IMI WiSe 2019 / Informatik 2 / Group 1

Students :Pavel Tsvyatkov

Email : s0559632@htw-berlin.de

Instructor: Prof. Dr. Debora Weber-Wulff

Laboratory Report

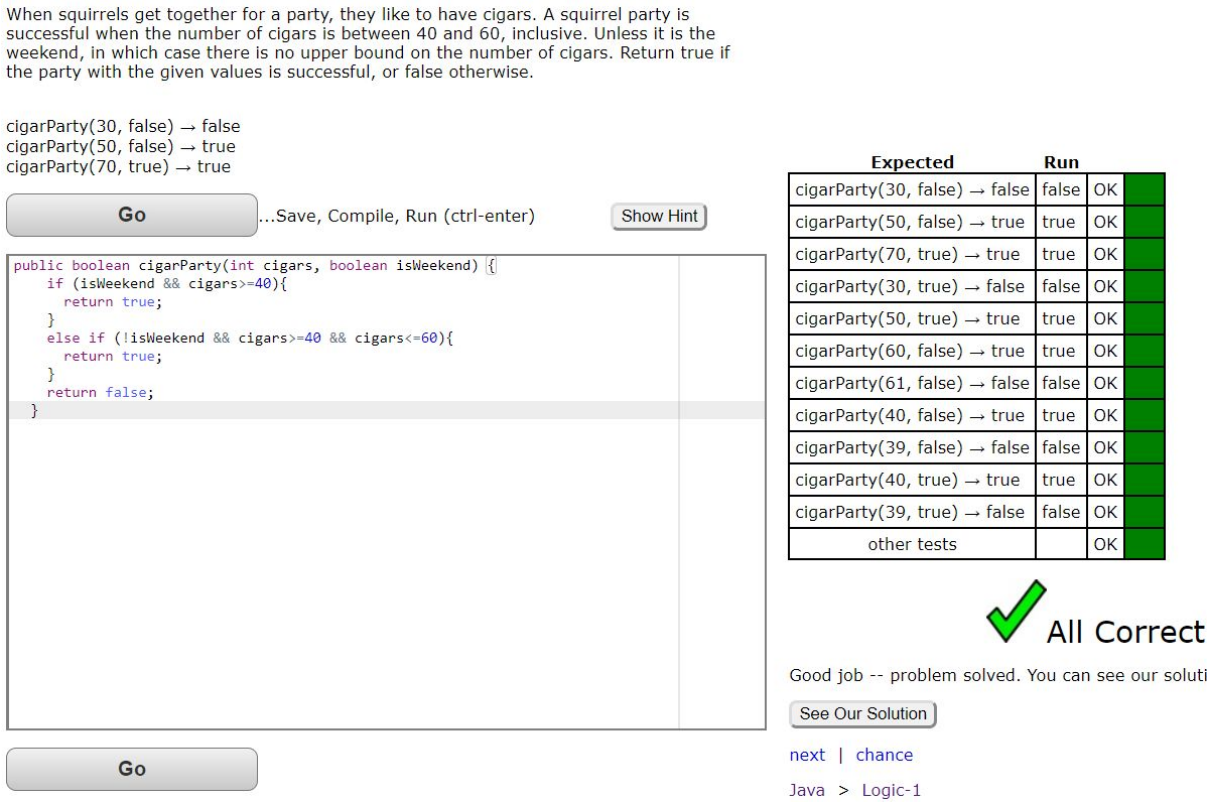
Exercise 4: Abstract Data Types

Assignment

We are using Nick Parlente's Coding Bat for the first set of exercises. Please document which ones you did and record any problems you encountered solving the.

1. Choose three exercises from the **simple logic** puzzles. Record the resulting code in your report.

The first exercise i chose from the simple logic puzzles was cigarParty.



As a second exercise i chose 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"

Go ...Save, Compile, Run (ctrl-enter)

```
public String alarmClock(int day, boolean vacation) {
    String alarm = "7:00";
    String alarmWeekend = "10:00";
    String alarmOff = "off";
    if (!vacation){
      if(day == 1 || day == 2 || day == 3 || day == 4 || day == 5 ){
        return alarm;
      }
      else{
        return alarmWeekend;
      }
    }
    else{
      if(day == 1 || day == 2 || day == 3 || day == 4 || day == 5 ){
        return alarmWeekend;
      }
      else {
        return alarmOff;
    }
  }
  }
```

Expected	Run		
alarmClock(1, false) → "7:00"	"7:00"	OK	
alarmClock(5, false) → "7:00"	"7:00"	OK	
alarmClock(0, false) → "10:00"	"10:00"	OK	
alarmClock(6, false) → "10:00"	"10:00"	OK	
alarmClock(0, true) → "off"	"off"	OK	
alarmClock(6, true) → "off"	"off"	OK	
alarmClock(1, true) → "10:00"	"10:00"	OK	
alarmClock(3, true) → "10:00"	"10:00"	OK	
alarmClock(5, true) → "10:00"	"10:00"	OK	
other tests		OK	

All Correct

next | chance

Java > Logic-1

The third exercise i chose was twoAsOne.

