Exercises - Unit 6
Control structures: iteration

Group I1E

Year 2017/2018

1. Write manually the result of executing the following loops for `n` equal to 5:

```
for (int i=n; i>=0; i--)
   System.out.println(i*i);
```

```
for (int i=0; i<=n; i++)
   System.out.println((n-i)*(n-i));
```

2. What is the output for the following code when `m=0`, `m=1`, and `m=3`?

```
n = m;
for(i=0; i<n; i++) n--;
System.out.print(i);
```

3. What is the output for the following loop?

```
for (int count=1; count<5; count++)
    System.out.println(2*count);
```

4. What is the output for the following loop?

```
for (int n=10; n>0; n=n-2) {
    System.out.print("Hello");
    System.out.println(n);
}
```

5. What is the output for the following loop?

```
double n=2.0;
for (; n>0; n=n-0.5)
    System.out.print(n);
```

6. What is the output for the following loop?

```
 for (int i=0,j=10; i<j; i++,j--)
    System.out.println("i:"+i+" j:"+j);
```

7. What is the output for the following loop?

```
for(int i=1; i<4; i++) {
    System.out.print(i);
    System.out.print(" ");
    for(j=i; j>=1; j--) {
        System.out.print(j);
        System.out.print(" ");
    }
    System.out.print("\n");
}
```

8. What is the output for the following loop?

```
i=1;
while(i*i<10) {
    j=i;
    while(j*j<100) {
        System.out.print(i + j);
        System.out.print(" ");
        j *= 2;
    }
    i++;
    System.out.print("\n");
}
System.out.print("\n*****");
```

9. The following succession $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$, in which each term is obtained by summing the two previous terms (except for the first two terms, that are predefined as 0 and 1) is called **Fibonacci's succession**, due to Leonardo Fibonacci, an XIII century Italian matematician that described the succession for first time and who extended the arabic number system in Europe.

   You must:

   (a) Define a recurrence which defines how to express each term based on the two previous terms

   (b) Write an iteration in Java, with its initialisation, guard, and body, that, from a given n value, $n \geq 0$, calculates the nth term of Fibonacci's succession

10. The value e^x can be approximately calculated by using the series $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$. The strategy consists of generating each of the terms of the series and accumulating them into a var, until the last generated term is small enough. Moreover, notice that each term can be generated from the previous one, except the first one whose value is 1.

    Write a Java static method that given a real value x and a small real value *epsilon* ($epsilon > 0$), calculates, using the iteration described in the previous strategy, the value e^x, by summing all the generated terms until the last calculated value is lower than *epsilon*. Use the method in program class (in the `main` method) to calculate the value of the e number and compare the calculated value with that given by the `Math.E` constant defined in Java.

11. Write a Java iteration to calculate, using successive subtractions, the quotient and the remainder between two integer numbers a and b, with $a \geq 0$ and $b > 0$. What would happen with your algorithm when initially b is equal to 0? And what would happen when $a < 0$?

12. Write a Java iteration that, only using increment and decrement by one, obtains in the var `difference`, the subtraction of two integer non-negative numbers `minuend` and `subtrahend`. You can assume that `minuend`$\geq$`subtrahend`.

13. Write a Java iteration to calculate the square of a number which is input as data, by only using additions and subtractions.

14. Write a Java static method that **returns** the value of the number resulting of the inversion of the digits of an integer number $n \geq 0$ given as parameter. For example:

$$inversion(0) = 0$$
$$inversion(32356) = 65323$$

    Do it only by using integer numbers operations, i.e., do not employ `String` conversion.

15. Notice that the number of digits in base 10 of an integer number greater than 0 has a direct relation with the integer part of the decimal logarithm of that number.

    (a) Which is that relation?

    (b) Using this relation, write a Java static method that calculates for a given n ($n > 0$), the integer part of $\log_{10}(n)$ without using `Math` methods, i.e., only by counting the digits of the number

    (c) Write a Java static method to calculate the number of digits of a non-negative integer number by using, instead of an iteration, the Java function `Math.log10(double)` (you will need a casting)

16. In the previous problem, when substituting the division by 10 by other different number (e.g., 2) the numeric base changes. Using this fact, write a Java static method to calculate the integer part of the logarithm in a given base b ($2 \leq b \leq 16$) of a given integer number n ($n > 0$). Use the program to calculate $\log_2(1024)$.

17. Implement a new version of the spoof game proposed in Unit 5 (exercise 29) that validates user inputs by using `do-while` loops.

18. Write a Java program class that reads integers until a 0 is inputed and then shows the maximum number that was inputed (excluding the final 0).

