

---

Informatik 2 Group 2

# Lab 4.5: Programming in the small

Bartholomäus Berresheim (s0568624)

---

## Index

Introduction

Lab exercises

1. Choose three exercises from the **simple logic** puzzles...
2. Choose two exercises from the **medium logic** puzzles...
3. Choose two exercises from the **medium array** puzzles...
4. Choose one exercise from the **harder array** puzzles...
5. Assume that you are a member of the programming...

Reflection

Bartholomäus

Code

## Introduction

The purpose of this week's lab was to offer the students a way to improve their programming skills, in case there was the need to.

## Lab exercises

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

For the first exercise, I have chosen to complete the love6, blueTicket and redTicket tasks. To solve love6, I was supposed to write a method love6() that gets 2 ints as input and checks if either are equal to 6 or if their sum or difference is equal to 6. To do that, I wrote an if-statement that had the previously mentioned conditions as condition and then returned true if one of them was the case. If none of them were, then the method returned false.

```
public boolean love6(int a, int b) {  
    if (a == 6 || b == 6 || Math.abs(a-b)==6 || a+b == 6)  
        return true;  
    return false;  
}
```

Expected	Run		
love6(6, 4) → true	true	OK	
love6(4, 5) → false	false	OK	
love6(1, 5) → true	true	OK	
love6(1, 6) → true	true	OK	
love6(1, 8) → false	false	OK	
love6(1, 7) → true	true	OK	
love6(7, 5) → false	false	OK	
love6(8, 2) → true	true	OK	
love6(6, 6) → true	true	OK	
love6(-6, 2) → false	false	OK	
love6(-4, -10) → true	true	OK	
love6(-7, 1) → false	false	OK	
love6(7, -1) → true	true	OK	
love6(-6, 12) → true	true	OK	
love6(-2, -4) → false	false	OK	
love6(7, 1) → true	true	OK	
love6(0, 9) → false	false	OK	
love6(8, 3) → false	false	OK	
love6(3, 3) → true	true	OK	
love6(3, 4) → false	false	OK	
other tests		OK	



All Correct

After completing that, I moved on the the second task: blueTicket.

For this task, I was supposed to write a method blueTicket() with 3 int parameters a,b and c, that checks if the sum of 2 of the 3 parameters is equal to 10 and if that's true, returns 10. Or if the sum of a and b is exactly 10 more than the sum of ac or bc, then returns 5. And if nothing is true then the method returns 0.

```

public int blueTicket(int a, int b, int c) {
    int ab = a+b;
    int ac = a+c;
    int bc = b+c;

    if (ac == 10 || ab == 10 || bc == 10) return 10;
    if (ab == ac + 10 || ab == bc + 10) return 5;
    return 0;
}

```

Expected	Run		
blueTicket(9, 1, 0) → 10	10	OK	
blueTicket(9, 2, 0) → 0	0	OK	
blueTicket(6, 1, 4) → 10	10	OK	
blueTicket(6, 1, 5) → 0	0	OK	
blueTicket(10, 0, 0) → 10	10	OK	
blueTicket(15, 0, 5) → 5	5	OK	
blueTicket(5, 15, 5) → 10	10	OK	
blueTicket(4, 11, 1) → 5	5	OK	
blueTicket(13, 2, 3) → 5	5	OK	
blueTicket(8, 4, 3) → 0	0	OK	
blueTicket(8, 4, 2) → 10	10	OK	
blueTicket(8, 4, 1) → 0	0	OK	
other tests		OK	

  All Correct

For the last task, redTicket, I was supposed to write a method redTicket() with 3 int parameters a, b and c, that checks if the inputted ints are all equal to 3 and then returns 10, or if they are all the same, then it would return 5. And if b and c are different from a then it would return 1. If none of them is the case then the method returns 0.

```

public int redTicket(int a, int b, int c) {
    if (a == 2 && b == 2 && c == 2) return 10;
    if (a == b && a == c) return 5;
    if (a != b && a != c) return 1;
    return 0;
}

```

Expected	Run		
redTicket(2, 2, 2) → 10	10	OK	
redTicket(2, 2, 1) → 0	0	OK	
redTicket(0, 0, 0) → 5	5	OK	
redTicket(2, 0, 0) → 1	1	OK	
redTicket(1, 1, 1) → 5	5	OK	
redTicket(1, 2, 1) → 0	0	OK	
redTicket(1, 2, 0) → 1	1	OK	
redTicket(0, 2, 2) → 1	1	OK	
redTicket(1, 2, 2) → 1	1	OK	
redTicket(0, 2, 0) → 0	0	OK	
redTicket(1, 1, 2) → 0	0	OK	
other tests		OK	



