Lab 1
CSC/CPE102 Review: Generic methods, Input management, Exception handling

I. Create a new directory for this lab (call it Lab1).

II. Design and implement a class (call it ListWork) containing three static methods: two generic methods search and print, and the main method.

- **search**: This is a generic method. Given an array of objects and a target object of some class, the search method returns true if the target is in the array and false otherwise. The signature of this method is the following:

  ```java
  public static <T> boolean search (T[] arr, T target)
  ```

- **print**: This is a generic method. Given an array of objects of some class, the print method outputs all elements of the array on the screen, one element per line. The signature of this method is the following:

  ```java
  public static <T> void print (T[] arr)
  ```

- **main**: In this method you will create a list and test if your search and print are working correctly. Here are the steps you are asked to have in main:

  1. Define an array to hold 10 integers.
     
     **Note**: since the intention is to use the generic search and print methods (these methods work with objects), you need to define an array of Integer type elements and not of type int.

  2. Prompt the user to input values, among which at least 10 integers.

  3. Using a loop, fill the array (input values from the keyboard and store in the array until it is filled). If a non-integer is inputted***, it should be IGNORED (not counted) and the work should be continued.
     
     **Note**: 1. there may be 0, 1, or more values on a line, and there may be more than one lines in the input.
     
     2. all values (integer or non-integer) inputted after the 10-th integer need to be skipped (i.e. read through) so they won’t cause problems/confusions for the next segment of your code.

  4. In a loop, for as long as the user wants to (ask the user if they want to search for an element):
     
     a) prompt the user to input a target to be searched in the array
     
     b) input the target and invoke the search method to find the target in the array. Output feedback to the user (target found or not). If a non-integer is inputted as a target***, output an error message – DON’T do any search for that invalid value.
     
     **Note**: For simplicity, in this exercise let’s agree that there will be exactly one value inputted as target.

     Once the user chooses not to do search any more, move to step 5.

  5. Print the list by invoking the print method.

**Important**: for the purposes of THIS exercise, you are NOT allowed to use the hasNextInt method of the Scanner class or any other method that checks the value’s type. To take care of the non-integer input values, you are **required** to use exception handling. When using the nextInt method of the Scanner class, if the inputted value is not an integer, an InputMismatchException type exception is thrown. You need to catch that exception and handle it with the appropriate action – if this is in the input loop (step 3), read the invalid value with next method and make sure the counter is appropriately set; and if this is the inputting of the target value (step 4b), print out an error message and finish the loop iteration (but not the whole loop).

  **Note**: InputMismatchException class is located in the java.util package. You need to import it.

III. Compile your program.

IV. Test your program thoroughly: Make sure the program works as expected. Here are some test cases:

  **Test cases for input values to be stored in the array:**
  
  a) there are exactly 10 values in the input and all of them are integers
  
  b) there are 10 integers and some non-integers (floats and/or strings) in the input
  
  c) there are more than 10 integers among input values (there may or may not be non-integer values)
  
  d) input values are inputted all from one line
  
  e) input values are inputted from more than one line (with possibly empty lines among them)

  **Test cases for the target value:**
  
  a) the target is an integer found in the list (make sure to check this functionality for differently located values – the target is the first, last, middle element, or a value that occurs multiple times in the list)
  
  b) the value is an integer not found in the list
  
  c) the value is not an integer.