

Informatics 1: Object Oriented Programming

Tutorial 01

Week 2: Pair Programming

Volker Seeker (volker.seeker@ed.ac.uk)

Vidminas Vizgirda (s1750767@ed.ac.uk)

1 Introduction

In this tutorial, we will practice working on algorithm problems. These are the kinds of questions that showcase fundamental concepts of programming (and will be covered in much more detail in second year, if you take the algorithms & data structures course). These concepts are often applicable to many kinds of problems and will likely help you with future courses, projects and programming career.

The tutorial workshops for the Inf1B course are designed to support collaborative learning and programming in groups of two or more people. Programming “in the wild” is rarely done by individuals alone. Even if you are working as a self-employed freelancer, you will likely share sources and libraries with other developers. It is therefore an essential skill to learn how to communicate with others about code, how to work with and extend code written by others and how to share programming tasks amongst multiple people.

We will start this first workshop with a technique that is commonly used in industry called *pair programming*.

- ☞ While you are welcome to read through the tutorial sheet beforehand, you are not required to prepare the exercises at home before the tutorial. The tutorials are designed as workshops which you work through during your tutorial session. As preparation work, please focus on keeping up with labs and lectures instead.

The tutorials are made for students of different levels – some will be joining the course after years of programming, others will have never coded before. It will not be possible to cover all the tutorial questions in the average group – that’s okay! Focus on learning as much as you can from each session and ask your tutors questions whenever something is unclear.

2 Warmup

Task 1 - Getting to know each other

◀ Task

The tutors will introduce an icebreaker activity. Follow their guidance.

After the activity, find other students that have the same assigned tutor as you and join them at a table.

Task 2 - Team forming time!

◀ Task

You will already have seen many programs before, both in last semester and in the lectures leading up to this tutorial. However, there are many ways of looking at and understanding programs.

One such way is to imagine yourself in a role of a variable, keeping data as the program executes.

This can scale up to bigger programs with multiple variables too. To try this out in practice, pair up with someone at your table (or join another table if needed; and if the class has an odd number of students, you can make a group of 3).

Task 3 - Manual walkthrough

◀ Task

Once you are in pairs, take a look together at these pieces of source code and choose the role of a computer component: one person will be the processor and perform calculations, the other will be the memory and keep track of results (and if your group has 3 people, the third person can be a second memory unit to help the others double check).

Your task is to “execute” the code snippets line-by-line, from top to bottom, that is – read them one by one and try to simulate what the code does on paper (on a piece of scrap paper or you can keep track in your head if you like, although this might be more difficult). Code is usually read right-to-left (the tutors can clarify). Whenever encountering a variable on the right-hand side of an expression, the processor will need to ask the memory unit(s) what the variable value is and substitute that into the line. When encountering a variable on the left-hand side, the processor will tell the memory unit(s) what value to store in that variable. At the end, write down what the values of each variable are. The tutors will show you an example with the first snippet.

Example 1: LineByLine

```
1 public class LineByLine {
2     public static void main(String[] args) {
3         int i = 0;
4         i = i + 5;
5         i = 1 - i;
6     }
7 }
```

In line 3, *i* is first assigned the value 0, then in line 4 the value $(0+5 = 5)$ and finally in line 5 the value $(1 - 5 = -4)$, so the final answer is $i = -4$. This example showcases that the order of lines of code matters, if lines 4 and 5 were reversed, the result would be different. Can you tell what it would be?

Example 2: TwoVars

```
1 public class TwoVars {
2     public static void main(String[] args) {
3         int i = 4;
4         int j = 3;
5         i = j;
6         j = 2;
7     }
8 }
```

In lines 4 and 5, *i* is set to 4 and *j* to 3. In line 5, *i* is set to *j* (which is 3). In line 6, *j* is set to 2. The final answer is $i = 3, j = 2$. This example showcases that assignments are carried out once, the variables *i* and *j* are not linked after line 5.

Example 3: ChainAssign

```
1 public class ChainAssign {
2     public static void main(String[] args) {
3         int i = 1, j = 2, k = 3;
4         j = k = 4;
5         i = k = 5;
6     }
7 }
```

On line 3, the variables `i`, `j`, and `k` are initialised to 1, 2, and 3 respectively. Then in line 4, we need to read right-to-left, first `k` is assigned the value 4, and then `j` is also assigned 4. In line 5, `k` is assigned 5, and then `i` is assigned 5. The final result is `i = 5, j = 4, k = 5`. This example shows how assignments can be chained.

