Java Mini Program Problems (Interview Prep)

Fizz Buzz Program

Ask the user for a number. Counting from zero (using a for loop!), replace every multiple of 3 with "fizz" and every multiple of 5 with "buzz". Multiples of BOTH should print out "fizz buzz". Every time you print out fizz, buzz, or fizz buzz - keep track. When the fizzes and buzzes and fizz buzzes reach the number provided by the user - stop! Finish it all up with a large, all caps printout of "TRADITION!!!!!" and do a little dance.

**What You Should See**

How many fizzing and buzzing units do you need in your life? 7  
0  
1  
2  
fizz  
4  
buzz  
fizz  
7  
8  
fizz  
buzz  
11  
fizz  
13  
14  
fizz buzz  
TRADITION!!!!!

public class TraditionalFizzBuzz {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("How many fizzing and buzzing units do you need in your life? ");

int num = sc.nextInt();

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

if (i % 3 == 0 && i % 5 == 0 && i != 0) {

System.out.println("fizz buzz");

} else if (i % 3 == 0 && i != 0) {

System.out.println("fizz");

} else if (i % 5 == 0 && i!= 0) {

System.out.println("buzz");

} else {

System.out.println(i);

}

}

System.out.println("TRADITION!!!!!");

}

}

--

public class AlarmClock {

// Given a day of the week encoded as

// 0=Sun, 1=Mon, 2=Tue, ...6=Sat, and a boolean indicating

// if we are on vacation, return a String of the form "7:00"

// indicating when the alarm clock should ring. Weekdays, the

// alarm should be "7:00" and on the weekend it should be "10:00".

// Unless we are on vacation -- then on weekdays it should be

// “10:00" and weekends it should be "off".

//

// alarmClock(1, false) → "7:00"

// alarmClock(5, false) → "7:00"

// alarmClock(0, false) → "10:00"

public String alarmClock(int day, boolean vacation) {

String result = "";

if (!vacation) {

if (day == 0 || day == 6) {

result = "10:00";

} else {

result = "7:00";

}

} else {

if (day == 0 || day == 6) {

result = "off";

} else {

result = "10:00";

}

}

return result;

}

}

--

public class CaughtSpeeding {

// You are driving a little too fast, and a police

// officer stops you. Write code to compute the

// result, encoded as an int value: 0=no ticket,

// 1=small ticket, 2=big ticket. If speed is 60 or

// less, the result is 0. If speed is between 61

// and 80 inclusive, the result is 1. If speed is

// 81 or more, the result is 2. Unless it is your

// birthday -- on that day, your speed can be 5

// higher in all cases.

//

// caughtSpeeding(60, false) → 0

// caughtSpeeding(65, false) → 1

//caughtSpeeding(65, true) → 0

public int caughtSpeeding(int speed, boolean isBirthday) {

int result = 0;

if (isBirthday) {

if (speed <= 65) {

result = 0;

} else if (speed > 65 && speed <= 85) {

result = 1;

} else {

result = 2;

}

} else {

if (speed <= 60) {

result = 0;

} else if (speed > 60 && speed <= 80) {

result = 1;

} else {

result = 2;

}

}

return result;

}

}

--

public class CommonEnd {

// Given 2 arrays of ints, a and b, return true if they

// have the same first element or they have the same

// last element. Both arrays will be length 1 or more.

//

// commonEnd({1, 2, 3}, {7, 3}) -> true

// commonEnd({1, 2, 3}, {7, 3, 2}) -> false

// commonEnd({1, 2, 3}, {1, 3}) -> true

public boolean commonEnd(int[] a, int[] b) {

int aFirst = a[0];

int aSize = a.length;

int aLast = a[aSize - 1];

int bFirst = b[0];

int bSize = b.length;

int bLast = b[bSize - 1];

if (aFirst == bFirst || aLast == bLast) {

return true;

} else {

return false;

}

}

}

--

public class DoubleX {

// Given a String, return true if the first instance

// of "x" in the String is immediately followed by

// another "x".