19. Write a Java program class whose `main` method reads a word inputed on the keyboard, and uses a class method which **writes on the standard output** sequentially in the same line the n characters of the word, but separated by `'-'`. Thus, if the word is not the empty word, the first character will be printed and after that all the rest of characters will be printed preceded by `'-'`.

    For example, for "Java", the output must be `J-a-v-a`.

20. Write a Java program class whose `main` method asks for a value $n > 0$ and shows on the screen the following figure, where the last line writes as many asterisks as n (in this case, $n = 6$):

```
*
**
***
****
*****
******
```

21. Write a Java program class whose `main` method asks for a natural number and shows on the screen the following figure where the last line has as many asterisks as the double of the input number. Print only spaces and asterisks.

```
        **
      ****
    ******
  ********
**********
```

22. Write a Java program class that asks for two words and writes on the screen how can they be *scrabbled*, i.e., how can they be crossed with a common letter. E.g., for words `java` and `program`, you must show:

```
 p
 r
 o
 g
 r
java
 m

   p
   r
   o
   g
   r
java
   m
```

    Supose first word is in horizontal and second in vertical. If no *scrabbling* is possible, no output is expected.

23. Write a Java program class whose `main` method reads a positive number n from the keyboard and shows on the screen, line by line, the pairs of integers i, j, such that $1 \leq i \leq n$, $1 \leq j \leq n$, and the value of the expression $i + j + 2 \cdot i \cdot j$. Since sum and multiplication are commutative, the expression has the same value for (j, i) than for (i, j), and consequently it will be shown for only one of those pairs. As an example, the output for $n = 4$ must be:

```
Pair 1,1: 1+1+2*1*1 is 4
Pair 1,2: 1+2+2*1*2 is 7
Pair 1,3: 1+3+2*1*3 is 10
Pair 1,4: 1+4+2*1*4 is 13
Pair 2,2: 2+2+2*2*2 is 12
Pair 2,3: 2+3+2*2*3 is 17
Pair 2,4: 2+4+2*2*4 is 22
Pair 3,3: 3+3+2*3*3 is 24
Pair 3,4: 3+4+2*3*4 is 31
Pair 4,4: 4+4+2*4*4 is 40
```

24. Write a Java static method to guess whether a positive integer number $n > 0$ given as parameter is prime or not. Use the algorithms provided in Unit 1 to solve the problem.

25. Write a Java static method which uses the Russian Multiplication method to obtain the product of two integer positive numbers. The Russian Multiplication method only uses double (multiply by 2) and half (divide by 2) operations. The method is:

    - Write `multiplicand` and `multiplier` side by side.
    - Make two columns below each operand and the following rule is applied until `multiplicand==1`
        - Divide the number on the left by 2, ignoring remainder
        - Double the number on the right
    - Write a third column by copying the second column numbers except those in the lines whose first column is an even number.
    - `product` is obtained as the result of summing the elements of the third column.

26. Write a Java static method that calculates $a \cdot n$ by only using sums, and products and divisions by 2. In this case, the following property must be applied: $a \cdot n = (a \cdot 2) \cdot n/2$, when n is even, or $a \cdot n = (a \cdot 2) \cdot n/2 + a$ when n is odd. For example, $4 \cdot 14 = 8 \cdot 7 = 16 \cdot 3 + \underline{8} = 32 \cdot 1 + \underline{16 + 8} = \underline{32 + 16 + 8}$. It is recommended to use a var `result` to keep the partial results that are underlined in the previous example.

27. Write a Java class that simulates a digital clock (`Clock` class). Its constructor will receive four integer numbers, first one for weekday (0 is Monday) and the rest for hour, minute, and second. If a parameter is incorrect, the corresponding attribute would be initialised to 0. Write a method that **shows** on the screen a message with the following format:

    ```
    dayname     hour     minutes     seconds
    ```

    Write a method that increases in one second the current time (take into account changes of minutes, hours and weekdays).

    Write a Java program class whose `main` method will ask the user for the weekday and the hour, e.g., for Saturday, 23:59:58 it will input:

    ```
    5    23    59    58
    ```

    Then, the program will show on the screen the day and time (hour, minutes and seconds) for the following 100 seconds, using the corresponding method of the `Clock` class. For the example given above:

    ```
    Saturday    23    59    59
    Sunday      0     0     0
    Sunday      0     0     1
        ...
    Sunday      0     1     38
    ```

28. Write a Java static method that, given two chars, shows on the screen the different characters that are between the two given chars, both included. For example, given 'a' and 'e', the characters to be shown on the screen are 'a', 'b', 'c', 'd', and 'e'.

