Lab104

Steven Glasford

2-19-2019

* Client.java

/\*\*

* A main class for the program .
* @author Steven Glasford

\*/

public class Client {

/\*\*

* A main class for the program .
* @param args none

\*/

public static void main ( String [] args ) {

int power = 8;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| long [][] | a S t a c k T i m e s = new | | | long [ power ][3]; |
| long [][] | l S t a c k T i m e s = new | | | long [ power ][3]; |
| long [][] | a Q u e u e T i m e s | = | new | long [ power ][3]; |
| long [][] | l Q u e u e T i m e s | = | new | long [ power ][3]; |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| long [][] | a L i s t Ti m e s = new long [ power ][3]; | | |  |
| // create | an | A r r a y S t a c k |  |  |
| A r r a y S ta c k | | astack = new A r r a y S t a ck ( 1 | 0 0 0 0 0 0 0 0 ) ; | |
| L i n k e d S t a c k | | lstack = new L i n k e d S t a c k | (); |  |
| A r r a y Q ue u e | | aqueue = new A r r a y Q u e ue ( 1 | 0 0 0 0 0 | 0 0 ) ; |
| L i n k e d Q u e u e | | lqueue = new L i n k e d Q u e u e | (); |  |

Ar r ayList alist = new A r rayList ( 1 0 0 0 0 0 0 0 0 ) ;

* start a timer

long start , stop ;

* a for loop to see what the push and pop times are for a thing

for ( int i = 0; i < power ; i ++){

a S t a c k T i m e s [ i ][0] = ( long ) Math . pow (10 ,( i ));

* Use the array stack method first start = System . nanoTime ();
* insert the number 0 ten times

for ( int j = 0; j < ( Math . pow (10 ,( i ))); j ++){ astack . push (0);

}

// end timer

stop = System . nanoTime ();

// save the time into the first part of the 2 d array

1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | a S t a c k T i m e s [ i ][1] | | = | ( long ) ( stop - start ); | | | |
|  | start = System . nanoTime (); | | | | |  |  |
|  | // remove the thing | | |  |  |  |  |
|  | for ( int j = 0; j < | | | Math . pow (10 ,( i )); j ++){ | | | |
|  | astack . pop (); | |  |  |  |  |  |
|  | } |  |  |  |  |  |  |
|  | // end | timer |  |  |  |  |  |
|  | stop = System . nanoTime (); | | | | |  |  |
|  | // save | the time into | | the first part | | | of the 2 d array |
|  | a S t a c k T i m e s [ i ][2] | | = | ( long ) | | ( stop - | start ); |
| } |  |  |  |  |  |  |  |
| // for the | | linked stack | |  |  |  |  |
| for | ( int i | = 0; i < power ; | | | i ++){ | |  |
|  | // Use | the linked | stack | | method first | |  |
|  | l S t a c k T i m e s [ i ][0] | | = | ( long ) | | Math . pow (10 ,( i )); | |
|  | start = System . nanoTime (); | | | | |  |  |
|  | // insert the number | | | 0 | ten | times |  |
|  | for ( int j = 0; j < | | | ( Math . pow (10 ,( i | | | ))); j ++){ |
|  | lstack . push (0); | | |  |  |  |  |
|  | } |  |  |  |  |  |  |
|  | // end | timer |  |  |  |  |  |
|  | stop = System . nanoTime (); | | | | |  |  |
|  | // save | the time into | | the first part | | | of the 2 d array |
|  | l S t a c k T i m e s [ i ][1] | | = | ( long ) ( stop - start ); | | | |
|  | start = System . nanoTime (); | | | | |  |  |
|  | // remove the thing | | |  |  |  |  |
|  | for ( int j = 0; j < | | | Math . pow (10 ,( i )); j ++){ | | | |
|  | lstack . pop (); | |  |  |  |  |  |
|  | } |  |  |  |  |  |  |
|  | // end | timer |  |  |  |  |  |
|  | stop = System . nanoTime (); | | | | |  |  |
|  | // save | the time into | | the first part | | | of the 2 d array |
|  | l S t a c k T i m e s [ i ][2] | | = | ( long ) | | ( stop - | start ); |
| } |  |  |  |  |  |  |  |
| // for the | | A r r a y Q u e u e |  |  |  |  |  |
| for | ( int i | = 0; i < power ; | | | i ++){ | |  |

a Q u e u e T i m e s [ i ][0] = ( long ) Math . pow (10 ,( i ));

