Elijah Tay, Alex Stalica Lab 10

ArrayListQueue.java

package queue;

import java.util.\*;

public class ArrayListQueue<T> implements IQueue<T>

{

ArrayList<T> body;

public ArrayListQueue()

{

body = new ArrayList<T>();

}

public boolean isEmpty()

{

return (body.size()== 0);

}

public void insert(T item)

{

body.add(item);

}

public T delete()

{

if (isEmpty())

return null;

T item = body.get(0);

body.remove(0);

return item;

}

public T front()

{

if (isEmpty())

return null;

T item = body.get(0);

return item;

}

public void printQueue()

{

for(int i = 0; i < body.size(); i++)

{

System.out.print(body.get(i) + " ");

}

System.out.println();

}

/\*

public boolean equals(ArrayListQueue<T> otherQueue) {

if(this.size() == otherQueue.size()) {

for(int i = 0; i<body.size(); i++) {

T thisElement = body.get(i);

T otherElement = otherQueue.body.get(i);

if(thisElement != otherElement)

return false;

}

return true;

}

return false;

}

\*/

public boolean equals(ArrayListQueue<T> queue) {

if (this.size() != queue.size() )

return false;

for(int i = 0; i < this.size(); i++)

if (!body.get(i).equals(queue.body.get(i)))

return false;

return true;

}

public int size() {

int count = 0;

for(int i =0; i<body.size(); i++) {

count++;

}

return count;

}

}

ArrayListStack.java

package queue;

import java.util.\*;

/\*\*

\* Implementation of a stack using an ArrayList to hold the

\* elements of the stack

\* @author trao

\*

\* @param <T>

\*/

public class ArrayListStack<T> implements IStack<T>

{

//--------------------------------------------------

// Body of the stack is held in an ArrayList

ArrayList<T> stackBody;

//--------------------------------------------------

/\*\*

\* Constructor: Instantiates the body

\*/

public ArrayListStack()

{

stackBody = new ArrayList<T>();

}

//--------------------------------------------------

/\*\*

\* Test for emptiness: size == 0 means stack is empty

\*/

public boolean isEmpty()

{

return (stackBody.size() == 0);

}

//--------------------------------------------------

/\*\*

\* Insert a new item at the top of the stack

\*/

public void push(T item)

{

stackBody.add(item);

}

//--------------------------------------------------

/\*\*

\* Delete the top-most element of the stack

\* Report error and return null if stack is empty

\*/

public T pop()

{

if (isEmpty())

{

System.out.println("Error in ArrayStack.pop() Stack Empty ");

return null;

}

else

{

T topElement = stackBody.get(stackBody.size()-1);

stackBody.remove(stackBody.size()-1);

return topElement;

}

}

//--------------------------------------------------

/\*\*

\* Fetch the top-most element of the stack without

\* removing it. Report error and return null if stack

\* is empty

\*/

public T top()

{

if (isEmpty())

{

System.out.println("Error in ArrayStack.top() Stack Empty ");

return null;

}

else

{

T topElement = stackBody.get(stackBody.size()-1);

return topElement;

}

}

public boolean equals(ArrayListStack<T> otherStack) {

if(this.size() != otherStack.size())

return false;

for(int i = 0; i < this.size(); i++) {

if(!this.stackBody.get(i).equals(otherStack.stackBody.get(i)))

return false;

}

return true;

}

public int size() {

int count = 0;

for(int i = 0; i < stackBody.size(); i++) {

count++;

}

return count;

}

}

StackQueueTester.java

package queue;

import java.util.\*;

public class StackQueueTester {

public static void main(String[] args) {

ArrayListStack<String> s1 = new ArrayListStack<String>();

ArrayListStack<String> s2 = new ArrayListStack<String>();

ArrayListStack<String> s3 = new ArrayListStack<String>();

String dataForS1 = "Not everything that can be counted counts, and not everything that counts can be counted";

String dataForS2 = "Not everything that can be counted counts, and not everything that counts can be counted";

String dataForS3 = "Only two things are infinite, the universe and human stupidity, and I'm not sure about the former.";

ArrayList<String> S1 = new ArrayList<String>(Arrays.asList(dataForS1.split(" ")));

ArrayList<String> S2 = new ArrayList<String>(Arrays.asList(dataForS2.split(" ")));

ArrayList<String> S3 = new ArrayList<String>(Arrays.asList(dataForS3.split(" ")));

for(String str : S1) {

s1.push(str);

}

for(String str : S2) {

s2.push(str);

}

for(String str : S3) {

s3.push(str);

}

/\*----------------------------------------------------------------------------------------------------------------------\*/

ArrayListQueue<String> q1 = new ArrayListQueue<String>();

ArrayListQueue<String> q2 = new ArrayListQueue<String>();

ArrayListQueue<String> q3 = new ArrayListQueue<String>();

String dataForQ1 = "Not everything that can be counted counts, and not everything that counts can be counted";

String dataForQ2 = "Not everything that can be counted counts, and not everything that counts can be counted";

String dataForQ3 = "Only two things are infinite, the universe and human stupidity, and I'm not sure about the former.";

ArrayList<String> Q1 = new ArrayList<String>(Arrays.asList(dataForQ1.split(" ")));

ArrayList<String> Q2 = new ArrayList<String>(Arrays.asList(dataForQ2.split(" ")));

ArrayList<String> Q3 = new ArrayList<String>(Arrays.asList(dataForQ3.split(" ")));

for(String str : Q1) {

q1.insert(str);

}

for(String str : Q2) {

q2.insert(str);

}

for(String str : Q3) {

q3.insert(str);

}

/\*----------------------------------------------------------------------------------------------------------------------\*/

System.out.println("s1 size: " + s1.size());

System.out.println("s2 size: " + s2.size());

System.out.println("s3 size: " + s3.size());

System.out.println("q1 size: " + q1.size());

System.out.println("q2 size: " + q2.size());

System.out.println("q3 size: " + q3.size());

System.out.println("\n");

System.out.println("s1 equals s2: " + s1.equals(s2));

System.out.println("s1 equals s3: " + s1.equals(s3));

System.out.println("s2 equals s3: " + s2.equals(s3));

System.out.println("q1 equals q2: " + q1.equals(q2));

System.out.println("q1 equals q3: " + q1.equals(q3));

System.out.println("q2 equals q3: " + q2.equals(q3));

}

}

StackQueueTester.java Output

s1 size: 15

s2 size: 15

s3 size: 17

q1 size: 15

q2 size: 15

q3 size: 17

s1 equals s2: true

s1 equals s3: false

s2 equals s3: false

q1 equals q2: true

q1 equals q3: false

q2 equals q3: false