Example 4: PostIncrement

```
1 public class PostIncrement {
2     public static void main(String[] args) {
3         int t = 60, x = 22;
4
5         if (t++ - 60 == 0) {
6             x = 20;
7         }
8     }
9 }
```

On line 3, `t` is initialised to 60 and `x` is initialised to 22. In line 5, we first evaluate the if condition: $(60 - 60 == 0)$, which is true, and then add 1 to `t`, so `t` is now 61. Then we enter the if block and set `x` to 20 on line 6. The final result is `t = 61, x = 20`. This example showcases how the post-increment operator is executed last despite its appearance in the middle of a line of code.

Example 5: PreIncrement

```
1 public class PreIncrement {
2     public static void main(String[] args) {
3         int t = 60, x = 22;
4
5         if (++t - 60 == 0) {
6             x = 20;
7         }
8     }
9 }
```

On line 3, `t` is initialised to 60 and `x` to 22. In line 5, we first add 1 to `t`, so it is now 61, and then evaluate the if condition $(61 - 60 == 0)$, which is false. We do not enter the if block. The final result is `t = 61, x = 22`. This example showcases the difference between the post-increment and pre-increment operators.

Example 6: PostDecrementAndCompare

```
1 public class PostDecrementAndCompare {
2     public static void main(String[] args) {
```

```

3     int x = 22;
4     int i = x; int j = 0;
5
6     while (i-- > 0) {
7         i -= 8;
8         j++;
9     }
10 }
11 }

```


1. In lines 3 and 4, x and i are set to 22, and j is set to 0.
 2. Then we come to the while-loop condition on line 6 where we check (22 > 0), which is true, and then decrement i by 1, so it is now 21.
 3. Then we enter the loop block and on line 7 subtract 8 from i (which is now 13). On line 8 we add 1 to j (which is now 1).
 4. Then we check the while-loop condition again: (13 > 0) which is true, and then decrement i by 1, so it is now 12.
 5. Then we enter the loop block again and subtract 8 from i (now 4) and add 1 to j (now 2).
 6. Then we try the loop condition again: (4 > 0) is true, then subtract 1 from i (now 3).
 7. Enter the loop body one more time: subtract 8 from i (now -5) and add 1 to j (now 3).
 8. Then check the loop condition (-5 > 0) is false. Don't forget to subtract 1 from i regardless of the fact the loop condition was false this time.
- The final result is i = -6, j = 3.

These exercises are an example of a “manual walkthrough”. In practice, going through code line-by-line with the values that variables hold (the program state) in mind can often help to figure out how a program works or identify sources of bugs.

3 Exercises

With the manual walkthrough problem-solving technique up your sleeves, try having a go at designing solutions to some concrete programming problems (the kind that might even come up in an internship interview!)

You will have about 5 minutes to work on the exercises individually, after which you will have the opportunity to discuss ideas with peers at your table.

 In designing a solution, avoid writing any code - instead try to explain how it would work in the most intuitive way you can. Think using drawings, diagrams, comics, stories, bullet points, or anything else, be creative! We will do the actual implementation later.

Task 1 - Scrabble Winner

◀ Task

This problem is about the popular board game Scrabble. If you have never played it before - it is a game where players get pieces with letters that they use to form words in a crossword-like puzzle. Each letter is worth a different amount of points according to this table:

A is worth 1 — B is worth 3 — C is worth 3 — D is worth 2

E is worth 1 — F is worth 4 — G is worth 2 — H is worth 4
 I is worth 1 — J is worth 8 — K is worth 5 — L is worth 1
 M is worth 3 — N is worth 1 — O is worth 1 — P is worth 3
 Q is worth 10 — R is worth 1 — S is worth 1 — T is worth 1
 U is worth 1 — V is worth 4 — W is worth 4 — X is worth 8
 Y is worth 4 — Z is worth 10

A word is worth the amount of points that its letters add up to.

For example, the word "gear" is worth $2 + 1 + 1 + 1 = 5$ points.