29. Capital and lowercase letters are separated in its encoding value by a value d for any enconding. This value d can be calculated as the difference between the code of the lowercase letter and the code of the corresponding capital letter (e.g., between 'a' and 'A'). Thus, for converting any lowercase letter into a capital letter, d is substracted to the code of the lowercase letter.

    Use this information to write a Java static method that receives a line (`String`) and changes all the lowercase letters into capital letters, and vice-versa (without altering the rest of characters), showing the result on the screen and returning the final number of changes.

30. A palindrome is a word that has the same lecture from the beginning to the end and from the end to the beginning; for example, "radar", "redivider", "rotator", and "kayak" are palindromes (among many others). Write a Java static method that receives a `String` that stores a word and returns if it is or not a palindrome.

31. Write in Java a static method that shows on the standard output, line by line, a certain word `s` (parameter) and all the possible variants that appear by moving all its characters towards the left and moving the first character to the last position. For example, for the word "Java" the output must be:

    ```
    Java
    avaJ
    vaJa
    aJav
    ```

    Any of the $n-1$ variants of `s` (where n is the length of `s`) can be written as the substring from the index i to the end followed by a substring from the initial index (0) to $i-1$. These substrings can be calculated by using the `substring` method of the `String` class.

    Remember that the method `substring(int beginIndex)`, when applied on a `String` object, returns the substring from the given index to the end (both included) and that the method `substring(int beginIndex,int endIndex)` returns the substring from index `beginIndex` included to `endIndex` excluded.

    You must use in the algorithm a "for" structure similar to the following one:

    ```
    System.out.println(s);
    int n=s.length();
    for (int i=1; i<=n-1; i++) {
        // Write the i-th variant of s
        // (variant that starts at character i):
        ......
    }
    ```

32. In the previous problem, the word `s` can be considered as the particular case of the variant 0 of `s`, where substring from position 0 to the end is written and the substring from 0 to -1 (empty) is written. Notice that `s.substring(0,0)` (from 0 to 0 excluded) is the empty substring.

    According to this, rewrite the method of the previous problem in a way such that the method code is similar to the following loop, in which initialisation must be completed:

    ```
    int n=s.length();
    for (int i=   ; i<=n-1; i++) {
        // Write the i-th variant of s:
        ......
    }
    ```

33. Rewrite the method of the previous problem in a way such that the ith variant of `s` is written by using the `charAt(int index)` method of the String class: the characters of `s` must be written one by one, from an index i to the last, and after that from index 0 to $i-1$, by using a consecutive pair of "for" loops.

34. Implement a Java static method for encoding strings. The method must receive a string (supposed to be in lowercase) to be encoded and a char with a lowercase letter. The string must be encoded taken into account that 'a' must be substituted by the given char, 'b' by the given char plus 1, etc.; in case the resulting char is higher than 'z', it must be ranged to ['a'-'z']. The resulting encoded string must be returned.

35. Implement in Java a static method that shows in the first line `n 'A'` and `n 'Z'` (where `n` is a parameter), and in the following lines the last character of the previous line goes to the first position and "pushes" the rest of characters, until in the last line all the `'A'` are at the end. For example, with `n` equal to 4:

```
AAAAZZZZ
ZAAAAZZZ
ZZAAAAZZ
ZZZAAAAZ
ZZZZAAAA
```

Suggestion: it is possible to solve the problem taking into account that three groups of letters (of `'Z'`, `'A'`, and `'Z'` again) are written in each line, and the number of letters of each group of `'Z'` depends on the number of line and they could be 0.

It is recommended to use "for" loops.

36. Write a variant of the previous method that does not requiere $\mathtt{n} \geq 0$. In the case of $\mathtt{n} < 0$, the output should be similar but movements will be in the opposite direction. For example:

```
AAAAZZZZ
AAAZZZZA
AAZZZZAA
AZZZZAAA
ZZZZAAAA
```

Notice that in this case there are again three different groups, the first with character c_1, the second with c_2 and the third with c_1. The values of c_1 and c_2, as well as the direction of the movement, depends on the sign on `n`.

37. Justify that function **max(a,b)** is a correct limiting funtion for the **GCD** algorithm presented in the lectures.

38. Demonstrate the termination of the following loop, where `x` is an integer variable with a positive value.

```
/* x>0 */
while (x%2==1)
  x = (3*x+1)/2;
```

Use as limiting function the number of consecutive zeros that appear to the right in the binary representation of `x`.

39. Write a Java static method that implements the **GCD** algorithm presented in the lectures.

40. Given the problem of determining the sum of the n first natural numbers ($n \geq 0$), i.e., to calculate $\sum_{k=0}^{n} k$, write:

    (a) The loop initialization
    (b) The body of the loop
    (c) The finishing condition
    (d) The guard

    What could be an invariant for the loop? What limiting function can you propose?