import java.util.Random;

/\*\*

\*

\* @author Samuel Swedberg

\* @version 9/9/22

\*

\* A client that demonstrates calling methods from the Scores class

\*/

public class Client {

public static void main(String[] args) {

Scores scores = new Scores(16); // Creates instance of scores

Random random = new Random(); // Creates instance of Random using java.util.Random

int i;

for(i=0; i<31; i++) // Generates random value from -10 to 10 and adds to Scores

{

int value = (random.nextInt(21)-10);

scores.add(value);

}

scores.toString(); // Prints contents of Scores

scores.add(6); // Adds 6 to Scores

System.out.println("scores.size() : " + scores.size()); // Prints size of Scores

scores.remove(); // Removes a random number from Scores

System.out.println("scores.size() after remove() : " + scores.size()); // Prints size of Scores

scores.toString(); // Prints contents of Scores to compare from before

int tempNumb = scores.get(15); // Stores the number from get(15) to be used in later

System.out.println("scores.get(15) : " + tempNumb); // Gets number at 15th index position

System.out.println("scores.getFrequencyOf(tempNumb) : " + scores.getFrequencyOf(tempNumb)); // Prints frequency of the tempNumb

scores.remove(scores.get(15)); // Removes first occurrence of the 15th index position

System.out.println("scores.size() after remove(15) : " + scores.size()); // Prints size of Scores

scores.toString(); // Prints contents of Scores to compare from before

System.out.println("scores.getFrequencyOf(scores.get(tempNumb)) : " + scores.getFrequencyOf(tempNumb)); // Prints frequency of tempNumb after removing the first occurrence

System.out.println("scores.contains(5) : " + scores.contains(5)); // Checks if Scores contains the number 5

}

}

import java.util.Random;

/\*\*

\*

\* @author Samuel Swedberg

\* @version 9/9/22

\*

\* The Scores class holds information related to accessing and holding scores

\*/

public class Scores {

private int list[], count, freqCount, i;

/\*\*

\* @param list

\*/

private Scores()

{

list = new int[2];

}

/\*\*

\*

\* @param value

\*/

public Scores( int value )

{

this.list = new int[value];

}

/\*\*

\*

\* @return count

\*/

public int size( ) { return count; }

/\*\*

\*

\* @return true if empty, false otherwise

\*/

public boolean isEmpty( )

{

if(size() == 0)

{

return true;

}

else

{

return false;

}

}

/\*\*

\*

\* @param num

\* @return count number of times num exists

\*/

public int getFrequencyOf( int num )

{

freqCount = 0;

for( i=0; i<list.length; i++)

{

if( num == list[i] )

{

freqCount++;

}

}

return freqCount;

}

/\*\*

\*

\* @param num

\* @return checks if list contains num

\*/

public boolean contains( int num )

{

boolean bool = false;

for(i=0; i<list.length; i++)

{

if( list[i] == num )

{

bool = true;

}

}

return bool;

}

/\*\*

\*

\* @param num add num to list

\*/

public void add ( int num )

{

if( size() == list.length )

{

int temp[] = new int[list.length\*2];

for(i=0; i<list.length; i++)

{

temp[i] = list[i];

}

list = temp;

temp = null;

list[size()] = num;

count++;

}

else

{

list[size()] = num;

count++;

}

}

/\*\*

\*

\* @param num

\* @return removes first occurence of num from list

\*/

public int remove ( int num )

{

boolean firstOccurence = false;

if( isEmpty() == false )

{

for(i=0; i<size()-1; i++)

{

if(num == list[i] && firstOccurence == false)

{

firstOccurence = true;

for(int x=i; x<size(); x++)

{

list[x] = list[x+1];

}

list[list.length-1] = 0;

--count;

}

}

}

return 0;

}

/\*\*

\*

\* @return removes random number from list, throws if list is empty

\*/

public int remove ( )

{

Random random = new Random();

if ( isEmpty() == false )

{

int rand = random.nextInt(size());

for( i=rand; i<size()-1; i++)

{

list[i] = list[i+1];

}

list[list.length-1] = 0;

--count;

}

else

{

throw new IllegalStateException("IllegalStateException: The remove() method cannot be called on an empty list");

}

return 0;

}

/\*\*

\*

\* @param i

\* @return returns number in i index of list, throws if i is out of bounds

\*/

public int get( int i )

{

if ( i > list.length)

{

throw new ArrayIndexOutOfBoundsException("ArrayIndexOutOfBoundsException: index is outside the bounds of the array.");

}

return list[i];

}

/\*\*

\*

\* @return contents of instance

\*/

public String toString( ) {

for(i=0;i<list.length;i++)

{

System.out.println(i + ": " + list[i]);

}

return null;

}

/\*\*

\*

\* @param o

\* @return true if equal, false otherwise

\*/

public boolean equals( Object o )

{

if ( !( o instanceof Scores ) )

return false;

Scores s = ( Scores ) o;

return super.equals( s )

&& list == s.list;

}

}

run:

0: 9

1: -9

2: 8

3: -1

4: -7

5: 8

6: 0

7: -8

8: -5

9: -3

10: 0

11: -3

12: -2

13: -7

14: 6

15: 1

16: -3

17: -9

18: -10

19: -1

20: 9

21: 2

22: 4

23: -8

24: 2

25: 6

26: 8

27: -10

28: -4

29: 5

30: 3

31: 0

scores.size() : 32

scores.size() after remove() : 31

0: 9

1: -9

2: 8

3: -1

4: -7

5: 8

6: 0

7: -8

8: -5

9: -3

10: 0

11: -3

12: -2

13: -7

14: 6

15: 1

16: -3

17: -10

18: -1

19: 9

20: 2

21: 4

22: -8

23: 2

24: 6

25: 8

26: -10

27: -4

28: 5

29: 3

30: 6

31: 0

scores.get(15) : 1

scores.getFrequencyOf(tempNumb) : 1

scores.size() after remove(15) : 30

0: 9

1: -9

2: 8

3: -1

4: -7

5: 8

6: 0

7: -8

8: -5

9: -3

10: 0

11: -3

12: -2

13: -7

14: 6

15: -3

16: -10

17: -1

18: 9

19: 2

20: 4

21: -8

22: 2

23: 6

24: 8

25: -10

26: -4

27: 5

28: 3

29: 6

30: 0

31: 0

scores.getFrequencyOf(scores.get(tempNumb)) : 0

scores.contains(5) : true

BUILD SUCCESSFUL (total time: 0 seconds)