Given three ints, a b c, return true if it is possible to add two of the ints to get the third.

twoAsOne(1, 2, 3) → true
twoAsOne(3, 1, 2) → true
twoAsOne(3, 2, 2) → false

Go ...Save, Compile, Run (ctrl-enter)

```
public boolean twoAsOne(int a, int b, int c) {
    if(a+b == c || a+c == b || b+c == a){
      return true;
    }
    return false;
  }
```

Go

Expected	Run		
twoAsOne(1, 2, 3) → true	true	OK	
twoAsOne(3, 1, 2) → true	true	OK	
twoAsOne(3, 2, 2) → false	false	OK	
twoAsOne(2, 3, 1) → true	true	OK	
twoAsOne(5, 3, -2) → true	true	OK	
twoAsOne(5, 3, -3) → false	false	OK	
twoAsOne(2, 5, 3) → true	true	OK	
twoAsOne(9, 5, 5) → false	false	OK	
twoAsOne(9, 4, 5) → true	true	OK	
twoAsOne(5, 4, 9) → true	true	OK	
twoAsOne(3, 3, 0) → true	true	OK	
twoAsOne(3, 3, 2) → false	false	OK	
other tests		OK	

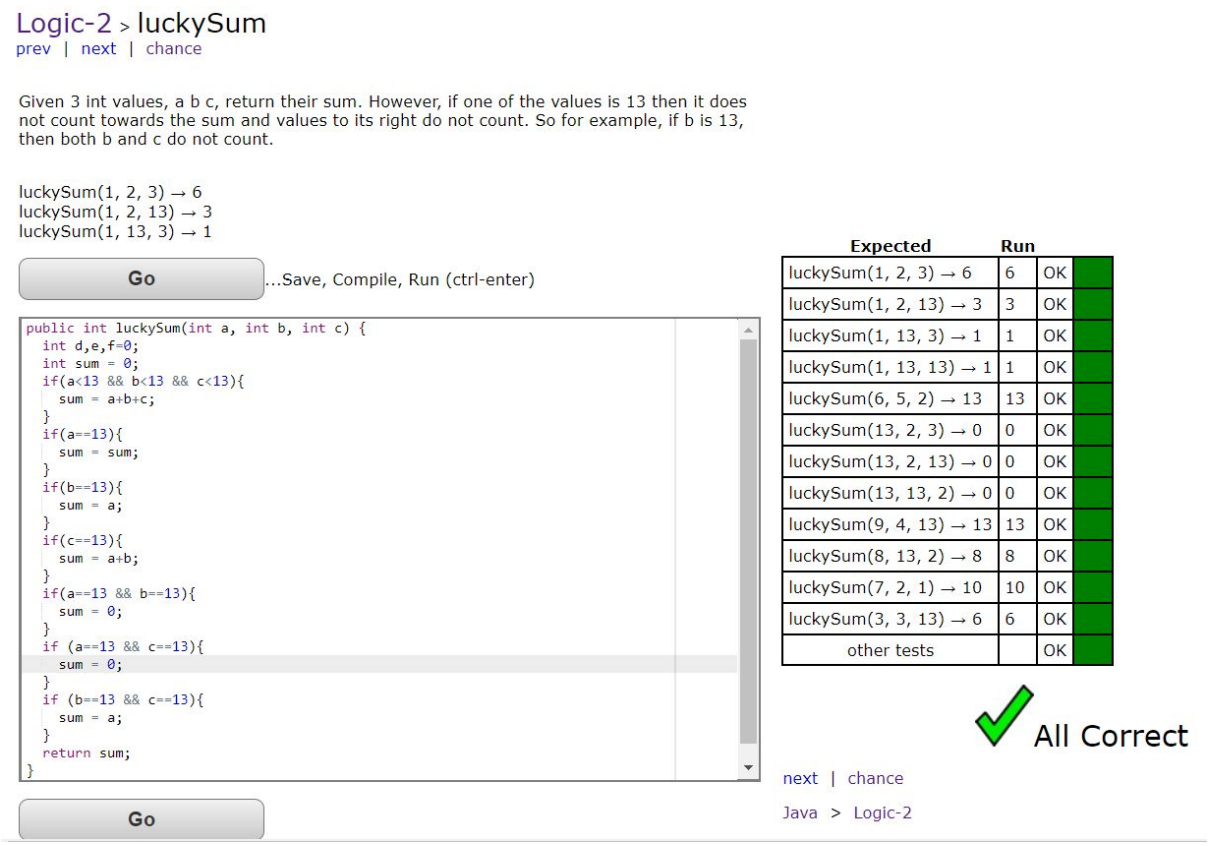
All Correct

next | chance

Java > Logic-1

2. Choose two exercises from the **medium logic** puzzles. Record the resulting code in your report.

The first exercise i chose was luckySum.



Logic-2 > luckySum
prev | next | chance

Given 3 int values, a b c, return their sum. However, if one of the values is 13 then it does not count towards the sum and values to its right do not count. So for example, if b is 13, then both b and c do not count.

luckySum(1, 2, 3) → 6
luckySum(1, 2, 13) → 3
luckySum(1, 13, 3) → 1

Go ...Save, Compile, Run (ctrl-enter)

```
public int luckySum(int a, int b, int c) {
  int d,e,f=0;
  int sum = 0;
  if(a<13 && b<13 && c<13){
    sum = a+b+c;
  }
  if(a==13){
    sum = sum;
  }
  if(b==13){
    sum = a;
  }
  if(c==13){
    sum = a+b;
  }
  if(a==13 && b==13){
    sum = 0;
  }
  if (a==13 && c==13){
    sum = 0;
  }
  if (b==13 && c==13){
    sum = a;
  }
  return sum;
}
```

Go

Expected	Run		
luckySum(1, 2, 3) → 6	6	OK	
luckySum(1, 2, 13) → 3	3	OK	
luckySum(1, 13, 3) → 1	1	OK	
luckySum(1, 13, 13) → 1	1	OK	
luckySum(6, 5, 2) → 13	13	OK	
luckySum(13, 2, 3) → 0	0	OK	
luckySum(13, 2, 13) → 0	0	OK	
luckySum(13, 13, 2) → 0	0	OK	
luckySum(9, 4, 13) → 13	13	OK	
luckySum(8, 13, 2) → 8	8	OK	
luckySum(7, 2, 1) → 10	10	OK	
luckySum(3, 3, 13) → 6	6	OK	
other tests		OK	

All Correct

next | chance

Java > Logic-2

And the second one i chose was roundSum.