Other possible word examples could be "apple" (9 points), "queen" (14 points), "koala" (9 points), "object" (17 points), etc.



Figure 1: Scrabble¹

Try thinking of an algorithm that takes a word as input and outputs the number of points that the word is worth.

Hint: consider the input word as an array of characters, so you can process the characters one by one...

[Fun fact: In 2015 a man from New Zealand memorized every French word in the French scrabble dictionary and won the French Scrabble Championship. He still doesn't speak any French. <https://www.npr.org/sections/thetwo-way/2015/07/21/424980378/winner-of-french-scrabble-title-does-not-speak-french>]

The pseudo-code for this particular problem could look like this:

¹Image source: <https://localscrabble.files.wordpress.com/2011/11/scrabble.jpg>

```

1 word <- input (given char array)
2 points <- 0
3
4 for c in input:
5     if c = A or E or N or S or ...
6         points <- points + 1
7     else if c = D or G or ...
8         points <- points + 2
9         ...
10
11 return points

```

Task 2 - Is it weekend yet?

◀ Task



Figure 2: Weekend Face²

Given a day of the week (as a string) and a integer number n , determine whether n days later is a weekend (Saturday or Sunday).

If so, print "WEEKEND!" otherwise print "not weekend yet..." to standard output.

For example, given the inputs "Monday" and 2, your algorithm should figure that 2 days from Monday is Wednesday and so print "not weekend yet..."

As another example, given "Saturday" and 1, the next day after Saturday is Sunday, so your algorithm should output "WEEKEND!".

Here a solution would typically involve assigning a number from 1 to 7 (or 0 to 6) according to what the given day is. Then adding n and finally checking if the result is 6 or 7 (or 5 or 6 if using 0-based system).

There are multiple considerations to take into account, for example if adding n days to the given day goes over the length of the week, performing a mod 7 operation can help to wrap-around the length of the week. Also, nothing is specified about n , only that it is an integer, so a good solution should look out for it being negative.

Task 3 - Optional: esrever

◀ Task

If you went through the first two exercises like a breeze, then this is an optional harder exercise to try. Otherwise, feel free to skip it.

Given a string containing a sentence, how would you output the sentence with the order of words in reverse? For example, given "I like ice cream" the output should be "cream ice like I".

It may help to consider the given string as a character array.

The simplest answer for this exercise involves looping from the end of the string to the beginning.

A pseudo-code solution may look like this:

²Image source: <https://www.groundzeroweb.com/wp-content/uploads/2016/10/Funny-Weekend-Memes-8.jpg>

```

1 sentence <- input
2 last_word <- sentence.length
3
4 for i: sentence.length - 1 to 0:
5     c <- sentence[i]
6
7     if c is a space:
8         print word between i and last_word
9         last_word <- i
10
11 print word between 0 and last_word

```

Here the first insight is that words are separated by spaces, so it is a good idea to split up the string by spaces and print words one by one. Beginning iteration from the end allows to do this with only limited back-tracking, but is not necessary.

A tricky part is recognising that the first word in a sentence is not preceded by a space, so it needs special handling. Without the last line, the solution would only output "cream ice like".

Pair programming

Task 1 - Pairs shuffle!

◀ Task

Now pair up with another partner (either at the same desk or another, that's up to you), but it should be someone different. Make sure you have a computer between the two of you. If the group is an odd number, one person can pair up with a tutor.

Now that you have lots of ideas for the exercises above, let's try implementing the exercises in pairs. With your partner, count the number of Scrabble points in each other's name. The one whose name is worth more points will start in the role of the driver and the other will be the navigator.

Task 2 - Time to code

◀ Task

You will have about 10 minutes to implement a solution to the first task. Then swap roles and try solving the second problem. Everyone should now have IntelliJ installed, but if you're facing technical difficulties this week, you can use this awesome online tool:

<https://www.jdoodle.com/online-java-compiler/>

If you were wondering how to go from a string to a character array - you can use the `toCharArray` function like this:


```

1 String sentence = "I like ice cream";
2 char[] buffer = sentence.toCharArray();

```



Figure 3: Let's get to driving!³

 From here on, you will need to print things to standard output quite often. Typing 'System.out.println();' every time can be really tedious.. Why wasn't it simply called 'print' like in Python..? If you use IntelliJ, fear not, because you can use its "Live Templates" feature to help. Try typing 'sout' and pressing enter. It should autocomplete to 'System.out.println()' for

³Image source: <https://www.nickfreemansolicitors.co.uk/wp-content/uploads/2016/11/child-driving.jpg>

you! You can use 'psvm' to write "public static void main(String[] args)", which is really nifty too. We'll learn more useful features and shortcuts in future tutorials.