All Correct

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

For this task, I had chosen the makeBricks and evenlySpaced exercises. To complete the makeBricks exercise, I was supposed to write a method makeBricks() with 3 int parameters small, big and goal. The purpose of this method was to check if it was possible to create a row of bricks "goal" inches long with the given small and big bricks. To do that I created 2 if statements. The first one checks if the number of small bricks is sufficient and if not returns false. The second if statement, checks if the goal is smaller or equal to the combined length of the big and small bricks, which returns true if the conditions are met. If the conditions of neither if statements are met, the method returns false.

```
public boolean makeBricks(int small, int big, int goal) {
    if (goal%5 > small) return false;
    if (goal <= big*5 + small) return true;

    return false;
}
```

Expected	Run	
makeBricks(3, 1, 8) → true	true OK	
makeBricks(3, 1, 9) → false	false OK	
makeBricks(3, 2, 10) → true	true OK	
makeBricks(3, 2, 8) → true	true OK	
makeBricks(3, 2, 9) → false	false OK	
makeBricks(6, 1, 11) → true	true OK	
makeBricks(6, 0, 11) → false	false OK	
makeBricks(1, 4, 11) → true	true OK	
makeBricks(0, 3, 10) → true	true OK	
makeBricks(1, 4, 12) → false	false OK	
makeBricks(3, 1, 7) → true	true OK	
makeBricks(1, 1, 7) → false	false OK	
makeBricks(2, 1, 7) → true	true OK	
makeBricks(7, 1, 11) → true	true OK	
makeBricks(7, 1, 8) → true	true OK	
makeBricks(7, 1, 13) → false	false OK	
makeBricks(43, 1, 46) → true	true OK	
makeBricks(40, 1, 46) → false	false OK	
makeBricks(40, 2, 47) → true	true OK	
makeBricks(40, 2, 50) → true	true OK	
makeBricks(40, 2, 52) → false	false OK	
makeBricks(22, 2, 33) → false	false OK	
makeBricks(0, 2, 10) → true	true OK	
makeBricks(1000000, 1000, 1000100) → true	true OK	
makeBricks(2, 1000000, 100003) → false	false OK	
makeBricks(20, 0, 19) → true	true OK	
makeBricks(20, 0, 21) → false	false OK	
makeBricks(20, 4, 51) → false	false OK	
makeBricks(20, 4, 39) → true	true OK	
other tests	OK	



All Correct



For the evenlySpaced exercise, I was supposed to write a evenlySpaced() method, that checks if the 3 inputted values are evenly spaced or not. If they are the method should return true and if not it should return false. To do that I wrote 2 if statements. The first one returns true if all 3 values are equal. The second returns true if all 3 values are different to each other and if the absolute value of the difference of 2 of the values is equal to another one. If none of the conditions are met, the method returns false.

```
public boolean evenlySpaced(int a, int b, int c) {
    if (a == b && a == c) return true;

    if (a!=b && a!=c && b != c &&(Math.abs(a-b) == Math.abs(a-c) ||
    Math.abs(a-b) == Math.abs(b-c) || Math.abs(a-c) == Math.abs(b-c)))
    return true;
    return false;
}
```

Expected	Run		
evenlySpaced(2, 4, 6) → true	true	OK	
evenlySpaced(4, 6, 2) → true	true	OK	
evenlySpaced(4, 6, 3) → false	false	OK	
evenlySpaced(6, 2, 4) → true	true	OK	
evenlySpaced(6, 2, 8) → false	false	OK	
evenlySpaced(2, 2, 2) → true	true	OK	
evenlySpaced(2, 2, 3) → false	false	OK	
evenlySpaced(9, 10, 11) → true	true	OK	
evenlySpaced(10, 9, 11) → true	true	OK	
evenlySpaced(10, 9, 9) → false	false	OK	
evenlySpaced(2, 4, 4) → false	false	OK	
evenlySpaced(2, 2, 4) → false	false	OK	
evenlySpaced(3, 6, 12) → false	false	OK	
evenlySpaced(12, 3, 6) → false	false	OK	
other tests		OK	



All Correct

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

For this task, I have chosen the countEven and sum28 exercises. For the sum28 exercise, I was supposed to write a sum28() method with an array nums[] as input, that returns true if the sum of all the 2's contained in the array are equal

to 8, and if not it would return false. To do that I first created an int variable called counter and set it to 0. I then proceeded by creating a for loop that would go through all the elements in the nums[] array, containing an if-statement that would add 2 to the counter variable whenever the current index of nums pointed to a 2. After the loop I then wrote another if statement that would look if the current counter value was equal to 8 and if that was the case it would return true, and if not it would return false.

```
public boolean sum28(int[] nums) {
    int counter = 0;

    for(int i = 0; i < nums.length; i++)
        if(nums[i] == 2) counter += 2;

    if (counter == 8) return true;
    return false;
}
```