Logic-2 > roundSum

prev | next | chance

For this problem, we'll round an int value up to the next multiple of 10 if its rightmost digit is 5 or more, so 15 rounds up to 20. Alternately, round down to the previous multiple of 10 if its rightmost digit is less than 5, so 12 rounds down to 10. Given 3 ints, a b c, return the sum of their rounded values. To avoid code repetition, write a separate helper "public int round10(int num) {" and call it 3 times. Write the helper entirely below and at the same indent level as roundSum().

roundSum(16, 17, 18) → 60
roundSum(12, 13, 14) → 30
roundSum(6, 4, 4) → 10

Go ...Save, Compile, Run (ctrl-enter)

```
public int roundSum(int a, int b, int c) {
  int result = round10(a)+round10(b)+round10(c);
  return result;
}
public int round10(int num){
  // leftover returns the 2nd digit
  // if i want the 6 in the number 16, 16%10 = 1 with rest of 6
  int leftover = num%10;
  if (leftover > 4){
    return  (((num/10)+1)*10);
  }
  if(leftover<5){
    return (num/10)*10;
  }
  return num;
}
```

Go

Editor font size %: 100

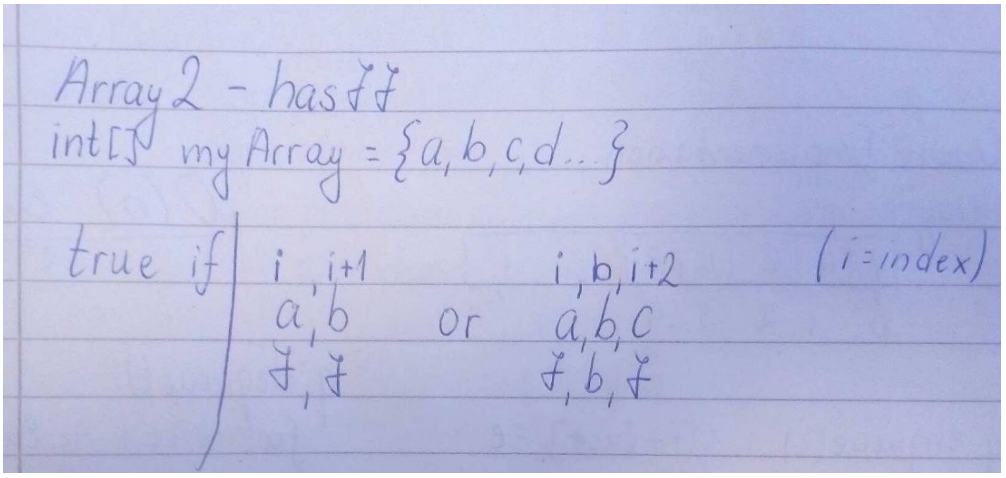
Shorter output

	Expected	Run	
roundSum(16, 17, 18) → 60	60	OK	
roundSum(12, 13, 14) → 30	30	OK	
roundSum(6, 4, 4) → 10	10	OK	
roundSum(4, 6, 5) → 20	20	OK	
roundSum(4, 4, 6) → 10	10	OK	
roundSum(9, 4, 4) → 10	10	OK	
roundSum(0, 0, 1) → 0	0	OK	
roundSum(0, 9, 0) → 10	10	OK	
roundSum(10, 10, 19) → 40	40	OK	
roundSum(20, 30, 40) → 90	90	OK	
roundSum(45, 21, 30) → 100	100	OK	
roundSum(23, 11, 26) → 60	60	OK	
roundSum(23, 24, 25) → 70	70	OK	
roundSum(25, 24, 25) → 80	80	OK	
roundSum(23, 24, 29) → 70	70	OK	
roundSum(11, 24, 36) → 70	70	OK	
roundSum(24, 36, 32) → 90	90	OK	
roundSum(14, 12, 26) → 50	50	OK	
roundSum(12, 10, 24) → 40	40	OK	
other tests		OK	

All Correct

3. Choose two exercises from the **medium array** puzzles. Record the resulting code in your report.

The first exercise that i chose was has77. I tried writing different lines of code for a while, but without success. Then I decided to take pen and paper and write down my thoughts. It suddenly all made sense to me when i also wrote down the indexes.



After writing that down, writing the code was easier.

Array-2 > has77

prev | next | chance

Given an array of ints, return true if the array contains two 7's next to each other, or there are two 7's separated by one element, such as with {7, 1, 7}.

has77([1, 7, 7]) → true
has77([1, 7, 1, 7]) → true
has77([1, 7, 1, 1, 7]) → false

Go ...Save, Compile, Run (ctrl-enter)

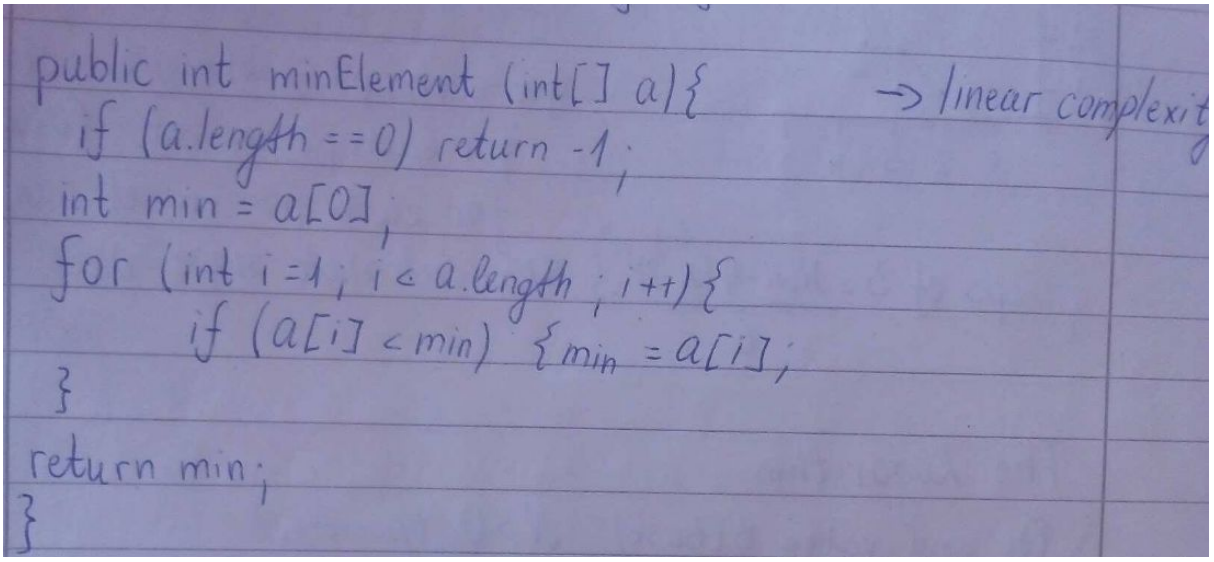
```
public boolean has77(int[] nums) {
  for (int i = 0; i<nums.length ; i++){
    if(i<nums.length - 1){
    if(nums[i] == 7 && (nums[i+1])==7){
      return true;
    }
    }
    if(i<nums.length -2){
    if(nums[i] == 7 && (nums[i+2])==7){
      return true;
    }
    }
  }
  return false;
}
```