//

// doubleX("axxbb") -> true

// doubleX("axaxxax") -> false

// doubleX("xxxxx") -> true

public boolean doubleX(String str) {

boolean decision = false;

char[] ch = str.toCharArray();

for (int i = 0; i <= ch.length - 1; i++) {

if (ch[i] == 'x') {

if (ch[i + 1] == 'x') {

decision = true;

break;

} else {

decision = false;

break;

}

}

}

return decision;

}

}

--

public class EveryOther {

// Given a String, return a new String made of every other

// char starting with the first, so "Hello" yields "Hlo".

//

// everyOther("Hello") -> "Hlo"

// everyOther("Hi") -> "H"

// everyOther("Heeololeo") -> "Hello"

public String everyOther(String str) {

char[] ch = str.toCharArray();

String result = "";

for (int i = 0; i < ch.length; i++) {

if (i % 2 == 0) {

result += ch[i];

}

}

return result;

}

}

--

public class FirstHalf {

// Given a String of even length, return the first half.

// So the String "WooHoo" yields "Woo".

//

// firstHalf("WooHoo") -> "Woo"

// firstHalf("HelloThere") -> "Hello"

// firstHalf("abcdef") -> "abc"

public String firstHalf(String str) {

int length = str.length();

int halfLength = length / 2;

char[] ch = str.toCharArray();

String result = "";

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

String cToS = Character.toString(ch[i]);

result += cToS;

}

return result;

}

}

--

public class InsertWord {

// Given an "out" String length 4, such as "<<>>", and a

// word, return a new String where the word is in the middle

// of the out String, e.g. "<<word>>".

//

// Hint: SubStrings are your friend here

//

// insertWord("<<>>", "Yay") -> "<<Yay>>"

// insertWord("<<>>", "WooHoo") -> "<<WooHoo>>"

// insertWord("[[]]", "word") -> "[[word]]"

public String insertWord(String container, String word) {

String subBeginning = container.substring(0, 2);

String subEnding = container.substring(2);

return subBeginning + word + subEnding;

}

}

--

public class LongInMiddle {

// Given 2 Strings, a and b, return a String of the form

// short+long+short, with the shorter String on the outside

// and the longer String on the inside. The Strings will not

// be the same length, but they may be empty (length 0).

//

// longInMiddle("Hello", "hi") -> "hiHellohi"

// longInMiddle("hi", "Hello") -> "hiHellohi"

// longInMiddle("aaa", "b") -> "baaab"

public String longInMiddle(String a, String b) {

int aSize = a.length();

int bSize = b.length();

String result = "";

if (aSize > bSize) {

result = b + a + b;

} else if (bSize > aSize) {

result = a + b + a;

}

return result;

}

}

--

public class Makes10 {

// Given two ints, a and b, return true if one if them is 10

// or if their sum is 10.

//

// makes10(9, 10) -> true

// makes10(9, 9) -> false

// makes10(1, 9) -> true

public boolean makes10(int a, int b) {

int sum = a + b;

if (sum == 10 || a == 10 || b == 10) {

return true;

} else {

return false;

}

}

}

--

public class MultipleEndings {

// Given a String, return a new String made of 3 copies

// of the last 2 chars of the original String. The String

// length will be at least 2.

//

// multipleEndings("Hello") -> "lololo"

// multipleEndings("ab") -> "ababab"

// multipleEndings("Hi") -> "HiHiHi"

public String multipleEndings(String str) {

char[] ch = str.toCharArray();

int size = ch.length;

char last = ch[size - 1];

String lastString = Character.toString(last);

char nextToLast = ch[size - 2];

String nextToLastString = Character.toString(nextToLast);

return nextToLastString + lastString + nextToLastString + lastString + nextToLastString + lastString;

}

}

--

public class NearHundred {

// Given an int n, return true if it is within 10 of 100

// or 200.

// Hint: Check out the Math class for absolute value

//

// nearHundred(103) -> true

// nearHundred(90) -> true

// nearHundred(89) -> false

public boolean nearHundred(int n) {

int num = Math.abs(n);

if ((num >= 90 && num <= 110) || (num >=190 && num <= 210)) {

return true;

} else {

return false;

}

}

}

--

public class PosNeg {

// Given two int values, return true if one is negative and

// one is positive. Except if the parameter "negative" is

// true, then return true only if both are negative.

