**Project 1 DemoBigO.java GIVEN code**

**Fill in the code in the indicated places.**

/\*

\* Project 1 Big O study

\*/

package project3;

import java.util.Map;

import java.util.Random;

import java.util.TreeMap;

/\*\*

\*

\* @author lulofse

\*/

public class DemoBigO {

public TreeMap<Integer, Integer> tree = new TreeMap();

public boolean throwDivideException() throws ArithmeticException {

double one = 1.0, two = 2.0, zero = 0.0;

double result = one / two;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write the single line of code that will cause a Divide by Zero Exception to be thrown

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

return true;

}

public void throwAnException() {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write the single line of code that will throw an ArithmeticException and produce the given test output

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

/\*

\* An obvious O(1)

\*

\*/

public void demoBigO1(long n) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write the code for a O(1) method

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

/\*

\* A O( Log(n) ) method studied in class

\*\*\*\* GIVEN

\*/

public void demoBigOLogN(long n) {

int nInt = (int) n;

//int[] dummy= new int[nInt];

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

while (n > 1) {

n = n / 2;

}

}

return;

}

/\*

\* An obvious O(n)

\*

\*/

public void demoBigOn(long n) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write the code for a O(n) method

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

/\*

\* Fill a tree of size n with random integers

\*\*\*\* GIVEN

\*/

public void demoBigONLogNInit(long n) {

int randomInt;

long i = 0;

//System.out.println("n = "+n );

Random r = new Random();

while (i < n) {

randomInt = (int) r.nextInt((int) ((long) 2 \* n));

//System.out.println("i= "+i+" randomInt = "+randomInt );

tree.put(randomInt, randomInt);

i++;

}

}

/\*

\* A traversal of a tree of size n that we will learn about in 302

\*\*\*\* GIVEN

\*/

public void demoBigONLogN(long n) {

int i;

for (Map.Entry<Integer, Integer> entry : tree.entrySet()) {

Integer key = entry.getKey();

Integer value = entry.getValue();

};

}

/\*

\* An obvious O(n\*n)

\*

\*/

public void demoBigOnn(long n) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write the code for a O(n\*n) method

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

}