Go

Expected	Run		
has77([1, 7, 7]) → true	true	OK	
has77([1, 7, 1, 7]) → true	true	OK	
has77([1, 7, 1, 1, 7]) → false	false	OK	
has77([7, 7, 1, 1, 7]) → true	true	OK	
has77([2, 7, 2, 2, 7, 2]) → false	false	OK	
has77([2, 7, 2, 2, 7, 7]) → true	true	OK	
has77([7, 2, 7, 2, 2, 7]) → true	true	OK	
has77([7, 2, 6, 2, 2, 7]) → false	false	OK	
has77([7, 7, 7]) → true	true	OK	
has77([7, 1, 7]) → true	true	OK	
has77([7, 1, 1]) → false	false	OK	
has77([1, 2]) → false	false	OK	
has77([1, 7]) → false	false	OK	
has77([7]) → false	false	OK	
other tests		OK	

All Correct

The next exercise i chose was bigDiff. We already had a similar exercise in class, where we had to find the minElement.



Now for the bigDiff exercise I have to keep track of both min and max.

Java | Python

Array-2 > bigDiff

prev | next | chance

Given an array length 1 or more of ints, return the difference between the largest and smallest values in the array. Note: the built-in Math.min(v1, v2) and Math.max(v1, v2) methods return the smaller or larger of two values.

bigDiff([10, 3, 5, 6]) → 7
bigDiff([7, 2, 10, 9]) → 8
bigDiff([2, 10, 7, 2]) → 8

Go ...Save, Compile, Run (ctrl-enter)

```
public int bigDiff(int[] nums) {
  //keeping track of both max and min
  int maxInt = nums[0];
  int minInt = nums[0];
  for (int i=0; i<nums.length ; i++){
    if(nums[i]>maxInt) maxInt = nums[i];
    if (nums[i]<minInt ) minInt = nums[i];
  }
  return maxInt-minInt;
}
```

Expected	Run		
bigDiff([10, 3, 5, 6]) → 7	7	OK	
bigDiff([7, 2, 10, 9]) → 8	8	OK	
bigDiff([2, 10, 7, 2]) → 8	8	OK	
bigDiff([2, 10]) → 8	8	OK	
bigDiff([10, 2]) → 8	8	OK	
bigDiff([10, 0]) → 10	10	OK	
bigDiff([2, 3]) → 1	1	OK	
bigDiff([2, 2]) → 0	0	OK	
bigDiff([2]) → 0	0	OK	
bigDiff([5, 1, 6, 1, 9, 9]) → 8	8	OK	
bigDiff([7, 6, 8, 5]) → 3	3	OK	
bigDiff([7, 7, 6, 8, 5, 5, 6]) → 3	3	OK	
other tests		OK	

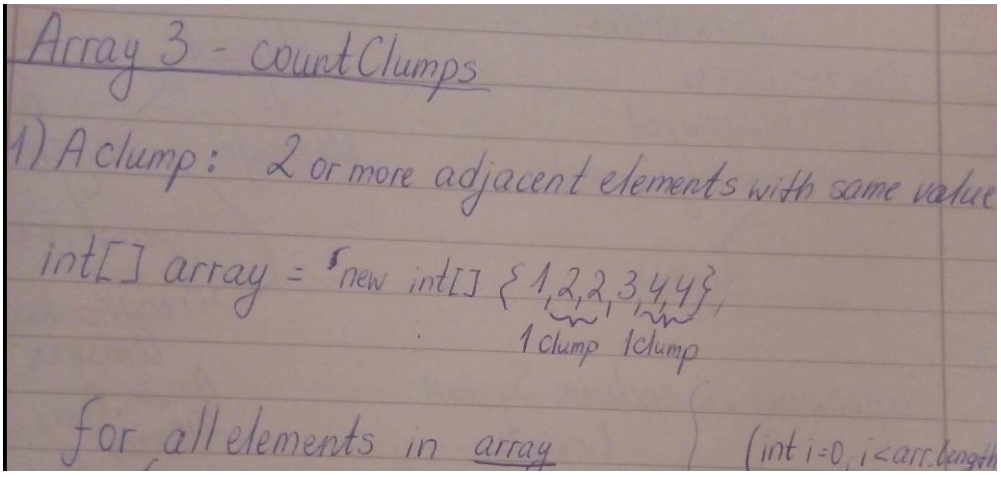
All Correct

next | chance

Java > Array-2

4. Chose one exercise from the **harder array** puzzles. Record the resulting code in your report.

The exercise that i chose was countClumps. I had to sit down with pen and paper again and write down my ideas. This one looked a bit similar to the has77 puzzle to me, but now instead of having specific numbers to look for, i had to look at the indexes themselves. Seeing the different details while writing down my thoughts made it easier to understand what the next step should be.



for all elements in array

if (element i == element i+1)

increase clumps

(int i=0; i<arr.length; ...