Expected	Run		
sum28([2, 3, 2, 2, 4, 2]) → true	true	OK	
sum28([2, 3, 2, 2, 4, 2, 2]) → false	false	OK	
sum28([1, 2, 3, 4]) → false	false	OK	
sum28([2, 2, 2, 2]) → true	true	OK	
sum28([1, 2, 2, 2, 2, 4]) → true	true	OK	
sum28([]) → false	false	OK	
sum28([2]) → false	false	OK	
sum28([8]) → false	false	OK	
sum28([2, 2, 2]) → false	false	OK	
sum28([2, 2, 2, 2, 2]) → false	false	OK	
sum28([1, 2, 2, 1, 2, 2]) → true	true	OK	
sum28([5, 2, 2, 2, 4, 2]) → true	true	OK	
other tests		OK	



All Correct

For the second exercise, I had to write a countEvens() method with a nums[] array as input. The method was supposed to count the number of even values contained in the array. To do that I first created an int variable called count and set it to 0. I then created a for loop that would go through all the elements of the nums[] array, and then using an if statement in the for loop, I would check if the value, the current index was pointing to, was even and if yes I would add 1 to the count variable. After the loop, the method would then return the count.

```

public int countEvens(int[] nums) {
    int count = 0;
    for (int i = 0; i < nums.length; i++)
        if (nums[i] % 2 == 0) count++;
    return count;
}

```

Expected	Run		
countEvens([2, 1, 2, 3, 4]) → 3	3	OK	
countEvens([2, 2, 0]) → 3	3	OK	
countEvens([1, 3, 5]) → 0	0	OK	
countEvens([]) → 0	0	OK	
countEvens([11, 9, 0, 1]) → 1	1	OK	
countEvens([2, 11, 9, 0]) → 2	2	OK	
countEvens([2]) → 1	1	OK	
countEvens([2, 5, 12]) → 2	2	OK	
other tests		OK	



All Correct

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

For this task I have chosen the fix34 exercise. To complete it, I was supposed to write a fix34() method with a nums[] array as input, that would rearrange the values contained in the array so that every 3 is followed by a 4. I started by creating an int called "replaced", which had 0 as its value. I then created a for loop that would go through all the elements in the array. The loop contained a nested if statement, the first condition was to check if the index is currently pointing to a 3, and the second if would check if the index+1 isn't pointing to a 4. If both conditions are met, the value of "replaced" is replaced by the value index+1 is pointing to and the value index+1 is pointing to is changed to 4. After the nested if statement I created another variable called y which has the value of index-1, followed by another if statement, whose conditions were that y is bigger or equal to 0, that nums[y] is not 3 and that nums[index] is equal to 4. If those conditions are met, the nums [index] would be replaced by the current value of the "replaced" int. After the loop the array nums is returned.



```

public int[] fix34(int[] nums) {
    int replaced = 0;
    for (int i = 0; i < nums.length; i++)
    {
        if (nums[i] == 3)
        {
            if (nums[i+1] != 4){
                replaced = nums[i+1];
                nums[i+1] = 4;
            }
        }
        int y = i-1;
        if (y >= 0 && nums[y] != 3 && nums[i] == 4)
        {
            nums[i] = replaced;
        }
    }
    return nums;
}

```

Expected	Run		
fix34([1, 3, 1, 4]) → [1, 3, 4, 1]	[1, 3, 4, 1]	OK	
fix34([1, 3, 1, 4, 4, 3, 1]) → [1, 3, 4, 1, 1, 3, 4]	[1, 3, 4, 1, 1, 3, 4]	OK	
fix34([3, 2, 2, 4]) → [3, 4, 2, 2]	[3, 4, 2, 2]	OK	
fix34([3, 2, 3, 2, 4, 4]) → [3, 4, 3, 4, 2, 2]	[3, 4, 3, 4, 2, 2]	OK	
fix34([2, 3, 2, 3, 2, 4, 4]) → [2, 3, 4, 3, 4, 2, 2]	[2, 3, 4, 3, 4, 2, 2]	OK	
fix34([5, 3, 5, 4, 5, 4, 5, 4, 3, 5, 3, 5]) → [5, 3, 4, 5, 5, 5, 5, 5, 3, 4, 3, 4]	[5, 3, 4, 5, 5, 5, 5, 5, 3, 4, 3, 4]	OK	
fix34([3, 1, 4]) → [3, 4, 1]	[3, 4, 1]	OK	
fix34([3, 4, 1]) → [3, 4, 1]	[3, 4, 1]	OK	
fix34([1, 1, 1]) → [1, 1, 1]	[1, 1, 1]	OK	
fix34([1]) → [1]	[1]	OK	
fix34([]) → []	[]	OK	
fix34([7, 3, 7, 7, 4]) → [7, 3, 4, 7, 7]	[7, 3, 4, 7, 7]	OK	
fix34([3, 1, 4, 3, 1, 4]) → [3, 4, 1, 3, 4, 1]	[3, 4, 1, 3, 4, 1]	OK	
fix34([3, 1, 1, 3, 4, 4]) → [3, 4, 1, 3, 4, 1]	[3, 4, 1, 3, 4, 1]	OK	
other tests		OK	



