Java module 2

Exercises Day 2

1.1 - Arrays	Find the maximum
Instructions	Write a function that finds the maximum number in an array of integers. Then use this function in a program that requests 5 integers to the user, stores them in an array and outputs the maximum.

Expected output	Number 1:
	>>> 9
	Number 2:
	>>> 167
	Number 3:
	>>> -43
	Number 4:
	>>> 33
	Number 5:
	>>>63
	The maximum is 167

Solution	See code below

```java
import java.util.Scanner;

public class Ex11 {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int[] numbers = new int[5];

        for (int i = 0; i < 5; i++) {
            System.out.print("Enter number " + (i+1) + ": ");
            numbers[i] = scanner.nextInt();
        }

        System.out.println("The highest number is " + getHighest(numbers));
        scanner.close();
    }

    public static int getHighest(int[] array) {
        int max = array[0];
        for (int i = 1; i < array.length; i++) {
            if (max < array[i]) {
                max = array[i];
            }
        }
        return max;
    }
}
```

1.2 - Lists	Store names
Instructions	Write two functions: - The first one should request names from the user, store them in a list and return this list. - The second one should receive a list as a parameter and print the names stored within the list. Then write a program that will call both functions.

Expected output

```
How many names would you like to store?
>>> 3
Name 1:
>>> Ana
Name 2:
>>> Carol
Name 3:
>>> John
The names are Ana, Carol, John
```

Solution

```java
import java.util.ArrayList;
import java.util.Scanner;

public class Ex12 {

    public static void main(String[] args) {
        printNamesList(getNamesList());
    }

    public static ArrayList<String> getNamesList() {
        ArrayList<String> namesList = new ArrayList<>();

        Scanner scanner = new Scanner(System.in);
        System.out.println("How many names do you want to store? ");
        int amountNames = scanner.nextInt();
        scanner.nextLine();
        for(int i = 0; i < amountNames; i++) {
            System.out.println("Enter a name: ");
            namesList.add(scanner.nextLine());
        }

        scanner.close();
        return namesList;
    }

    public static void printNamesList(ArrayList<String>
namesList) {
        System.out.print("The names are ");
        for (int i = 0; i < namesList.size(); i++) {
            System.out.print(namesList.get(i));
            if (i != namesList.size() - 1) {
                System.out.print(", ");
            }
        }
    }
}
```

2 Linear search	Search for a number
Instructions	Create a variable with the following array: int[] myArray = {1, 92, 23, 404, 5, 1027}; Create a function that receives an array of integers and one integer to search as parameters. The function should return the position of the integer in the array if it is found, or -1 if not found. Create a program that asks the user to input a number and replies with the position of the number in myArray.
Expected output 1	Enter a number: >>> 23 The number is in position 2
Expected output 2	Enter a number: >>> 17 The number was not found

Solution	`import java.util.Scanner;`
	
	`public class Ex2 {`

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

        int[] myArray = {1, 92, 23, 404, 5, 1027};

        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = scanner.nextInt();
        int numberPosition = findNumber(myArray, number);
        if (numberPosition == -1) {
            System.out.println("The number was not found");
        } else {
            System.out.println("The number is in position "
+ numberPosition);
        }

        scanner.close();
    }

    /**
     * This function will search for a number in an array.
     * It will return the position of the number in the
array if it was found or -1 if it was not found.
     * @param array the array in which to search.
     * @param number the number to search.
     * @return the position of the number in the array or
-1 if it was not found.
     */
    public static int findNumber(int[] array, int number)
{
        for (int i = 0; i < array.length; i++) {
            if (number == array[i]) {
                return i;
            }
        }
        return -1;
    }
}
```

3 Sorting	Sort an array of numbers
Instructions	Create a variable with the following array:

	int[] myArray = {1, 92, 23, 404, 5, 1027, -20, -3, 209}; Create a program that is able to sort myArray using Bubble sort, Selection sort or Insertion sort.
Expected output	The sorted array is: -20, -3, 1, 5, 23, 92, 209, 404, 1027

Solution

```
import java.util.Arrays;

public class Ex3 {
    public static void main (String[] args) {

        int[] myArray = {1, 92, 23, 404, 5, 1027, -20, -3,
209};

        //Print the sorted array using Bubble Sort
        System.out.println("The sorted array is: " +
Arrays.toString(bubbleSort(myArray)));

        //Print the sorted array using Selection Sort
        System.out.println("The sorted array is: " +
Arrays.toString(selectionSort(myArray)));

        //Print the sorted array using Insertion Sort
        System.out.println("The sorted array is: " +
Arrays.toString(insertionSort(myArray)));
    }

    public static int[] bubbleSort(int[] inputArray) {
        int[] array = inputArray.clone();
        int n = array.length;
        boolean swapped; // Flag to check if any elements
were swapped in the last iteration

        // Outer loop to iterate over the array
        for (int i = 0; i < n - 1; i++) {
            swapped = false; // Initialize swapped as false
for each iteration

            // Inner loop to perform comparisons and swap
adjacent elements if necessary
            for (int j = 0; j < n - i - 1; j++) {
```

```
        // Compare adjacent elements and swap if
they are in the wrong order
        if (array[j] > array[j + 1]) {
          int temp = array[j];
          array[j] = array[j + 1];
          array[j + 1] = temp;
          swapped = true; // Set swapped to true
if a swap occurs
        }
      }

      // If no two elements were swapped in the inner
loop, the array is already sorted
      if (!swapped) {
        break;
      }
    }
    return array;
  }

  public static int[] selectionSort(int[] inputArray) {
    int[] array = inputArray.clone();
    int n = array.length;

    // One by one move boundary of unsorted subarray
    for (int i = 0; i < n - 1; i++) {
      // Find the minimum element in unsorted array
      int minIndex = i;
      for (int j = i + 1; j < n; j++) {
        if (array[j] < array[minIndex]) {
          minIndex = j;
        }
      }

      // Swap the found minimum element with the
first element
      int temp = array[minIndex];
      array[minIndex] = array[i];
      array[i] = temp;
    }
    return array;
    }

    public static int[] insertionSort(int[] inputArray) {
        int[] array = inputArray.clone();
        int n = array.length;

        // Traverse through the array starting from the
second element
        for (int i = 1; i < n; i++) {
            int key = array[i]; // Select the current
element to be inserted
            int j = i - 1;

            // Move elements of array[0..i-1], that are
greater than key,
            // to one position ahead of their current
position
            while (j >= 0 && array[j] > key) {
                array[j + 1] = array[j];
                j--;
            }
            array[j + 1] = key; // Insert key into its
correct position
        }
        return array;
    }
}
```