Suhyoon Bae

Lab 05

CIS 120 Data Structures

2020-03-12

Lab 05

Extending the Queue ADT at the Application Level

Page(s) File

1-2 Code printouts – Methods.java

3-4 Code printouts - ApplicationLevelMethods.java

5 Junit Test

**1. Code Printouts**

**1.1 Methods.java**

**package** ApplicationLevelMethods;

**import** queuePackage.LinkedUnbndQueue;

**public** **class** Methods {

// make a copy of orig queue, leaving it unchanged

**public** **static** LinkedUnbndQueue<String> copyQueue(LinkedUnbndQueue<String> origQ) {

LinkedUnbndQueue<String> q1, q2;

q1 = **new** LinkedUnbndQueue<String>();

q2 = **new** LinkedUnbndQueue<String>();

// copyQueue code goes here

**if** (origQ ==**null**) {

**return** **null**;

}

**else** {

**while** (!origQ.isEmpty()) {

String string = origQ.dequeue();

q1.enqueue(string);

q2.enqueue(string);

}

**while** (!q2.isEmpty()) {

origQ.enqueue(q2.dequeue());

}

**return** q1;

}

//copyStr=String.copyValuOf(tempStr)

}

// return a count of the number of elements of the orig queue, leaving it unchanged

**public** **static** **int** count(LinkedUnbndQueue<String> origQ) {

**int** count = 0;

**if** (origQ == **null**) {

**return** count;

} **else** **if** (origQ.isEmpty()) {

**return** count;

} **else** {

LinkedUnbndQueue<String> q1, q2;

q1 = **new** LinkedUnbndQueue<String>();

q2 = **new** LinkedUnbndQueue<String>();

// count code goes here

**while** (!origQ.isEmpty()) {

String string = origQ.dequeue();

q1.enqueue(string);

q2.enqueue(string);

count++;

}

**while** (!q2.isEmpty()) {

origQ.enqueue(q2.dequeue());

}

**return** count;

}

}

// return a nicely formatted string representing the queue, leaving it unchanged

**public** **static** String showQueue(LinkedUnbndQueue<String> origQ) {

String string;

**if** (origQ == **null**) {

string = "queue has a null reference";

}

**else** **if** (origQ.isEmpty()){

string = "empty queue";

//returnStr = "empty queue";

}

**else** {

// now handle the queues with meaningful data!

string = "front|";

LinkedUnbndQueue<String> q1;

q1 = **new** LinkedUnbndQueue<String>();

**while** (!origQ.isEmpty()) {

String element = origQ.dequeue();

q1.enqueue(element);

string = string + element + "|";

}

**while** (!q1.isEmpty()) {

origQ.enqueue(q1.dequeue());

}

string = string + "rear";

}

**return** string;

}

}

**1.2. AplicationLevelMethods.java**

**package** ApplicationLevelMethods;

**import** queuePackage.LinkedUnbndQueue;

**public** **class** Methods {

// make a copy of orig queue, leaving it unchanged

**public** **static** LinkedUnbndQueue<String> copyQueue(LinkedUnbndQueue<String> origQ) {

LinkedUnbndQueue<String> q1, q2;

q1 = **new** LinkedUnbndQueue<String>();

q2 = **new** LinkedUnbndQueue<String>();

// copyQueue code goes here

**if** (origQ ==**null**) {

**return** **null**;

}

**else** {

**while** (!origQ.isEmpty()) {

String string = origQ.dequeue();

q1.enqueue(string);

q2.enqueue(string);

}

**while** (!q2.isEmpty()) {

origQ.enqueue(q2.dequeue());

}

**return** q1;

}

//copyStr=String.copyValuOf(tempStr)

}

// return a count of the number of elements of the orig queue, leaving it unchanged

**public** **static** **int** count(LinkedUnbndQueue<String> origQ) {

**int** count = 0;

**if** (origQ == **null**) {

**return** count;

} **else** **if** (origQ.isEmpty()) {

**return** count;

} **else** {

LinkedUnbndQueue<String> q1, q2;

q1 = **new** LinkedUnbndQueue<String>();

q2 = **new** LinkedUnbndQueue<String>();

// count code goes here

**while** (!origQ.isEmpty()) {

String string = origQ.dequeue();

q1.enqueue(string);

q2.enqueue(string);

count++;

}

**while** (!q2.isEmpty()) {

origQ.enqueue(q2.dequeue());

}

**return** count;

}

}

// return a nicely formatted string representing the queue, leaving it unchanged

**public** **static** String showQueue(LinkedUnbndQueue<String> origQ) {

String string;

**if** (origQ == **null**) {

string = "queue has a null reference";

}

**else** **if** (origQ.isEmpty()){

string = "empty queue";

//returnStr = "empty queue";

}

**else** {

// now handle the queues with meaningful data!

string = "front|";

LinkedUnbndQueue<String> q1;

q1 = **new** LinkedUnbndQueue<String>();

**while** (!origQ.isEmpty()) {

String element = origQ.dequeue();

q1.enqueue(element);

string = string + element + "|";

}

**while** (!q1.isEmpty()) {

origQ.enqueue(q1.dequeue());

}

string = string + "rear";

}

**return** string;

}

}

**2. Junit Test**