* Use the array queue method first start = System . nanoTime ();
* insert the number 0 ten times

for ( int j = 0; j < ( Math . pow (10 ,( i ))); j ++){ aqueue . enqueue (0);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| } |  |  |  |  |  |
| // end timer | |  |  |  |  |
| stop = System . nanoTime (); | | | | |  |
| // save the time into | | | | the first part of | the 2 d array |
| a Q u e u e T i m e s [ i ][1] | | | = | ( long ) ( stop - start ); | |
| start = System . nanoTime (); | | | | |  |
| // remove | the | thing |  |  |  |
| for ( int | j = | 0; j | < | Math . pow (10 ,( i )); | j ++){ |

aqueue . dequeue ();

}

2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | // end | timer | |  |  |  |  |  |  |  |  |  |  |
|  | stop = System . nanoTime (); | | | | | |  |  |  |  |  |  |  |
|  | // save | the time | | into | the | | first | part | of the 2 d array | | | | |
|  | a Q u e u e T i m e s [ i ][2] = | | | | ( long ) ( stop - | | | | start ); | |  |  |  |
| } |  |  |  |  |  |  |  |  |  |  |  |  |  |
| for | ( int i | = | 0; i < | power ; | | i ++){ | |  |  |  |  |  |  |
|  | // Use | the | linked | queue | | method first | | |  |  |  |  |  |
|  | l Q u e u e T i m e s [ i ][0] = | | | | ( long ) Math . pow (10 ,( i )); | | | | | | |  |  |
|  | start = System . nanoTime (); | | | | | | |  |  |  |  |  |  |
|  | // insert | | the number | | 0 | ten | times |  |  |  |  |  |  |
|  | for ( int j = 0; | | | j < | ( Math . pow (10 ,( i ))); | | | | | j ++){ | |  |  |
|  | lqueue . enqueue (0); | | | | |  |  |  |  |  |  |  |  |
|  | } |  |  |  |  |  |  |  |  |  |  |  |  |
|  | // end | timer | |  |  |  |  |  |  |  |  |  |  |
|  | stop = System . nanoTime (); | | | | | |  |  |  |  |  |  |  |
|  | // save | the time | | into | the | | first | part | of | the | 2 | d | array |
|  | l Q u e u e T i m e s [ i ][1] = | | | | ( long ) ( stop - start ); | | | | | |  |  |  |
|  | start = System . nanoTime (); | | | | | | |  |  |  |  |  |  |
|  | // remove | | the thing | |  |  |  |  |  |  |  |  |  |
|  | for ( int j = 0; | | | j < | Math . pow (10 ,( i )); j ++){ | | | | | |  |  |  |
|  | lqueue . dequeue (); | | | | |  |  |  |  |  |  |  |  |
|  | } |  |  |  |  |  |  |  |  |  |  |  |  |
|  | // end | timer | |  |  |  |  |  |  |  |  |  |  |
|  | stop = System . nanoTime (); | | | | | |  |  |  |  |  |  |  |
|  | // save | the time | | into | the | | first | part | of the 2 d array | | | | |
|  | l Q u e u e T i m e s [ i ][2] = | | | | ( long ) ( stop - | | | | start ); | |  |  |  |
| } |  |  |  |  |  |  |  |  |  |  |  |  |  |
| // For the | | lists | |  |  |  |  |  |  |  |  |  |  |
| for | ( int i | = | 0; i < | power ; | | i ++){ | |  |  |  |  |  |  |
|  | // Use | the | linked | queue | | method first | | |  |  |  |  |  |
|  | a L i s t T im e s [ i ][0] | | | = ( long ) | | | Math . pow (10 ,( i )); | | | |  |  |  |
|  | start = System . nanoTime (); | | | | | | |  |  |  |  |  |  |
|  | // insert | | the number | | 0 | ten | times |  |  |  |  |  |  |
|  | for ( int j = 0; | | | j < | ( Math . pow (10 ,( i ))); | | | | | j ++){ | |  |  |
|  | alist . add (0); | | | |  |  |  |  |  |  |  |  |  |
|  | } |  |  |  |  |  |  |  |  |  |  |  |  |
|  | // end | timer | |  |  |  |  |  |  |  |  |  |  |
|  | stop = System . nanoTime (); | | | | | |  |  |  |  |  |  |  |
|  | // save | the time | | into | the | | first | part | of | the | 2 | d | array |
|  | a L i s t T im e s [ i ][1] | | | = ( long ) ( stop - start ); | | | | | | |  |  |  |
|  | start = System . nanoTime (); | | | | | | |  |  |  |  |  |  |
|  | // remove | | the thing | |  |  |  |  |  |  |  |  |  |
|  | for ( int j = 0; | | | j < | Math . pow (10 ,( i )); j ++){ | | | | | |  |  |  |
|  | alist . remove (); | | | |  |  |  |  |  |  |  |  |  |
|  | } |  |  |  |  |  |  |  |  |  |  |  |  |
|  | // end | timer | |  |  |  |  |  |  |  |  |  |  |
|  | stop = System . nanoTime (); | | | | | |  |  |  |  |  |  |  |
|  | // save | the time | | into | the | | first | part | of | the | 2 | d | array |
|  | a L i s t T im e s [ i ][2] | | | = ( long ) | | | ( stop | - start ); | | |  |  |  |
| } |  |  |  |  |  |  |  |  |  |  |  |  |  |