(array[i] == array[i+1])

clumps++;

but now.. we will check the "clumped" value as well.. e.g {1,2,2,3}

i=0 i=1 i=2 i=3

{2,2,2,2} = 1 clump!!!

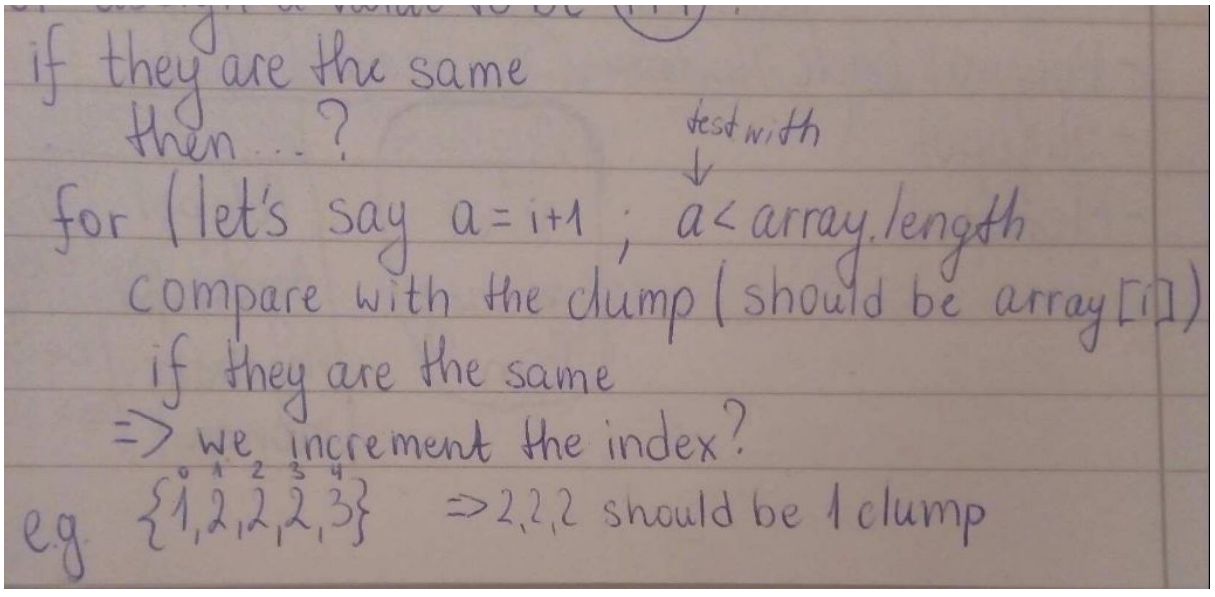
another for loop?

compare something to (i+1)

or assign a value to be (i+1)?

if they are the same

then ..?



if they are the same

then ..?

test with

for (let's say a = i+1 ; a < array.length

compare with the clump (should be array[i])

if they are the same

=> we increment the index?

e.g {1,2,2,2,3} => 2,2,2 should be 1 clump

Array-3 > countClumps

prev | next | chance

Say that a "clump" in an array is a series of 2 or more adjacent elements of the same value. Return the number of clumps in the given array.

countClumps([1, 2, 2, 3, 4, 4]) → 2
countClumps([1, 1, 2, 1, 1]) → 2
countClumps([1, 1, 1, 1, 1]) → 1

Go ...Save, Compile, Run (ctrl-enter)

```
public int countClumps(int[] nums) {
      int clumps = 0;
      if(nums.length == 0) {
        clumps = 0;
      }
      //e.g {1,1,1,2}, comparing 1 to 1 first
      for (int i = 0; i<nums.length-1; i++)
        if (nums[i] == nums[i+1]) {
         clumps++;
         // but in the {1,1,1,2} array 1,1,1 doesn't make 2 clumps
         // let's say a is the second "1"
           for (int a=i+1; a<nums.length; a++)
         // if it was the same as the previous value, and in this case it is
             if (nums[a] == nums[i])
               i++;
        }
      return clumps;
  }
```

Expected	Run		
countClumps([1, 2, 2, 3, 4, 4]) → 2	2	OK	
countClumps([1, 1, 2, 1, 1]) → 2	2	OK	
countClumps([1, 1, 1, 1, 1]) → 1	1	OK	
countClumps([1, 2, 3]) → 0	0	OK	
countClumps([2, 2, 1, 1, 1, 2, 1, 1, 2, 2]) → 4	3	X	
countClumps([0, 2, 2, 1, 1, 1, 2, 1, 1, 2, 2]) → 4	3	X	
countClumps([0, 0, 2, 2, 1, 1, 1, 2, 1, 1, 2, 2]) → 5	4	X	
countClumps([0, 0, 0, 2, 2, 1, 1, 1, 2, 1, 1, 2, 2]) → 5	4	X	
countClumps([]) → 0	0	OK	
other tests		OK	

Correct for more than half the tests

Your progress graph for this problem

I still had some issues with getting the program to do exactly as required.

Since I didn't find anyone else doing the clump puzzle, I talked to a friend(Dimitar) who is also studying programming. He said I was on the right path, but suggested that I should use a while loop instead of a second for loop.

