Lab Assignment 2: Working with Classes, Objects and Methods

Course: Advanced Programming (EE423)
Institution: IGEE, Boumerdes University

October 20, 2022

1 Objectives

Within this lab, you will learn to use the different types of operators (*,/,&, %, ...) and instructions (for-loops, switch, if-else, ...) offered by Java.

2 Assignment

2.1 Exercise 1: if-else-then

Code a game where you ask a user to guess a random number between 1 and 1000. Each time the user makes a guess, the code gives a hint `bigger` or `smaller` until the right number is correctly guessed.

However, the random number changes each time but stays within the range `inf` and `sup` initialized to 1 and 1000. If the hint is `bigger` update the `inf` variable to the guessed value. If the hint is `smaller` update the `sup` variable to the guessed value and regenerate the new random value within this new range. At the end, print how many times the user guessed.

For example:

```
Guess a number between 1 and 1000: 8        Bigger
Guess a number between 8 and 1000: 600      Smaller
Guess a number between 20 and 600: 350
Yes! You've got the right answer after 3 tries.
```

You can use the `if-else-then` statements to compare the guessed value to the expected value.

```
if(condition) {
  // run this if condition is true
} else {
  // run this if condition is false
}
```

You can use the `do-while` loop to create an infinite loop

```
do {
  // infinite loop
} while(true);
```

To read an input in Java, you can use these lines:

```
import java.util.Scanner; //add this line at the beginning of the file

Scanner scanner = new Scanner(System.in);
int a = scanner.nextInt();
```

To generate a random value you can use these lines:

```
import java.util.Random; //add this line at the beginning of the file

Random r = new Random();
r.nextInt(20); // gives a random value between
               // 0(inclusive) and 20(exclusive)
```

2.2 Exercise 2: Integer to Binary conversion

Write a function `String toBinary(int i)` that prints the binary representation of an integer using a loop. For example:

```
System.out.println(toBinary(10));
// outputs a string: 1010
```

Note: to check if your code is correct, you can compare your result with the result that you get using this function: `Integer.toBinaryString(5)`

10	2			
0	5	2		
↖	**1**	2	2	
	↖	**0**	1	2
		↖	**1**	0

Successive divisions by 2 of 10 till we reach 0. We then concatenate the remainders after each division from the last one till the first on.

2.3 Exercise 3: Using Math Library

In Java you can use mathematical functions from the class Math.

```
Math.pow(45,3);
// 45 to the power of 3 (45*45*45)
```

```
Data: a ∈ N, a ≠ 0
Result: String toBinary($a$)
String result = "";
while $a! = 0$ do
    int remainder = a%2;
    a = a/2;
    result = remainder + result;
end
return toBinary($a$) = $a_{base2}$;
```

Algorithm 1: Binary conversion

Write a function `static double power(double nb, int pw)` that calculates `nb` to the power of `pw`, using loops and compare your result with the result of the function `Math.pow` from Java. Notice, that you can use many other math-related functions (`sin`, `cos`, `abs`, `floor`, `ceil`, ...) using the Math library.

You can use the for-loop statement to implement this:

```
for(int i = 0; i < pw; i++) {
  // loop
}
```

2.4 Exercise 4: Loops

Write a function `static int sumSquaredOdd(int n)` that computes the sum of the first `n` odd numbers squared.
For example: `sumSquaredOdd(5)` $= 1^2 + 3^2 + 5^2 + 7^2 + 9^2 = 165$

2.5 Exercise 5: Bit-wise operators

Write a function `void printBitwiseOperators(int a, int b, int c)` that prints:

1. a in binary
2. b in binary
3. c in binary
4. $a|b$ in binary
5. $a\&b$ in binary
6. $a\hat{}b$ in binary
7. $-a$ in binary
8. $b << 1$ in binary
9. $b << 2$ in binary

10. $b >> 1$ in binary

11. $c >> 1$ in binary

12. $c >>> 1$ in binary

```
For Example:
a = 12 b = 10  c = -10
--------------------------------
a in Binary : 0000 0000 0000 1100
b in Binary : 0000 0000 0000 1010
c in Binary : 1111 1111 1111 0110
--------------------------------
a | b : 0000 0000 0000 1110 (or)
a & b : 0000 0000 0000 1000 (and)
a ^ b : 0000 0000 0000 0110 (xor)
~a    : 1111 1111 1111 0011 (not)
b<<1  : 0000 0000 0001 0100 (shift once to the left)
b<<2  : 0000 0000 0010 1000 (shift twice to the left)
b>>1  : 0000 0000 0000 0101 (shift once to the right)
c>>1  : 1111 1111 1111 1011 (shift once to the right, keep the sign)
c>>>1 : 0111 1111 1111 1011 (shift once to the right, ignore the sign)
```

2.6 Exercise 6: Count set bits

Write a function `int countSetBits(int i)` to count the number of bits set to 1 of an integer. For instance, `125 (0b01111101)` has six (6) bits set to 1.
Hint: You can decrement the value and use the & operator to set each bit to zero. Repreat this process till all the bits are set to 0. This means the loop stops if the value = 0 and the number of bits is the number of iterations.

```
125        01111101
124        01111100
125 & 124  01111100 (124)

124        01111100
123        01111011
124 & 123  01111000 (120)

120        01111000
119        01110111
120 & 119  01110000 (112)

112        01110000
111        01101111
112 & 111  01100000 (96)
```

```
96        01100000
95        01011111
96 &  95  01000000 (64)

64        01000000
63        00111111
64 &  63  00000000 (0) END
```

2.7 Exercise 7 : Single Number

Given a list of numbers. All the numbers are repeated only once, except one. Find this element in an array that is not repeated using the xor operator. **Notice**, that if you xor one variable with itself the result will always be 0 ($a\hat{}a = 0$) and if you xor a variable with 0 the result will always be the variable itself ($a\hat{}0 = a$).

```
Input: int[] nums = { 4, 1, 2, 9, 1, 4, 2 }; // an array of int
Output: 9
// You can use this loop
for(int i =0; i<nums.length; i++) {
// calculation
}
```

2.8 Exercise 8 : Get First Set Bit

Given an integer, find the position of the first set-bit (1) from the right.

```
Input: n = 18 (0b10010) Output: 2
```

Hint:

```
0b10010 & 0b00001 =  0b00000 (0)
0b10010 & 0b00010 =  0b00010 (1) found at the second iteration
```

2.9 Exercise 9 : Printing and loops

```
System.out.print("*"); // prints * without appending the new line character
System.out.println("*"); // prints a * with a new line
```

- Write a function `static void line(int n)` that prints a line containing n symbols of *
- Write a function `static void square_fill(int n)` that prints a filled square of n by n symbols of *
- Write a function `static void square_no_fill(int n)` that prints a not filled square of n by n of *

- Write a function `static void triangle(int n)` that prints a triangle with a base and a height of n of *
- Write a function `static void triangle_centered(int n)` that prints a triangle of a height of n but this time the triangle must be centered.

****	*****	*****	*	*
	*****	* *	**	***
	*****	* *	***	*****
	*****	* *	****	*******
	*****	*****	*****	*********
`line(5)`	`square_fill(5)`	`square_no_fill(5)`	`triangle(5)`	`triangle_centered(5)`

3 Conclusion

After finishing this assignment, you should now know how to use the different instructions and operators offered by Java to implement your algorithms.