//

// posNeg(1, -1, false) -> true

// posNeg(-1, 1, false) -> true

// posNeg(-4, -5, true) -> true

public boolean posNeg(int a, int b, boolean negative) {

boolean result = false;

if (negative) {

if (a < 0 && b < 0) {

result = true;

} else {

result = false;

}

} else {

if (a < 0 || b < 0) {

result = true;

} else {

result = false;

}

}

return result;

}

}

--

public class RotateLeft {

// Given an array of ints, return an array with the elements

// “rotated left" so {1, 2, 3} yields {2, 3, 1}.

//

// rotateLeft({1, 2, 3}) -> {2, 3, 1}

// rotateLeft({5, 11, 9}) -> {11, 9, 5}

// rotateLeft({7, 0, 0}) -> {0, 0, 7}

public int[] rotateLeft(int[] numbers) {

int size = numbers.length;

int[] result = new int[size];

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

if (i == size - 1) {

result[i] = numbers[0];

} else {

result[i] = numbers[i + 1];

}

}

return result;

}

}

--

public class SkipSum {

// Given 2 ints, a and b, return their sum. However, sums

// in the range 10..19 inclusive are forbidden, so in that case just return 20.

//

// skipSum(3, 4) → 7

// skipSum(9, 4) → 20

// skipSum(10, 11) → 21

public int skipSum(int a, int b) {

int sum = a + b;

if (sum >= 10 && sum <= 19) {

sum = 20;

}

return sum;

}

}

--

public class Sum {

// Given an array of ints, return the sum of all the elements.

//

// sum({1, 2, 3}) -> 6

// sum({5, 11, 2}) -> 18

// sum({7, 0, 0}) -> 7

public int sum(int[] numbers) {

int sum = 0;

for (int i : numbers) {

sum += i;

}

return sum;

}

}

--

public class SpringSplosion {

// Given a non-empty String like "Code" return a String like

// “CCoCodCode". (first char, first two, first 3, etc)

//

// stringSplosion("Code") -> "CCoCodCode"

// stringSplosion("abc") -> "aababc"

// stringSplosion("ab") -> "aab"

public String stringSplosion(String str) {

char[] ch = str.toCharArray();

int size = ch.length;

String sub = "";

String result = "";

for (int i = 1; i <= size; i++) {

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

sub = Character.toString(ch[j]);

result += sub;

}

}

return result;

}

}

--

public class TrimOne {

// Given a String, return a version without the first and

// last char, so "Hello" yields "ell". The String length will be at least 2.

//

// trimOne("Hello") -> "ell"

// trimOne("java") -> "av"

// trimOne("coding") -> "odin"

public String trimOne(String str) {

int length = str.length();

String toPrint = str.substring(1, length - 1);

return toPrint;

}

}

--

public class CanHazTable {

// You and your date are trying to get a table at a restaurant.

// The parameter "you" is the stylishness of your clothes, in

// the range 0..10, and "date" is the stylishness of your date's

// clothes. The result getting the table is encoded as an int

// value with 0=no, 1=maybe, 2=yes. If either of you is very stylish,

// 8 or more, then the result is 2 (yes). With the exception that if

// either of you has style of 2 or less, then the result is 0 (no).

// Otherwise the result is 1 (maybe).

//

// canHazTable(5, 10) → 2

// canHazTable(5, 2) → 0

// canHazTable(5, 5) → 1

public int canHazTable(int yourStyle, int dateStyle) {

if (yourStyle >= 8 || dateStyle >= 8) {

if (yourStyle <= 2 || dateStyle <= 2) {

return 0;

} else {

return 2;

}

} else if (yourStyle <= 2 || dateStyle <=2) {

return 0;

} else {

return 1;

}

}

}

--

public class FirstLast6 {

// Given an array of ints, return true if 6 appears as either the

// first or last element in the array. The array will be length 1 or more.