Array-3 > countClumps

prev | next | chance

Say that a "clump" in an array is a series of 2 or more adjacent elements of the same value. Return the number of clumps in the given array.

countClumps([1, 2, 2, 3, 4, 4]) → 2
countClumps([1, 1, 2, 1, 1]) → 2
countClumps([1, 1, 1, 1, 1]) → 1

Go ...Save, Compile, Run (ctrl-enter)

```
public int countClumps(int[] nums) {
      int clumps = 0;
      if(nums.length == 0) {
        clumps = 0;
      }
      //e.g {1,1,1,2}, comparing 1 to 1 first
      for (int i = 0; i<nums.length-1; i++)
        if (nums[i] == nums[i+1]) {
         clumps++;
         // but in the {1,1,1,2} array ,  1,1,1 doesn't make 2 clumps
         // so if the thing we checked was the same as
         // the next value, and in this case it is ( comparing 1 with 1)
         // we should increment the index to go to the next value in the array
         while(i+1<nums.length && nums[i+1]==nums[i]){
              i++;
         }

  }
  return clumps;
}
```

Go

Expected	Run		
countClumps([1, 2, 2, 3, 4, 4]) → 2	2	OK	
countClumps([1, 1, 2, 1, 1]) → 2	2	OK	
countClumps([1, 1, 1, 1, 1]) → 1	1	OK	
countClumps([1, 2, 3]) → 0	0	OK	
countClumps([2, 2, 1, 1, 1, 2, 1, 1, 2, 2]) → 4	4	OK	
countClumps([0, 2, 2, 1, 1, 1, 2, 1, 1, 2, 2]) → 4	4	OK	
countClumps([0, 0, 2, 2, 1, 1, 1, 2, 1, 1, 2, 2]) → 5	5	OK	
countClumps([0, 0, 0, 2, 2, 1, 1, 1, 2, 1, 1, 2, 2]) → 5	5	OK	
countClumps([]) → 0	0	OK	
other tests		OK	

All Correct

next | chance

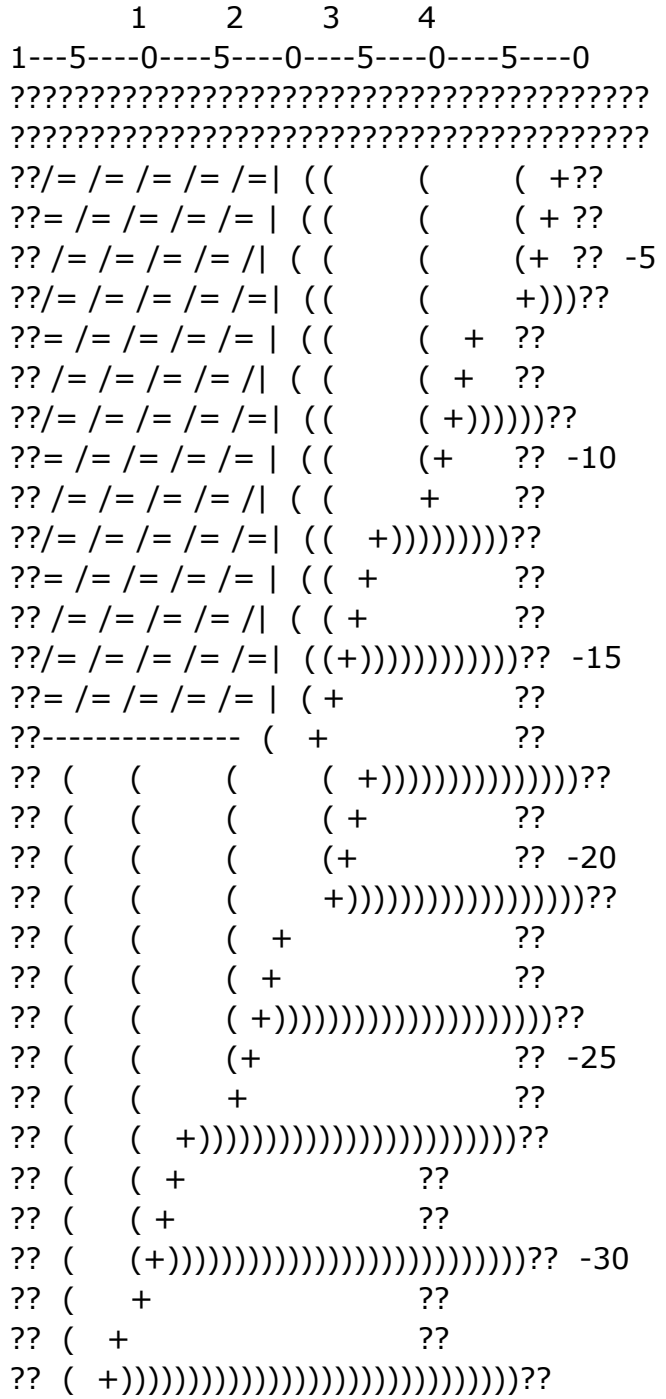
Java > Array-3

ptsvyatkov97@gmail.com done page

Your progress graph for this problem

5. Since you were doing lots of flags in GDM, here's another flag problem.

Assume that you are a member of the programming committee for implementing a method to print the new EU flag for after Brexit. The surprising decision is given below:



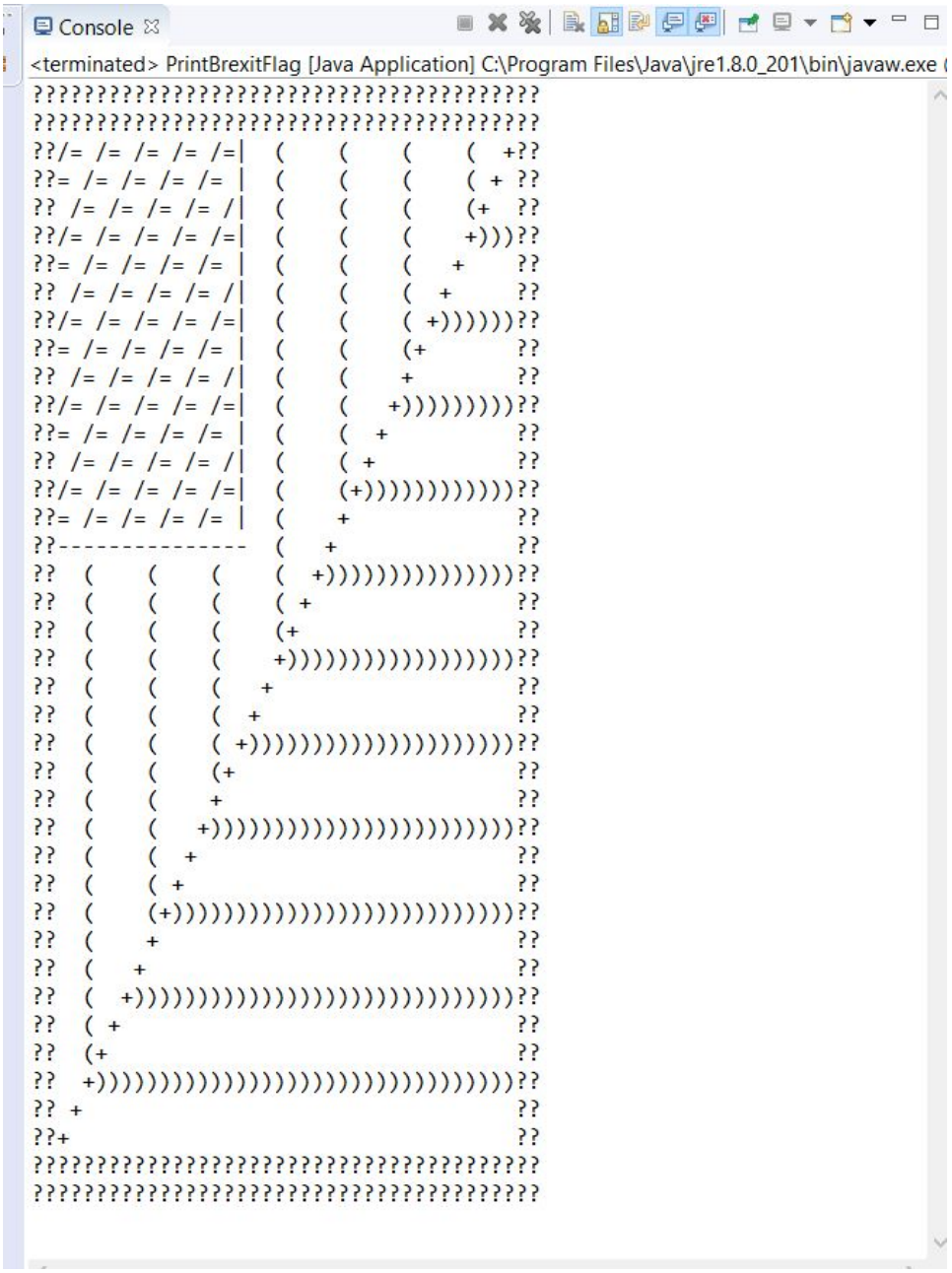
```
??  ( +                    ??
??  (+                     ??  -35
??  +))))))))))))))))))))))))))))))))))??
??  +                      ??
??+                        ??
????????????????????????????????????????
????????????????????????????????????????  -40
```

The method public char determineCharacter (int column, int row);
needs implementation, so that it can be called from the nested loop

```
String outputLine;
for (int row = 1; row <= 40; row++){
   outputLine = "";
   for (int column = 1; column <= 40; column++){
      outputLine = outputLine+determineCharacter (column, row);
   }
   System.out.println (outputLine);
}
```

Document the body of the method in your report, including a screenshot of it working.

For this exercise i collaborated together with Martin. We sat down and talked about the exercise, shared ideas and tested how the printing works. First we started with the outer border and then the border of the smaller rectangle in the upper left. I decided to write a lot of comments so I can easily keep track of changes and if something had to be checked again. Counting the rows and columns and looking for repeated patterns was the most important thing to do.



Personal Reflection:

Working on the exercises in CodingBat was very interesting, because some of them were challenging and I really liked how I started writing down everything step by step to figure out what I had to do. I'm going to check the other exercises as well, it really helped me to improve my problem solving skills. While working on the flag problem I was able to understand some very important concepts about printing flags.

Time Invested:
Simple and Medium Logic - 30 min
Medium and Hard Array - around 50-60 min
PrintFlag - around 45-60 min

Appendix

Code

Simple Logic Puzzles:

```
public boolean cigarParty(int cigars, boolean isWeekend) {
        if (isWeekend && cigars>=40){
          return true;
        }
        else if (!isWeekend && cigars>=40 && cigars<=60){
          return true;
        }
        return false;
      }

public String alarmClock(int day, boolean vacation) {
        String alarm = "7:00";
```

```
    String alarmWeekend = "10:00";
    String alarmOff = "off";
    if (!vacation){
      if(day == 1 || day == 2 || day == 3 || day == 4 || day == 5 ){
        return alarm;
      }
      else{
        return alarmWeekend;
      }
    }
    else{
      if(day == 1 || day == 2 || day == 3 || day == 4 || day == 5 ){
        return alarmWeekend;
      }
      else {
        return alarmOff;
      }
    }
    }

public boolean twoAsOne(int a, int b, int c) {
    if(a+b == c || a+c == b || b+c == a){
      return true;
    }
```

```
    return false;
  }
```

Medium Logic Puzzles:

```
public int luckySum(int a, int b, int c) {
    int sum = 0;
    if(a<13 && b<13 && c<13){
      sum = a+b+c;
    }
    if(a==13){
      sum = 0;
    }
    if(b==13){
      sum = a;
    }
    if(c==13){
      sum = a+b;
    }
    if(a==13 && b==13){
      sum = 0;
    }
    if (a==13 && c==13){
      sum = 0;
    }
```

```
        if (b==13 && c==13){
          sum = a;
        }
        return sum;
      }
}
public int roundSum(int a, int b, int c) {
        int result = round10(a)+round10(b)+round10(c);
        return result;
      }
 public int round10(int num){
        // leftover returns the 2nd digit
        // if i want the 6 in the number 16, 16%10 = 1 with rest of 6
        int leftover = num%10;
        if (leftover > 4){
          return  (((num/10)+1)*10);
        }
              if(leftover<5){
                return (num/10)*10;
              }
        return num;
      }
```

