# Guide 03 - Data structures and persistence

## Goals

- Solving small problems using flow control statements and data structures (before introducing class and object models)
- Decomposition of complexity through methods/functions
- Text manipulation and formatting – Strings *(java.lang.String type objects)*
- Persistence of information beyond program execution

## Topics

- Decision Statements (*if*, *switch*)
- Cycles (*for, while, do .. while*)
- Functions (static methods)
- Java Collections
- Files

## Exercise 1.

Write a program that reads a positive integer from the keyboard and returns the sum of all prime numbers up to that value (inclusive). Note that you must validate the input value by repeating the reading if the value is invalid (positive).
You must implement a function that returns whether a number is a prime number. A natural number is a prime number when it has exactly two distinct natural divisors: the number 1 and itself.

## Exercise 2.

The HighLow game consists of trying to guess a number (integer) between 1 and 100. The program chooses a number at random. Then, the user enters an attempt and the program indicates whether it is too high or too low. This is repeated until the user hits the number. The game should indicate how many attempts have been made and then ask, "Do you intend to continue? Press (Y)es". The program ends if the answer is different from "Y" or "Yes".
*Suggestion: To read a word use the next method: String response = sc.next();*

## Exercise 3.

Using the following code (*CollectionTester.java*) compare the performance of some of the Java collections, such as *ArrayList*, *LinkedList*, *HashSet,* and *TreeSet*.

```
public class CollectionTester {

   public static void main(String[] args) {
      int DIM = 5000;

      Collection<Integer> Col = new ArrayList<>();
      checkPerformance(Col, DIM);
   }

   private static void checkPerformance(Collection<Integer> Col, int DIM) {
      Double start, stop, delta;
      Add
      Start = System.nanoTime(); clock snapshot before
      Is(int i=0; i<DIM; i++ )
         Col.add( i );
      Stop = System.nanoTime();  clock snapshot after
      delta = (Stop-Start)/1e6; convert to milliseconds
      System.Out.println(Col.size()+ ": Add to " +
         Col.getClass().getSimpleName() +" took " + delta + "ms");
      Search
      Start = System.nanoTime(); clock snapshot before
      Is(int i=0; i<DIM; i++ ) {
         int n = (int) (Math.random()*DIM);
         if (!Col.contains(n))
            System.Out.println("Not found???"+n);
      }
      Stop = System.nanoTime();  clock snapshot after
      delta = (Stop-Start)/1e6; convert nanoseconds to milliseconds
      System.Out.println(Col.size()+ ": Search to" +
         Col.getClass().getSimpleName() +" took " + delta + "ms");
      Removes
      Start = System.nanoTime(); clock snapshot before
      Iterator<Integer> Iterator = Col.iterator();
      while (Iterator.hasNext()) {
          Iterator.next();
          Iterator.remove();
      }
      Stop = System.nanoTime();  clock snapshot after
      delta = (Stop-Start)/1e6; convert nanoseconds to milliseconds
      System.Out.println(Col.size() + ": Remove from "+
         Col.getClass().getSimpleName() +" took " + delta + "ms");
   }
}
```

a) Adapt the program to measure the results for various dimensions of the collection (for example, by creating a table similar to the following). Analyze the results by comparing structures and operations.

| Collection | 1000 | 5000 | 10000 | 20000 | 40000 | 100000 |
|---|---|---|---|---|---|---|
| ArrayList | | | | | | |
| add | 0,5 | ... | | | | |
| Search | 11,5 | | | | | |
| Removes | 1,2 | | | | | |
| LinkedList | | | | | | |
| ... | | | | | | |

Tips: Modify the checkPerformance method to return the 3 measurements (add, search, and remove) and remove all println statements within this method.

```
private static double[] checkPerformance(Collection<Integer> col, int DIM) {
...
```

## Exercise 4.

Start by implementing a program that reads from the keyboard the grades of the practical (*gradeP*) and theoretical (*gradeT*) component of a class.
Define a function to calculate a student's final grade from the grades of the practical and theoretical components, as follows:

- 66 (failed by minimum grade), if he/she has obtained less than 7.0 in at least one of the components;
- 0.4 * *gradeT* + 0.6 * gradeP (rounded to the nearest unit) in other cases.

Informed by the results of exercise 3.3., create a data structure to store the grades of the theoretical and practical components of all students. Process the notes and print the results in the following format:

```
gradeT gradeP finalGrade
  11.3    9.3     10
  16.7    5.1     66
   7.8   18.9     14
  10.6   15.9     14
  16.9    5.9     66
   1.9   12.7     66
  17.6    4.8     66
   0.7   12.1     66
   8.7    8.6      9
  19.2    1.4     66
  17.5    3.4     66
  11.6   11.4     11
   7.2    8.5      8
   1.9    1.4     66
  19.3   14.9     17
     0   12.1 12
```

a) Alter the program to read these values from a file in which each line has (separated by "\t") name, Tnote and Pnote – in that order
b) Print on the screen which  students have passed this subject

## Exercise 5.  (LLM exploration)

Using a public AI engine (e.g.,  ChatGPT, Deepseek, other) askand:

"Give me a sample java code that reads a tab separated file containing name, gradeP and gradeT in each line and in the end calculates the average per parameter . Should store also present the names of the persons with gradeT above the overall average without using Records. "

a) Execute the program
a) Asking the AI engine "can you use records in the solution?"
b) Analise the response and execute the program
c) Try changing the program without using AI to present students with final grade above the class average

Note: create your own test file or ask the public AI engine to deliver a separate file.

## Exercise 6.

Write a program that reads from the keyboard a date in the format *mm/yyyy* (validating the values) and the day of the week that the month begins (1 = Monday, 2 = Tuesday, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday, 7 = Sunday). The program must then present on the monitor the calendar of that month in the format presented below. Implement the desired functionality with three functions: reading values, with validation, calculating days in the month, considering leap years, and printing results.

```
   February 2025
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28
```

- Change the program to take today's date on the day of the week if a negative number is entered

How to Determine Day of Week by Passing Specific Date in Java?
https://www.baeldung.com/java-get-day-of-week

Get the day of week for a particular date in Java
https://www.tutorialspoint.com/get-the-day-of-week-for-a-particular-date-in-java

## Exercise 7.

The following code excerpt allows you to read all the words (word for word) of a text file, which must be located in the project folder. You can create this file with a text editor or use any java code file.

```
public static void main(String[] args) throws IOException{
    Scanner input = new Scanner(new FileReader("words.txt"));
    while (input.hasNext()) {
        String Word = input.next();
        System.Out.println(Word);
    }
}
```

a) Test the code snippet to list the contents of the file.
b) Store all words longer than 2 characters in a suitable data structure.
c) List all words ending in 's'.
d) Remove from the structure all words that contain characters other than letters.