Increment and Decrement operators + Introduction to Iteration

Objectives

- Recap previous session
- Learn about increment and decrement operators
- Combining assignment and arithmetic
- Introduction to Iteration
- `while` Loops
- Class exercises
- Homework Exercises
- Guidelines

Recap previous sessions

- How does it look the syntax of a simple `if` statement?
- How does it look the syntax of an `if-else` statement?
- Can we have an `if-else` statement inside another conditional statement? If yes, how it is named? If no, why?
- What is a variable?
- Should we always initialize a variable when we declare it?
- What is the difference between a `method call` and a `method declaration`? Give an example
- How can we obtain the remainder of a division operation?

Learn about increment and decrement operators

- By `increment`, we usually mean adding `1` to another variable/value. For example, in the snippet below, incrementing the age variable simply means adding `1` to it:

```
int age = 31;
age = age + 1
```

  - What is happening here is that we want to set the new value for the age variable to be the previous value + 1. That's why we are reusing it in the expression on the left handside.
  - There are 2 syntactic sugar expression to the previous increment operations. The first one allows us to skip the usage of the variable name in the left side expression and it looks like:

```
int age = 31;
age += 1;
```

    - This expression and the previous one are generating the same results.
    - Note: There should be no space between the `+` symbol and the `=` symbol

- The second syntactic sugar is the most important one and also the most popular in various programming language (and JAVA of course 😃), this is the `++` operator. See the snippet below:

```
int age = 11;
age++;
System.out.println(age); // It will display 12
```

- There is also another usage of this operator, namely when you place it in front of the variable:

```
int age = 11;
++age;
System.out.println(age); // It will display 12;
```

- As you can see, in our example they behave the same but there is a very big difference between them which in theory sounds like this:
  - If the `++` symbol is in front of the variable name (`prefix -> ++age`), first the variable is incremented and if on the same line, the variable is used in other expression, it will contain the updated value.
  - If the `++` symbol is after the variable name (`postfix -> age++`), first the variable is used (if there is an expression containing it) with the old value, and after that, the value is updated.
  - For a better understanding, run the snippet below:

```
public class Application {
public static void main(String[] args) {
    System.out.println("Postfix Operator:");
    int age = 11;
    System.out.println(age++);
    System.out.println(age);

    System.out.println("Prefix Operator: ");
    int grade = 12;
    System.out.println(++grade);
    System.out.println(grade);
}
}
```

- By `decrement` we mean subtracting 1 from a certain variable/value. All the same rules as above applies for this decrement operator only that it is a subtraction and not addition.
  - You can also use the `-=` shortcut
  - You can use the postfix decrement operator: `e.g age--`
  - You can use the prefix decrement operator: `e.g --age`
  - Just to clarify it, run the snippet below:

```
    public class Application {

    public static void main(String[] args) {
        System.out.println("Postfix Decrement Operator:");
        int age = 11;
        System.out.println(age--);
        System.out.println(age);

        System.out.println("Prefix Decrement Operator: ");
        int grade = 11;
        System.out.println(--grade);
        System.out.println(grade);
    }
}
```

Combining assignment and arithmetic

- In Java you can combine arithmetic operation and assignment. For example the instruction:

```
age += 3;
```

  is the shortcut for

```
age = age + 3;
```

- Similarly: `items *=2` is a shortcut for `items = items * 2;`
- This applies to all math operators like: `+`, `-`, `*`, `/`

Introduction to iteration

- By iteration we mean the way of running the same task for multiple times, until a specific goal is reached.
- There are multiple iteration constructs in the JAVA language, for this session, we will start with the `while` instruction.
- Before making a deep dive into the syntax of the `while` instruction, let's see a simple example.

1. Create a JAVA program which displays in descending order, first `n` numbers read from the keyboard

- Solution:

```
import java.util.Scanner;

public class Application {

    public static void main(String[] args) {
```

```
            Scanner keyboard = new Scanner(System.in);
            System.out.print("Enter n: ");
            int n = keyboard.nextInt();
            while (n >= 0) {
                System.out.println(n + " ");
                n--;
            }
        }
    }
```

- Now, the explanation for the snippet above is the following:
  - First we read the number from the standard input (or keyboard)
  - Then we use the `while` instruction which executes a block of code repeatedly
    - In our case, the block contains a printing statement and a decrement operations
  - A condition controls how many times the iteration is performed. You can almost read it as plain english: `while n is greater than 0`
  - You can see some similarities between a `while` and an `if` statement as both check for some condition

- The syntax for the while loop is very basic:

```
    while (<condition>) {
        //statements go here
    }
```

- The idea of the `while` loop is that the statements inside its scope are executed as long as the condition is true. But keep in mind that the condition should end at a specific moment in time.
- If we take the snippet above, the flow is like this:
  - Suppose we read 4 from the keyboard
  1. It checks if n is greater or equal to 0 and it passes (n = 4)
  2. Displays 4 (because n = 4)
  3. Decrements n (n will have the value 3)
  4. It checks if n is greater or equal to 0 and it passes (n = 3)
  5. Displays 3 (because n = 3)
  6. Decrements n (n will have the value 2)
  7. It checks if n is greater or equal to 0 and it passes (n = 2)
  8. Displays 2 (because n = 2)
  9. Decrements n (n will have the value 1)
  10. It checks if n is greater or equal to 0 and it passes (n = 1)
  11. Displays 1 (because n = 1)
  12. Decrements n (n will have the value 0)
  13. It checks if n is greater or equal to 0 and it passes (n = 0)
  14. Displays 0 (because n = 0)
  15. Decrements n (will have the value -1)
  16. It checks if n is greater or equal to 0 and it fails because n = -1. Now, the statements inside its scope are skipped

- You can easily see that a lot of stuff got repeated but at each step we were closer and closer to evaluate the condition to `false`. If we do not take care, our loop will run forever and as soon as the JVM will figure it out, we will get an error.