A s c i i T a b l e S t a c k ( aStackTimes , " A r r a y S t a c k " );

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

3

A s c i i T a b l e S t a c k ( lStackTimes , " Li n kStac k " );

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

A s c i i T a b l e Q u e u e ( aQueueTimes , " A r r a y Q u e u e " );

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

A s c i i T a b l e Q u e u e ( lQueueTimes , " L i n k e d Q u e u e " );

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

A s c i i T a b l e L i s t ( aListTimes , " Ar r ayList " );

}

* use the a standard of 80 c h aracter max for the table , because of latex public static void A s c i i T a b l e S t a c k ( long [][] data , String title ){

String table = String . format ( "+% -77s+" , "" ). replace ( " " , " -" );

table = table + "\n|" + center ( title ,77) + "|\n" ;

table = table + String . format ( "+% -12s+" , "" ). replace ( " " , " -" )

* String . format ( "% -32s+" , "" ). replace ( " " , " -" )
* String . format ( "% -31s+\n" , "" ). replace ( " " , " -" );

table = table + "|" + center ("N" ,12) + "|" + center ( " Push " , 32) + "|" + center ( "Pop " , 31) + "|\n" ;

table = table + String . format ( "+% -12s+" , "" ). replace ( " " , " -" )

* String . format ( "% -32s+" , "" ). replace ( " " , " -" )
* String . format ( "% -31s+\n" , "" ). replace ( " " , " -" );

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| for ( long [] | | data1 : data ) | | { |  |  |  |  |  |
| table | = | table | + "|" + | center ( Long . toString ( data1 [0]) ,12) | | | | + | "|" + |
|  |  | center ( Long . toString ( data1 [1]) , | | | 32) | + | "|" + |  |  |
|  |  | center ( Long . toString ( data1 [2]) , | | | 31) | + | "|\n"; |  |  |
| table | = | table | + String . format ( "+% -12s+" , "" ). replace (" | | | | | " , | " -") |

* String . format ( "% -32s+" , "" ). replace ( " " , " -" )
* String . format ( "% -31s+\n" , "" ). replace ( " " , " -" );

}

System . out . println ( table );

return ;

}

public static void A s c i i T a b l e Q u e u e ( long [][] data , String title ){ String table = String . format ( "+% -77s+" , "" ). replace ( " " , " -" );

table = table + "\n|" + center ( title ,77) + "|\n" ;

table = table + String . format ( "+% -12s+" , "" ). replace ( " " , " -" )

* String . format ( "% -32s+" , "" ). replace ( " " , " -" )
* String . format ( "% -31s+\n" , "" ). replace ( " " , " -" );

table = table + "|" + center ("N" ,12) + "|" + center ( " Enqueue " , 32) + "|" + center ( " Dequeue " , 31) + "|\n" ;

4

table = table + String . format ( "+% -12s+" , "" ). replace ( " " , " -" )

* String . format ( "% -32s+" , "" ). replace ( " " , " -" )
* String . format ( "% -31s+\n" , "" ). replace ( " " , " -" );