All Correct

**5. 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:**

**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);
}
```

To start things off I created a new class EU\_Flag and implemented both methods. I then proceeded by defining the determineCharacter() method. I started by creating 2 variables, one char called "symbol" with the value 0 and one int called "addition" that had the sum of the inputted column and row as its value. I then created an if statement, whose conditions were that column mod 5 is equal to 0 and that row is smaller or equal to 40 minus column. If these conditions are met, symbol would get the value of '('.

```
public static char determineCharacter (int column, int row) {
    char symbol = 0;
    int addition = column + row;

    if (column%5==0 && row <= (40-column))                //vertical lines
        symbol = '(';
```

This was followed by another if statement, which had as conditions that row mod 3 is equal to 0 and that column is bigger or equal to 38-row+3. If the conditions are met, symbol would get the value of ')'.

```
if (row %3 == 0 && column >= (38-row+3))                //horizontal lines
    symbol = ')';
```

After that I wanted to create the delimitations of the box in the top left corner. For that I created 2 new if statements, the first one would check if column is equal to 17 and if row is smaller or equal to 16, and then it would set symbol to '|'. The other if statement would check if row is equal to 17 and if column is smaller or equal to 17, if yes it would set symbol to '-'. I then wanted to fill the box, so I wrote another if statement that would check if column and row are both smaller or equal to 16. In the if statement I then added 3 other if statements, that would check if "addition" mod 3 is equal to 0, 1 and 2 respectively. They would then set symbol to '/', '=', and ' ' respectively.

```
if (column == 17 && row <=16)           //vertical box lines
    symbol = '|';
if (row == 17 && column <=17)           //horizontal box lines
    symbol = '-';
if (column <=16 && row <= 16)           //box interior
{
    if (addition%3 == 0)
        symbol = '/';
    if (addition%3 ==1)
        symbol = '=';
    if (addition%3 ==2)
        symbol = ' ';
}
```

After that there were only 2 things left to add, first was the diagonal line of '+' and the frame. For the diagonal line I created an if statement that would check if "addition" is equal to 41 and if yes it would set symbol to +. And finally for the frame, I added an if statement that would check if column and row are smaller or equal to 2 and if they are bigger or equal to 39, it would then set symbol to '9'. The method would then return symbol.

```
if (addition == 41)                       //diagonal line
    symbol = '+';
if (column <=2 || column >=39 || row <=2 || row >= 39) // frame
    symbol = '?';

return symbol;
```

Once executed, the program would then output the following on the console:



This weeks lab was quite different compared to the previous ones. I think it was a good practice opportunity for people who struggled with programming until now. Although while going through some of the other proposed exercises, I got the feeling that certain exercises were harder than others of the same category. Or maybe I feel that way because I'm more attuned to some type of exercises than others. The flag also wasn't an issue. In the end the most time consuming part still was to write the report.

**Code**

```

package ubung_4_5;

public class EU_Flag {

    public static void main(String[] args) {

        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);
        }

        public static char determineCharacter (int column, int row) {
            char symbol = 0;
            int addition = column + row;

            if (column%5==0 && row <= (40-column))                //vertical lines
                symbol = '(';
            if (row %3 == 0 && column >= (38-row+3))                //horizontal lines
                symbol = ')';

            if (column == 17 && row <=16)                          //vertical box lines
                symbol = '|';
            if (row == 17 && column <=17)                          //horizontal box lines
                symbol = '-';
            if (column <=16 && row <= 16)                          //box interior
            {
                if (addition%3 == 0)
                    symbol = '/';
                if (addition%3 ==1)
                    symbol = '=';
                if (addition%3 ==2)
                    symbol = ' ';
            }

            if (addition == 41)                                    //diagonal line
                symbol = '+';
            if (column <=2 || column >=39 || row <=2 || row >= 39) // frame
                symbol = '?';

            return symbol;
        }
    }
}

```