//

// firstLast6({1, 2, 6}) -> true

// firstLast6({6, 1, 2, 3}) -> true

// firstLast6({13, 6, 1, 2, 3}) -> false

public boolean firstLast6(int[] numbers) {

int first = numbers[0];

int size = numbers.length;

int last = numbers[size - 1];

if (first == 6 || last == 6) {

return true;

} else {

return false;

}

}

}

--

public class FrontTimes {

// Given a String and a non-negative int n, we'll say that the

// front of the String is the first 3 chars, or whatever is there

// if the String is less than length 3. Return n copies of the front;

//

// frontTimes("Chocolate", 2) -> "ChoCho"

// frontTimes("Chocolate", 3) -> "ChoChoCho"

// frontTimes("Abc", 3) -> "AbcAbcAbc"

public String frontTimes(String str, int n) {

String partial = str.substring(0, 3);

String result = "";

for(int i = 1; i <= n; i++) {

result += partial;

}

return result;

}

}

--

public class GreatParty {

// When squirrels get together for a party, they like to have cigars.

// A squirrel party is successful when the number of cigars is between

// 40 and 60, inclusive. Unless it is the weekend, in which case there

// is no upper bound on the number of cigars. Return true if the party

// with the given values is successful, or false otherwise.

// greatParty(30, false) → false

// greatParty(50, false) → true

// greatParty(70, true) → true

public boolean greatParty(int cigars, boolean isWeekend) {

if(isWeekend) {

if(cigars >= 40) {

return true;

} else {

return false;

}

} else {

if(cigars >= 40 && cigars <= 60) {

return true;

} else {

return false;

}

}

}

}

--

public class SameFirstLast {

// Given an array of ints, return true if the array is length

// 1 or more, and the first element and the last element are equal.

//

// sameFirstLast({1, 2, 3}) -> false

// sameFirstLast({1, 2, 3, 1}) -> true

// sameFirstLast({1, 2, 1}) -> true

public boolean sameFirstLast(int[] numbers) {

if (numbers.length > 1) {

int first = numbers[0];

int size = numbers.length;

int last = numbers[size - 1];

if (first == last) {

return true;

} else {

return false;

}

} else {

return false;

}

}

}

--

public class SayHi {

//Given a String name, like "Bob," output a greeting with that name

public String sayHi(String name) {

return "Hello " + name + "!";

}

}

--

public class SleepingIn {

// The parameter weekday is true if it is a weekday, and the

// parameter vacation is true if we are on vacation. We sleep

// in if it is not a weekday or we're on vacation. Return true

// if we sleep in.

//

// canSleepIn(false, false) -> true

// canSleepIn(true, false) -> false

// canSleepIn(false, true) -> true

public boolean canSleepIn(boolean isWeekday, boolean isVacation) {

if (!isWeekday || isVacation) {

return true;

} else {

return false;

}

}

}

--

public class abba {

// Given two Strings, a and b, return the result of putting

// them together in the order abba, e.g. "Hi" and "Bye"

// returns "HiByeByeHi".

//

// abba("Hi", "Bye") -> "HiByeByeHi"

// abba("Yo", "Alice") -> "YoAliceAliceYo"

// abba("What", "Up") -> "WhatUpUpWhat"

public String abba(String a, String b) {

return "" + a + b + b + a + "";

}

}

--

public class areWeInTrouble {

// We have two children, a and b, and the parameters aSmile and

// bSmile indicate if each is smiling. We are in trouble if they

// are both smiling or if neither of them is smiling. Return true

// if we are in trouble.

//

// areWeInTrouble(true, true) -> true

// areWeInTrouble(false, false) -> true

// areWeInTrouble(true, false) -> false

public boolean areWeInTrouble(boolean aSmile, boolean bSmile) {

if(aSmile && bSmile || !aSmile && !bSmile) {

return true;

} else {

return false;

}

}

}

--

public class stringTimes {

// Given a String and a non-negative int n, return a larger String

// that is n copies of the original String.

//

// stringTimes("Hi", 2) -> "HiHi"

// stringTimes("Hi", 3) -> "HiHiHi"

// stringTimes("Hi", 1) -> "Hi"

public String stringTimes(String str, int n) {

String result = "";

for(int i = 1; i <= n; i++) {

result += str;

}

return result;

}

}