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| for ( long [] | | data1 : data ) | | { |  |  |  |  |  |
| table | = | table | + "|" + | center ( Long . toString ( data1 [0]) ,12) | | | | + | "|" + |
|  |  | center ( Long . toString ( data1 [1]) , | | | 32) | + | "|" + |  |  |
|  |  | center ( Long . toString ( data1 [2]) , | | | 31) | + | "|\n"; |  |  |
| table | = | table | + String . format ( "+% -12s+" , "" ). replace (" | | | | | " , | " -") |

* String . format ( "% -32s+" , "" ). replace ( " " , " -" )
* String . format ( "% -31s+\n" , "" ). replace ( " " , " -" );

}

System . out . println ( table );

return ;

}

public static void A s c i i T a b l e L i s t ( long [][] data , String title ){ String table = String . format ( "+% -77s+" , "" ). replace ( " " , " -" );

table = table + "\n|" + center ( title ,77) + "|\n" ;

table = table + String . format ( "+% -12s+" , "" ). replace ( " " , " -" )

* String . format ( "% -32s+" , "" ). replace ( " " , " -" )
* String . format ( "% -31s+\n" , "" ). replace ( " " , " -" );

table = table + "|" + center ("N" ,12) + "|" + center ( "Add " , 32) + "|" + center ( " Remove " , 31) + "|\n" ;

table = table + String . format ( "+% -12s+" , "" ). replace ( " " , " -" )

* + String . format ( "% -32s+" , "" ). replace ( " " , " -" )
  + String . format ( "% -31s+\n" , "" ). replace ( " " , " -" );
* add the data from the array into the table

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| for ( long [] | | data1 : data ) | | { |  |  |  |  |  |
| table | = | table | + "|" + | center ( Long . toString ( data1 [0]) ,12) | | | | + | "|" + |
|  |  | center ( Long . toString ( data1 [1]) , | | | 32) | + | "|" + |  |  |
|  |  | center ( Long . toString ( data1 [2]) , | | | 31) | + | "|\n"; |  |  |
| table | = | table | + String . format ( "+% -12s+" , "" ). replace (" | | | | | " , | " -") |

* String . format ( "% -32s+" , "" ). replace ( " " , " -" )
* String . format ( "% -31s+\n" , "" ). replace ( " " , " -" );

}

System . out . println ( table );

return ;

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| /\*\* |  |  |  |  |  |  |  |
| \* | Take a | string | and | center | it within a certain amount of given space . | | |
| \* | @param | text | The | text you | want | to center |  |
| \* | @param | len | The | possible | space | you want | to center within |

* @return

\*/

public static String center ( String text , int len ){

if ( len <= text . length ())

return text . substring (0 , len );

5

int before = ( len - text . length ())/2;

if ( before == 0)

return String . format ( "% - " + len + "s" , text );

int rest = len - before ;

return String . format ( "%" + before + "s% - " + rest + "s" , "" , text );

}

}

6

* ArrayList.java

/\*\*

\* Data S t r u c t ur e s & A l g o r i t hm s 6th Edition

\* Goodrick , Tamassia , G o l d w a s se r

\* Code F r agments 7.2 , 7.3 , 7.4 and 7.5

\*

\* An i m p l e m e n t a t i o n of a simple A r rayList class .

\* \*/

public class ArrayList <E > i m p l e m e n t s List <E > {

// instance v a riab les

// default array capacity

public static final int CAPACITY = 16;

* generic array used for storage private E [] data ;
* current number of elements private int size = 0;
* c o n s t r u c t o r s
  + c o n s t r u c t s list with default capacity public A r rayList () { this ( CAPACITY );}
* c o n s t r u c t s list with given capacity public A r rayList ( int capacity ){
  + safe cast ; compiler may give warning

data = ( E []) new Object [ capacity ];

}

* public methods /\*\*

\* Returns the number of elements in the array list . \*/

@O v erride

public int size () { return size ;}

/\*\*

* Returns whether the array list is empty .

\*/

@O v erride

public boolean isEmpty () { return size == 0;}

/\*\*

* Returns (but does not remove ) the element at index i.

\*/

@O v erride

public E get ( int i ) throws I n d e x O u t O f B o u n d s E x c e p t i o n { c h e c k I nd e x (i , size );

return data [ i ];

}

/\*\*

\* Replaces the element at index i with e, and returns the replaced

* element .