Here is a full Java solution for the scrabble exercise:

```
1 public class MyClass {
2     private static int charPoints(char c) {
3         if (c == 'a' || c == 'e' || c == 'i' || c == 'l' || c == 'n'
4             || c == 'o' || c == 'r' || c == 's' || c == 't' || c == 'u')
5             return 1;
6         if (c == 'd' || c == 'g')
7             return 2;
8         if (c == 'b' || c == 'c' || c == 'm' || c == 'p')
9             return 3;
10        if (c == 'f' || c == 'h' || c == 'v' || c == 'w' || c == 'y')
11            return 4;
12        if (c == 'k')
13            return 5;
14        if (c == 'j' || c == 'x')
15            return 8;
16        if (c == 'q' || c == 'z')
17            return 10;
18
19        System.out.println("Unknown character!");
20        return 0;
21    }
22
23    public static void main(String args[]) {
24        // args[0] = object;
25        char[] word = args[0].toCharArray();
26        int points = 0;
27
28        for (int i = 0; i < word.length; ++i) {
29            points += charPoints(word[i]);
30            System.out.print(word[i]);
31        }
32        System.out.println(" is worth " + points + " Scrabble points");
33    }
34 }
```

Here is a sample Java solution for the weekend problem:


```

1  public class MyClass {
2      private static int dayOfWeek(String day) {
3          if (day.equals("Monday"))
4              return 0;
5          if (day.equals("Tuesday"))
6              return 1;
7          if (day.equals("Wednesday"))
8              return 2;
9          if (day.equals("Thursday"))
10             return 3;
11          if (day.equals("Friday"))
12             return 4;
13          if (day.equals("Saturday"))
14             return 5;
15          if (day.equals("Sunday"))
16             return 6;
17
18          System.out.println("Unknown day!");
19          return -1;
20      }
21
22      public static void main(String args[]) {
23          // args[0] = "Monday"
24          String day = args[0];
25          // args[1] = 2
26          int n = Integer.parseInt(args[1]);
27
28          int nextDay = (dayOfWeek(day) + n) % 7;
29
30          if (nextDay == 5 || nextDay == 6) {
31              System.out.println("WEEKEND");
32          } else {
33              System.out.println("not weekend yet...");
34          }
35      }
36  }

```

The full Java solution for the reversing problem could be like this:

```
1 public class MyClass {
2     // print a word between two indices
3     private static void print_word(char[] text, int startIndex, int endIndex) {
4         for (int i = startIndex; i < endIndex; ++i) {
5             System.out.print(text[i]);
6         }
7         System.out.print(" "); // print trailing space
8     }
9
10    public static void main(String args[]) {
11        // args[0] = "I like ice cream";
12        char[] sentence = args[0].toCharArray();
13        int lastWordIndex = sentence.length;
14
15        for (int i = lastWordIndex - 1; i > 0; --i) {
16            if (sentence[i] == ' ') {
17                print_word(sentence, i+1, lastWordIndex); // skip leading space
18                lastWordIndex = i;
19            }
20        }
21        print_word(sentence, 0, lastWordIndex);
22    }
23 }
```

At this point it is worth noting the fact that the Java library provides functionality which makes this task a program with only a few lines compared to a manual solution.

A Java solution using library functions could be like this:

```
1 import java.util.Collections;
2 import java.util.Arrays;
3
4 public class MyClass {
5
6     public static void main(String args[]) {
7         // args[0] = "I like ice cream";
8         String[] words = args[0].split(" ");
9         Collections.reverse(Arrays.asList(words));
10        System.out.println(Arrays.asList(words));
11    }
12 }
```

See this link for more details <https://www.geeksforgeeks.org/reverse-an-array-in-java/>.