Medium Array Puzzles:

```
public boolean has77(int[] nums) {
  for (int i = 0; i<nums.length ; i++){
    if(i<nums.length - 1){
    if(nums[i] == 7 && (nums[i+1])==7){
      return true;
    }
    }
    if(i<nums.length -2){
    if(nums[i] == 7 && (nums[i+2])==7){
      return true;
    }
    }
  }
  return false;
}
public int bigDiff(int[] nums) {
  //keeping track of both max and min
  int maxInt = nums[0];
  int minInt = nums[0];
  for (int i=0; i<nums.length ; i++){
    if(nums[i]>maxInt) maxInt = nums[i];
    if (nums[i]<minInt ) minInt = nums[i];
  }
```

```
  return maxInt-minInt;
}
```

Harder Array Puzzle:

```
public int countClumps(int[] nums) {
            int clumps = 0;
            if(nums.length == 0) {
              clumps = 0;
            }
            //e.g {1,1,1,2}, comparing 1 to 1 first
            for (int i = 0; i<nums.length-1; i++)
              if (nums[i] == nums[i+1]) {
               clumps++;
               // but in the {1,1,1,2} array ,  1,1,1 doesn't make 2 clumps
               // so if the thing we checked was the same as
               // the next value, and in this case it is ( comparing 1 with 1)
                 // we should increment the index to go to the next value in the
array
               while(i+1<nums.length && nums[i+1]==nums[i]){
                   i++;
               }

        }
        return clumps;
}
```

Print Flag:

```
public class PrintBrexitFlag {


	public static void main(String[] args) {
		PrintBrexitFlag flag = new PrintBrexitFlag();
		// System.out.println("0  1  2  3  4  5  6  7  8");
		String outputLine;
		for (int row = 1; row <= 40; row++){
		   outputLine = "";
		   for (int column = 1; column <= 40; column++){
				outputLine  =  outputLine+flag.determineCharacter
(column, row);
		   }
		   System.out.println (outputLine);
		}
	}


	public char determineCharacter (int column, int row) {
		//Border
		if (column<3) {
			return '?';
```

```
		}


		if (column>38) {
			return '?';
		}
		if (row<3) {
			return '?';
		}


		if (row>38) {
			return '?';
		}


		//upper left rectangle right border
		if(column==17 && row<17 && row>2) {
			return '|';
		}
		if (row==17 && column>2 && column<18) {
			return '-';
		}
		// row 3,6,9,12,15 and column 3,5,7,9,11,13,15
		// but it should be both 3,6,9,12,15
		// change from column%2==1 to column%3==0 to get 3,6,9,12,15
with a "/"
```

```
            // then a bracket ) appears for some reason
            // but the / is correct
            if (row>2 && row<17 && column>2 && column<18 && row%3==0
&& column%3==0) {
                  return '/';
            }

            //row 4,7,10,13,16 and column 4,6,8,10,12,14,16
            // row is correct but column looks 1 off (has to be 5,8,11,14)
            // trying to change from column%2==0 to column%3==2
            // looks like it worked but some more bracket ( appeared
            if (row>2 && row<17 && column>2 && column<18 && row%3==1
&& column%3==2) {
                  return '/';
            }

            //row 5,8,11,14 and column 4,6,8,10,12,14,16
            //row is correct but column should be 4,7,10,13,16
            // trying to change from column%2==0 to column%3==1
            // looks like it worked
            if (row>2 && row<17 && column>2 && column<18 && row%3==2
&& column%3==1) {
                  return '/';
            }
```

```
		//row 3,6,9,12,15 and column 4,6,8,10,12,14,16

		// row is correct but column should be 4,7,10,13,16

		// tried to change column%2==0 to column%3==1

		// looks like it worked

		if (row>2 && row<17 && column>2 && column<18 && row%3==0 && column%3==1) {

			return '=';

		}

		//row 4,7,10,13,16 and column 3,5,7,9,11,13,15

		//row is correct but column should be 3,6,9,12,15

		//trying to change column%2==1 to column%3==0

		//looks like it worked

		if (row>2 && row<17 && column>2 && column<18 && row%3==1 && column%3==0) {

			return '=';

		}


		//row 5,8,11,14 and column 3,5,7,9,11,13,15

		// row looks correct but column should be 5,8,11,14

		// trying to change from column%2==1 to column%3==2

		// looks like it worked

		if (row>2 && row<17 && column>2 && column<18 && row%3==2 && column%3==2) {

			return '=';

		}
```

```
		// now need to fix the weird bracket ( appearing, go to line 114
				// or maybe add if statement with return "space"?


		// row 3,6,9,12,15 column 5,8,11,14 has "space"
		if(row>2 && row<17 && column>2 && column<18 && row%3==0 && column%3==2) {
			return ' ';
		}


		// row 4,7,10,13,16 column 4,7,10,13,16 has "space"
		if(row>2 && row<17 && column>2 && column<18 && row%3==1 && column%3==1) {
			return ' ';
		}


		// row 5,8,11,14 column 3,6,9,12,15 has "space"
		if(row>2 && row<17 && column>2 && column<18 && row%3==2 && column%3==0) {
			return ' ';
		}



		//diagonal
		if(row==(column*(-1)+41)) {
```

```
                return '+';
            }



            if(row<(column*(-1)+41)) {
                if (column%5==0) {
                    return '(';
                }else {
                    return ' ';
                }

            } else {
                if (row%3==0) {
                    return ')';
                } else {
                    return ' ';
                }
            }
        }
}
```