\*/

@O v erride

public E set ( int i , E e ) throws I n d e x O u t O f B o u n d s E x c e p t i o n { c h e c k I nd e x (i , size );

* temp = data [ i ]; data [ i ] = e ; return temp ;

}

7

/\*\*

\* Inserts element e to be at index i, shifting all s u b s e q u e n t

* elements later .

\*/

@O v erride

public void add ( int i , E e ) throws I n d e x O u t O f B o u n d s E x c e p t i o n ,

I l l e g a l S t a t e E x c e p t i o n {

c h e c k I nd e x (i , size + 1);

// not enough capacity

if ( size == data . length )

* + so double the current capacity resize (2\* data . length );
* start by shifting r i ghtmost

for ( int k = size - 1; k >= i ; k - -)

data [ k +1] = data [ k ];

* ready to place the new element size ++;

}

/\*\*

\* Removes / returns the element at index i, shifting s u b s e q u e n t

* elements earlier

\*/

@O v erride

public E remove ( int i ) throws I n d e x O u t O f B o u n d s E x c e p t i o n { c h e c k In d e x (i , size );

* temp = data [ i ];
  + shift elements to fill hole

for ( int k = i ; k < size - 1; k ++)

data [ k ] = data [ k +1];

* help gargbage c o l l e c t i o n data [ size -1] = null ;

size - -; return temp ;

}

* remove at the end of the thing public E remove (){

return remove ( size -1);

}

* add at the end of the thing public void add ( E e ){

add ( size , e );

}

* utility methods

/\*\*

\* Checks whether the given index is in the range [0 , n -1].

* @param i
* @param n

\*/

pr o tected void c h e c k I nd e x ( int i , int n ) throws I n d e x O u t O f B o u n d s E x c e p t i o n { if ( i < 0 || i >= n )

throw new I n d e x O u t O f B o u n d s E x c e p t i o n ( " Illegal index : " + i );

}

/\*\*

\* Resizes internal array to have given capacity >= size .

\* @param capacity

8

\*/

pr o tected void resize ( int capacity ){

* safe cast ; compiler may give warning E [] temp = ( E []) new Object [ capacity ]; for ( int k = 0; k < size ; k ++)

temp [ k ] = data [ k ];

* start using the new array

data = temp ;

}

}

9

* ArrayQueue.java

/\*\*

\* I m p l e m e n t a t i o n of the queue ADT using a fixed - length array

\*

* @author Michael T . Goodrich
* @author Roberto Tamassia
* @author Michael H . G o ldwater
* @author Steven Glasford
* @version 2 -21 -2019
* @param <E>

\* @todo figure out why the CAPACITY thing doesn 't work \*/

public class ArrayQueue <E > i m p l e m e n t s Queue <E >{

* instance v a riab les
  + generic array used for storage private E [] data ;
* index of the front element

private int f = 0;

* current number of elements private int sz = 0;
* default array capacity

public static final int CAPACITY = 1000;

* c o n s t r u c t o r s
  + c o n s t r u c t s queue with given default capacity public A r r a y Q u e u e () { this ( CAPACITY );}
* c o n s t r u c t s queue with given capacity

public A r r a y Q u e u e ( int capacity ){

data = ( E []) new Object [ capacity ];

}

// methods

/\*\*

* Returns the number of elements in the queue .

\*/

public int size () { return sz ;}

/\*\*

* Tests whether the queue is empty .

\*/

public boolean isEmpty () { return ( sz == 0);}

/\*\*

* Inserts an element at the rear of the queue .

\*/

public void enqueue ( E e ) throws I l l e g a l S t a t e E x c e p t i o n {

if ( sz == data . length ) throw new I l l e g a l S t a t e E x c e p t i o n (" Queue is full " ); // use modular a r i t h m e t i c

int avail = ( f + sz ) % data . length ;

data [ avail ] = e ;

sz ++;

}

/\*\*

\* Returns , but does not remove , the first element of the queue

* ( null if empty ).

\*/

public E first () {

if ( isEmpty ()) return null ;

10

return data [ f ];

}

/\*\*

* Removes and returns the first element of the queue ( null if empty ).