Class exercises

1. Create a JAVA program which will read numbers from the keyboard until the user presses the digit `0`. Next, the program should count how many of the read numbers were even.
   - Sample Input: 1 2 3 4 6 8 3 2 4 5 9 8 0
   - Sample Output: We have read 7 even numbers
   - Solution:

```
    import java.util.Scanner;

    public class Application {

        public static void main(String[] args) {
            Scanner keyboard = new Scanner(System.in);
            System.out.print("Enter a number: ");
            int number = keyboard.nextInt();
            int evenNumbersCount = 0;
            while(number != 0) {
                if(number % 2 == 0) {
                    evenNumbersCount++;
                }
                System.out.print("Enter a number: ");
                number = keyboard.nextInt();
            }
            System.out.println("We have read " + evenNumbersCount + "
even numbers");
        }
    }
```

2. Create a JAVA program which will read n numbers from the Standard Input and then will display on one line the first n natural numbers not equal to 0 in ascending order and on another line, the same numbers but in descending order.
   - Sample Input: 5
   - Sample Output:
     - 1 2 3 4 5
     - 5 4 3 2 1
     - Solution:

```
     import java.util.Scanner;

     public class Application {
```

```
        public static void main(String[] args) {
            Scanner keyboard = new Scanner(System.in);
            System.out.print("Enter a number: ");
            int number = keyboard.nextInt();
            int counter = 1;
            while (counter <= number) {
                System.out.print(counter);
                counter++;
            }
            System.out.println();

            while (number > 0) {
                System.out.print(number);
                number--;
            }
        }
    }
```

3. Create a JAVA program wich will compute the average of n numbers (n read from the keyboard).
    - Sample Input: n = 5
        - 2 5 78 9 11
    - Sample Output: 21.0
    - Solution:

```

```

4. Create a JAVA program which will read `n` numbers from the keyboard and then will verify all of them if they are perfect squares.
    - Sample Input: n = 5
    - Sample Output:
        - 1 Perfect Square
        - 9 Perfect Square
        - 16 Perfect Square
        - 18 Not a perfect square
        - 25 Perfect Square
    - Solution:

```
    import java.util.Scanner;

    public class Application {

        public static void main(String[] args) {
            Scanner in = new Scanner(System.in);
            System.out.print("How many numbers will we read?");
            int numbersRead = in.nextInt();
```

```
            while (numbersRead > 0) {
                System.out.print("Enter a number to check if it is a
Perfect Square: ");
                double number = in.nextDouble();
                double squareRoot = Math.sqrt(number);
                double roundedDownSquareRoot = Math.ceil(squareRoot);
                if(squareRoot - roundedDownSquareRoot == 0) {
                    System.out.println(number + " Perfect Square");
                } else {
                    System.out.println(number + " Not a perfect
square");
                }

                numbersRead--;
            }
        }
    }
```

5. Create a JAVA program wihch will read n numbers from the keyboard and then will display the following pyramid:

```
1
1 2
1 2 3
.............
1 2 3 ... n
```

   - Solution:

```
    import java.util.Scanner;

    public class Application {

        public static void main(String[] args) {
            Scanner in = new Scanner(System.in);
            System.out.print("N = ");
            int n = in.nextInt();
            int i = 1;

            while (i <= n) {
                int line = 1;

                while (line <= i) {
                    System.out.print(line + " ");
                    line++;
                }
                System.out.println();
                i++;
```

```
            }
        }
    }
```

6. Create a JAVA program which will read two nubers: n and p. The program will display all the powers of n which are smaller than p.
    - Sample Input:
        - N = 2
        - P = 21
    - Sample Output: 0, 1, 2, 3, 4
    - Solution:

```
import java.util.Scanner;

public class Application {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("N = ");
        int n = in.nextInt();
        System.out.print("P = ");
        int p = in.nextInt();

        int power = 0;
        while (Math.pow(n, power) < p) {
            System.out.print(power + " ");
            power++;
        }

    }
}
```

Homework exercises

1. Try to imagine what is the result of the following snippets, without using the Eclipse IDE. Afterwards, check it againd the IDE and see if you were right or you need to read the first part again:
    -
```
int score = 12;
System.out.println( score++ * 12 )
System.out.println(++score);
```
    -
```

```

2. Create a JAVA program which will read the integer number `n` from the keyboard and it will display, in descending order, the even numbers which are smaller or equal to `n` but different than 0.
    - Sample Input: 8
    - Sample Output: 8 6 4 2
3. Create a JAVA program which will display the first n odd numbers.
    - Sample Input: 5
    - Sample Output: 1 3 5 7 9
4. Create a JAVA program which will compute the Harmonic Mean of n numbers (n read from the keyboard).
    - Sample Input: 6
        - 1 2 3 4 5 6
    - Sample Output: 2.44
5. Create a JAVA program which will compute `a` to the power `b` without using the `Math.pow` method.
    - Sample Input:
        - a = 3
        - b = 4
    - Sample Output: 81

Guidelines

- As always, try to resolve the class exercises by yourself and then compare your answers with the provided solution
- Everytime you encounter something new, try to write it down, it will help the process of memorizing it.