toString

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

main

`public static void` **`main`**`(java.lang.String[] args)`

csu.csci325
Class Maze

```
java.lang.Object
  |
  +-csu.csci325.Maze
```

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

Constructor Summary

public	`Maze()`

Method Summary

boolean	`canTraverse(int cxPos, int cyPos, int exPos, int eyPos)`
char[][]	`canTraverseSLL(char[][] maze, int cxPos, int cyPos, int exPos, int eyPos)` Discovers a path from (cxPos, cyPos) to (exPos, eyPos) and marks the path with 'p'.
boolean	`canTraverseSLL(int cxPos, int cyPos, int exPos, int eyPos)`
static void	`main(java.lang.String[] args)`
void	`printMaze()`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Maze

```
public Maze()
```

Methods

canTraverse

```
public boolean canTraverse(int cxPos,
                           int cyPos,
                           int exPos,
                           int eyPos)
```

(continued from last page)

canTraverseSLL

```
public char[][] canTraverseSLL(char[][] maze,
         int cxPos,
         int cyPos,
         int exPos,
         int eyPos)
```

Discovers a path from (cxPos, cyPos) to (exPos, eyPos) and marks the path with 'p'. This algorithm uses the iterative singly linked list stack implementation.

Parameters:

`maze` - the maze to search
`cxPos` - starting x coordinate
`cyPos` - starting y coordinate
`exPos` - ending x coordinate
`eyPos` - ending y coordinate

Returns:

true if a path exists, false otherwise.

canTraverseSLL

```
public boolean canTraverseSLL(int cxPos,
         int cyPos,
         int exPos,
         int eyPos)
```

printMaze

```
public void printMaze()
```

main

```
public static void main(java.lang.String[] args)
```

csu.csci325
Class Palindrome

```
java.lang.Object
  |
  +-csu.csci325.Palindrome
```

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

Constructor Summary

`public`	`Palindrome()`

Method Summary

`boolean`	`isPalindrome(java.lang.String str)` Returns if str is a palindrome (a sequence of characters that reads the same backward as forward) EX: isPalindrome("madam") -> true EX: isPalindrome("hello") -> false NOTE: You must use a queue and/or stack in your implementation!

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Palindrome

`public Palindrome()`

Methods

isPalindrome

`public boolean isPalindrome(java.lang.String str)`

Returns if str is a palindrome (a sequence of characters that reads the same backward as forward) EX: isPalindrome("madam") -> true EX: isPalindrome("hello") -> false NOTE: You must use a queue and/or stack in your implementation!

Parameters:

`str` - the string to test if it is a palindrome

Returns:

true if str is a palindrome, false otherwise.

csu.csci325
Interface Queue

All Known Implementing Classes:
LinkedListQueue, CircularArrayQueue

---

public interface **Queue**
extends

Method Summary

abstract java.lang.Object	`dequeue()` Removes and return the element at the head of the queue.
abstract void	`enqueue(java.lang.Object element)` Adds on element to the tail of the queue.
abstract java.lang.Object	`first()` Returns, without removing, the element at the head of the queue.
abstract boolean	`isEmpty()` Returns true if the queue contains no elements, false otherwise.
abstract int	`size()` Returns the number of element in the queue.
abstract java.lang.String	`toString()` Returns a string representation of the queue.

Methods

enqueue

```
public abstract void enqueue(java.lang.Object element)
```

Adds on element to the tail of the queue.

Parameters:
`element` - the element to be added to the tail of the queue

dequeue

```
public abstract java.lang.Object dequeue()
```

Removes and return the element at the head of the queue.

Returns:
the element at the front of the queue, or null if the queue is empty

(continued from last page)

first

```
public abstract java.lang.Object first()
```

Returns, without removing, the element at the head of the queue.

Returns:
the first element in queue, or null if the queue is empty

isEmpty

```
public abstract boolean isEmpty()
```

Returns true if the queue contains no elements, false otherwise.

Returns:
true if the queue contains no elements, false otherwise.

size

```
public abstract int size()
```

Returns the number of element in the queue.

Returns:
the number of element in the queue

toString

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

Returns a string representation of the queue.

Returns:
a string representation of the queue.

csu.csci325
Interface Stack

All Known Implementing Classes:
LinkedListStack

public interface **Stack**
extends

Method Summary

`abstract boolean`	`isEmpty()` Returns true fi the stack contains no elements, false otherwise.
`abstract java.lang.Object`	`peek()` Returns, without removing, the top element of the stack.
`abstract java.lang.Object`	`pop()` Removes and returns the top element from the stack.
`abstract void`	`push(java.lang.Object element)` Adds the specified element to the top of the stack.
`abstract int`	`size()` Returns the number of elements in the stack.
`abstract java.lang.String`	`toString()` Returns a string representation of the stack.

Methods

push

```
public abstract void push(java.lang.Object element)
```

Adds the specified element to the top of the stack.

Parameters:
`element` - element to be pushed onto the stack

pop

```
public abstract java.lang.Object pop()
```

Removes and returns the top element from the stack.

Returns:
the element removed from the stack

(continued from last page)

peek

`public abstract java.lang.Object **peek**()`

Returns, without removing, the top element of the stack.

Returns:
the element on top of the stack, or null if the stack is empty.

isEmpty

`public abstract boolean **isEmpty**()`

Returns true fi the stack contains no elements, false otherwise.

Returns:
tur if the stack is empty, false otherwise

size

`public abstract int **size**()`

Returns the number of elements in the stack.

Returns:
the number of elements in the stack

toString

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

Returns a string representation of the stack.

Returns:
a string representation of the stack.