\*/

public E dequeue () {

if ( isEmpty ()) return null ;

* answer = data [ f ];
  + de r efence to help garbage c o lection data [ f ] = null ;

f = ( f + 1) % data . length ; sz - -;

return answer ;

}

}

11

* List.java

/\*\*

\* A s i m p l if i e d version of the " java . util . List " i n terface

\*

* @author Michael T . Goodrich
* @author Roberto Tamassia
* @author Michael H . G o ldwater
* @author Steven Glasford
* @version 2 -21 -2019
* @param <E>

\*/

public i n terface List <E > {

/\*\*

\* Returns the number of elements in this list .

* @return

\*/

int size ();

/\*\*

* Returns whether the list is empty
* @return

\*/

boolean isEmpty ();

/\*\*

\* Returns (but does not remove ) the element at index i.

* @param i
* @return

\*/

E get ( int i ) throws I n d e x O u t O f B o u n d s E x c e p t i o n ;

/\*\*

\* Replaces the element at index i with e, and returns the replaced

* element .
* @param i
* @param e
* @return

\*/

E set ( int i , E e ) throws I n d e x O u t O f B o u n d s E x c e p t i o n ;

/\*\*

\* Inserts element e to be at index i, shifting all s u b s e q u e n t

* elements later .
* @param i
* @param e

\*/

void add ( int i , E e ) throws I n d e x O u t O f B o u n d s E x c e p t i o n ;

/\*\*

\* Removes / returns the element at index i, shifting s u b s e q u e n t

* elements earlier .
* @param i
* @return

\*/

E remove ( int i ) throws I n d e x O u t O f B o u n d s E x c e p t i o n ;

}

12

* LinkedQueue.java

/\*\*

\* R e a l i z a t i o n of a FIFO queue as an i m p l e m e n t a t i o n of a S i n g l y L i n k e d S e t .

\*

* @author Michael T . Goodrich
* @author Roberto Tamassia
* @author Michael H . G o ldwater
* @author Steven Glasford
* @version 2 -21 -2019
* @param <E>

\*/

public class LinkedQueue <E > i m p l e m e n t s Queue <E > {

// an empty list

private final SinglyLinkedList <E > list = new SinglyLinkedList < >();

* new queue relies on the i n itially empty list public L i n k e d Q u e u e () {}

@O v erride

public int size () { return list . size ();}

@O v erride

public boolean isEmpty () { return list . isEmpty ();}

@O v erride

public void enqueue ( E element ) { list . addLast ( element );}

@O v erride

public E first () { return list . first ();}

@O v erride

public E dequeue () { return list . r e m o v e F i r s t ();}

}

13

* Queue.java

/\*\*

* @author Michael T . Goodrich
* @author Roberto Tamassia
* @author Michael H . G o ldwater
* @author Steven Glasford
* @version 2 -21 -2019
* @param <E>

\*/

public i n terface Queue <E > {

/\*\*

\* Returns the number of elements in the queue

* @return

\*/

int size ();

/\*\*

* Tests whether the queue is empty
* @return

\*/

boolean isEmpty ();

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| /\*\* |  |  |  |  |  |
| \* | Inserts | an element | at the rear | of the queue |  |
| \* | @param | e |  |  |  |
| \* | @todo | modify so | that this is | required to | throw a queue Full E x ception |

* if called on a full queue

\*/

void enqueue ( E e );

/\*\*

\* returns , but does not remove , the first element of the queue

* ( null if empty ).
* @return

\*/

E first ();

/\*\*

\* Removes and returns the first element of the queue ( null if empty )

* @return

\*/

E dequeue ();

}

14

* ArrayStack.java

/\*\*

\*

* @author Michael T . Goodrich
* @author Roberto Tamassia
* @author Michael H . G o ldwater
* @author Steven Glasford
* @version 2 -21 -2019
* @param <E>

\*/

public class ArrayStack <E > i m p l e m e n t s Stack <E > {

// default array capacity

public static final int CAPACITY = 1000;

* generic array used for storage private E [] data ;
* index of the top element in the stack private int t = -1;
* c o n s t r u c t s stack with default capacity public A r r a y S t a c k () { this ( CAPACITY );}
* c o n s t r u c t s stack with given capacity public A r r a y S t a c k ( int capacity ){
* safe cast ; compiler may give warning data = ( E []) new Object [ capacity ];

}

@O v erride

public int size () { return ( t +1);}

@O v erride

public boolean isEmpty () { return ( t == -1);}

@O v erride

public void push ( E e ) throws I l l e g a l S t a t e E x c e p t i o n {

if ( size () == data . length ) throw new I l l e g a l S t a t e E x c e p t i o n (" Stack "

* + "is full " );
* in c rement t before storing new item data [++ t ] = e ;

}

@O v erride

public E top (){

if ( isEmpty ()) return null ;

return data [ t ];

}

@O v erride

public E pop () {

if ( isEmpty ()) return null ;

* answer = data [ t ];
  + d e r e f e r e n c e to help garbage c o l l e c t i o n data [ t ] = null ;

t - -;

return answer ;

}

}

15

* LinkedStack.java

/\*\*

* @author Michael T . Goodrich
* @author Roberto Tamassia
* @author Michael H . G o ldwater
* @author Steven Glasford
* @version 2 -21 -2019
* @param <E>

\*/

public class LinkedStack <E > i m p l e m e n t s Stack <E > {

// an empty list

private final SinglyLinkedList <E > list = new SinglyLinkedList < >();

* new stack relies on the i n itially empty list public L i n k e d S t a c k () {}

@O v erride

public int size () { return list . size ();}

@O v erride

public boolean isEmpty () { return list . isEmpty ();}

@O v erride

public void push ( E element ) { list . addFirst ( element ); }

@O v erride

public E top () { return list . first (); }

@O v erride

public E pop () { return list . r e m o v e F i r s t (); }

}

16

* Stack.java

/\*\*

\* A c o l l e ct i o n of objects that are inserted and removed a c c ording to the \* last - in first - out p r inciple ; although similar in purpose , this

* i n terface differs from " java . util . Stack "
* @author Michael T . Goodrich
* @author Roberto Tamassia
* @author Michael H . G o ldwater

\*

* @version 2 -21 -2019 s

\*/

public i n terface Stack <E > {

/\*\*

\* Returns the number of elements in the stack

* @return number of elements in the stack

\*/

int size ();

/\*\*

* Tests whether the stack is empty .
* @return true if the stack is empty , false o t herwise .

\*/

boolean isEmpty ();

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| /\*\* |  |  |  |  |  |  |  |
| \* | Inserts | an element | at the | | top | of | the stack |
| \* | @param | e the element to | | be | inserted | | |
| \* | @todo | modify so | this | method | | is | required to trow a Stack Full |

* exception if called on a full stack

\*/

void push ( E e );

/\*\*

\* Returns , but does not remove , the element at the top of the stack

* @return to element in the stack (or null if empty )

\*/

E top ();

/\*\*

\* Removes and returns the top element from the stack .

* @return element removed (or null if empty )

\*/

E pop ();

}

17

10 SinglyLinkedList.java

/\*\*

\*

* S i n g l y L i n k e d L i s t Class
* Code F r agments 3.14 , 3.15
* from
* Data S t r u c t ur e s & Algorithms , 6th edition

\* by Michael T . Goodrich , Roberto Tamassia & Michael H. G o l d w a s s e r

* Wiley 2014
* T r a n s c r i b e d by
* @author Steven Glasford
* @version January 31 , 2019
* @param <E> a generic p l a c e h o l d e r name

\*/

public class SinglyLinkedList <E > {

/\*\*

\*

\* @param <E> a generic p l a c e h o l d e r name

\*

* A subclass creating the Node

\*/

private static class Node <E >{

* re f erence to the element stored at this node private final E element ;
* re f erence to the s u b s e q u e n t node in the list private Node <E > next ;

public Node ( E e , Node <E > n ){ element = e ;

next = n ;

}

/\*\*

\*

* @return Return the current element

\*/

public E g e t E l e m e nt (){ return element ;}

/\*\*

\*

* @return return the address of the next item in the linked list

\*/

public Node <E > getNext () { return next ;}

/\*\*

\*

* @param n the next item in the list

\*/

public void setNext ( Node <E > n ) { next = n ;}

}

* head node of the list (or null if empty ) private Node <E > head = null ;
* last node of the list (or null if empty ) private Node <E > tail = null ;
* number of nodes in the list

private int count = 0;

/\*\*

* c o n s t ru c t s an i n itially empty list

\*/

18

public S i n g l y L i n k e d L i s t (){}

* access methods /\*\*

\*

\* @return Return the size of the linked list \*/

public int size () { return count ;}

/\*\*

\*

* @return D e termine if the linked list is empty

\*/

public boolean isEmpty () { return count == 0;}

/\*\*

\*

\* @return return the first element in the list

\*

* returns (but does not remove ) the first element

\*/

public E first (){

if ( isEmpty ()) return null ;

return head . g e t E l e m e n t ();

}

/\*\*

\*

\* @return the last element in the linked list

\*

* returns (but does not remove the last element

\*/

public E last (){

if ( isEmpty ()) return null ;

return tail . g e t E l e m e n t ();

}

// update methods

/\*\*

\*

* @param e A generic element
* adds element e to the front of the list

\*/

public void addFirst ( E e ){

* create and link a new node head = new Node < >( e , head );
* special case : new node becomes tail also if ( count == 0)

tail = head ; count ++;

}

/\*\*

\*

* @param e A generic item
* adds element e to the end of the list

\*/

public void addLast ( E e ) {

19

* node will e v e n t u a l l y be the tail

Node <E > newest = new Node < >( e , null );

* special case : p r e v i o u s l y empty list if ( isEmpty ())

head = newest ;

else

tail . setNext ( newest ); tail = newest ;

count ++;

}

/\*\*

\*

\* @return return the item that was removed

\*

* removes and returns the first element

\*/

public E r e m o v e F i r s t (){

// nothing to remove

if ( isEmpty ()) return null ;

* answer = head . g e t E l e me n t ();
  + will become null if list had only one node head = head . getNext ();

count - -;

* + special case as list is now empty

if ( count == 0)

tail = null ;

return answer ;

}

}

20

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 11 | Output | |  |  |  |  |
| + - - | - - - - - - - - - - | - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | |  |  | A r r a y S t ac k |  |  | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | N | | | Push | | | Pop | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1 | | | 31908 | | | 3845 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10 | | | 3600 | | | 2754 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 100 | | | 22330 | | | 15529 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1000 | | | 427833 | | | 266389 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10000 | | | 1463029 | | | 1140004 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 100000 | | | 11006404 | | | 5365249 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1000000 | | | 69012080 | | | 1154951 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10000000 | | | 23738758 | | | 10577428 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| + - - | - - - - - - - - - - | - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | |  |  | LinkStack |  |  | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | N | | | Push | | | Pop | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1 | | | 475822 | | | 13075 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10 | | | 3221 | | | 4011 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 100 | | | 32825 | | | 41070 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1000 | | | 309093 | | | 448840 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10000 | | | 277333 | | | 232481 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 100000 | | | 1850521 | | | 438892 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1000000 | | | 15512319 | | | 4423880 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10000000 | | | 3 6 9 9 2 1 3 59 6 | | | 47636940 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| + - - | - - - - - - - - - - | - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | |  |  | A r r a y Q u eu e |  |  | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | N | | | Enqueue | | | Dequeue | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1 | | | 41097 | | | 15727 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10 | | | 2155 | | | 2221 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 100 | | | 14910 | | | 26210 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |

21

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| | | 1000 | | | 119292 | | | 191374 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10000 | | | 218049 | | | 211432 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 100000 | | | 1136130 | | | 826973 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1000000 | | | 9043308 | | | 7927018 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10000000 | | | 91833727 | | | 82702689 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| + - - | - - - - - - - - - - | - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | |  |  | L i n k e d Q u e u e |  |  | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | N | | | Enqueue | | | Dequeue | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1 | | | 50168 | | | 7429 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10 | | | 8595 | | | 993 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 100 | | | 30588 | | | 11056 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1000 | | | 326901 | | | 83288 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10000 | | | 547648 | | | 87950 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 100000 | | | 1665471 | | | 462619 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1000000 | | | 8243608 | | | 4648521 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10000000 | | | 2 5 7 0 5 6 6 51 0 | | | 46642004 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| + - - | - - - - - - - - - - | - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | |  |  | ArrayList |  |  | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | N | | | Add | | | Remove | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1 | | | 52038 | | | 8228 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10 | | | 3717 | | | 2677 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 100 | | | 29048 | | | 31450 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1000 | | | 291466 | | | 236959 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10000 | | | 409196 | | | 405911 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 100000 | | | 847668 | | | 398307 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 1000000 | | | 5943577 | | | 3813028 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |
| | | 10000000 | | | 56899101 | | | 38164740 | | |
| + - - | - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - - | + - - - - - - - - - - - | - - - - - - - - - - - - - - - - - - - - | + |

22