Lab 5 : Arrays and Methods – Selected Solutions

Exercise 2: Write a program that will output the command line arguments in the same format as above, including the argument index and the quotes around the argument values.

```
public class PrintArguments {
   public static void main(String[] args) {
      int index = 0;
      while(index < args.length) {
         System.out.println("args[" + index + "] = \""
               + args[index] + "\"");
         index++;
      }
   }
}
```

Exercise 4: Modify the program you created in Exercise 3 so that once the program has started, it reads integers from keyboard input and outputs whether or not the number that has been input is contained in the original command-line numbers. Make sure your program has an appropriate stopping condition, for example when the user enters "exit" that the program stops.

```
import java.util.Scanner;

public class ArgumentSearch {
   public static void main(String[] args) {
      int[] numbers = new int[args.length];
      int index = 0;
      // read the arguments and store as integers
      while(index < args.length) {
         numbers[index] = Integer.parseInt(args[index]);
         index++;
      }

      Scanner keyboard = new Scanner(System.in);
      while(keyboard.hasNext()) {
         System.out.print("Enter a value: ");
         String input = keyboard.next();
         if(input.equals("exit")) {
            break;
         }

         int value = Integer.parseInt(input);
         index = 0;
         boolean found = false;
         while(index < numbers.length) {
            if(numbers[index] == value) {
               found = true;
               break;
            }
            index++;
         }

         if(found) {
            System.out.println("That number is in the array.");
         } else {
            System.out.println("That number is not in the array.");
         }
      }
```

```
38
39          System.out.println("Exiting program.");
40      }
41  }
```

Exercise 5: Write a class called SortArray to `sort` a given array, and then print the values out. You will need to use the Array.sort() method to sort an array. Please check this method from the `Arrays` API for details.

```
1   import java.util.Arrays;
2
3   public class SortArray {
4       public static void main(String[] args) {
5           int[] givenArray = new int[] {2, 6, 9, 10, -7, 12, -3, 0};
6
7           Arrays.sort(givenArray);
8           String readable = Arrays.toString(givenArray);
9
10          System.out.println("Result: " + readable);
11      }
12  }
```

Exercise 7: Implement the `calculatePercentage` method in a class called `Scores`. Then write a program takes two command-line arguments and makes use of the `calculatePercentage` method to calculate the percentage score of the input numbers, where the first number is the score and the second number is the maximum possible score.

```
1   public class Scores {
2       public static void main(String[] args) {
3           int inputScore = 0;
4           int intputTotal = 0;
5
6           // Read the two integers from arguments
7           inputScore = Integer.parseInt(args[0]);
8           intputTotal = Integer.parseInt(args[1]);
9
10          // Print the percentage
11          if (intputTotal > 0) {
12              double percentage = calculatePercentage(inputScore, intputTotal);
13              System.out.println("Percentage: " + percentage);
14          }else {
15              System.out.print("Total must be a positive number.");
16          }
17      }
18
19      public static double calculatePercentage(int score, int total) {
20          return ((double)score / total) * 100.0;
21      }
22  }
```

Exercise 8: Write a method prototype for each of the following methods:

a) A method that accepts two integers and returns the sum of the numbers

```
1   public int calculateSum(int a, int b)
```

b) A method that accepts three words and returns all words together, each separated by a space

```
public String joinWords(String first, String second, String third)
```

d) A method that accepts an array of real numbers and returns the minimum of all the numbers

```
public double findMinimum(double[] values)
```

g) A method that prints the sum of three randomly generated numbers

```
public void printRandomSum()
```

Exercise 9: Write a full method (including implementation) for the following methods. Try using your methods to see if they work as expected. Do you see any problems with any of the methods? Can you suggest improvements to them?

a) A method that will accept one number. If the number is odd, the number will be set to zero, otherwise the number is left alone.

```
// You cannot change the original value in place, so you need to
// return a changed value.
public int setOddToZero(int value) {
   if(value % 2 != 0) {
      return 0;
   }
   return value;
}
```

c) A method that accepts an array of numbers and sets the odd numbers to zero. Note: this method is **destructive**; i.e. it will modify the original array.

```
// This method will change the original array in place, so
// no need to return anything.
public void setAllOddToZero(int[] values) {
   for(int i = 0; i < values.length; i++) {
      if(values[i] % 2 != 0) {
         values[i] = 0;
      }
   }
}
```

d) A method that accepts an array of numbers and returns only the even numbers. Note: this method is **non-destructive**; i.e. it will not modify the original array.

```
public int[] getEvenNumbers(int[] values) {
   int evenCount = 0;
   // Count the number of even numbers
   for(int i = 0; i < values.length; i++) {
      if(values[i] % 2 == 0) {
         evenCount++;
      }
   }
   // Fill a new array with those even numbers
   int[] onlyEvens = new int[evenCount];
   int evenIndex = 0;
   for(int i = 0; i < values.length; i++) {
      if(values[i] % 2 == 0) {
         onlyEvens[evenIndex] = values[i];
```

```
            evenIndex++;
        }
    }
}
```

Extension 1: Write a `CaesarCypher`: a program that can rotate the letters in a string around the alphabet by n places, where n is provided. For example, given the string "hello" and $n = 1$, the cypher should rotate 'h' to 'i', 'e' to 'f', 'l' to 'm' and 'o' to 'p'. The letters should wrap around, so rotating 'z' should move it back to the beginning of the alphabet again.

Preserve the case! The letters should stay in the same case.

Useful to know to answer this one:

```
    char ch = 'a';
    Character.isLowerCase(ch); // true iff ch is lower case
```

Also it's useful to know that the characters are stored in order, so the "value" of 'a' is one less than the "value" of 'b', etc.

```
public class CaesarCypher {
    public static void main(String[] args) {
        String input = args[0];
        int cypherDistance = Integer.parseInt(args[1]);

        // get the number of characters in the alphabet
        int alphabetSize = 'z' - 'a' + 1;

        int index = 0;
        String cypher = "";
        while(index < input.length()) {
            char letter = input.charAt(index);

            // leave character if it is not a letter
            if(!Character.isLetter(letter)) {
                cypher += letter;
                index++;
                continue;
            }

            // if the character is uppercase, calculations will be
            // made with respect to uppercase 'A'; otherwise with
            // respect to lowercase 'a'
            char offset = 'A';
            if(Character.isLowerCase(letter)) {
                offset = 'a';
            }

            // reduce the letter to a number from 0 to 25
            letter = (char) (letter - offset);

            // add the cypher distance to the character
            // and wrap around using modulus (still 0 to 25)
            letter = (char) ((letter + cypherDistance) % alphabetSize);

            // return the letter back to its original position
            letter = (char) (letter + offset);

            cypher += letter;
```

```
40              index++;
41          }
42
43          System.out.println("Cypher: " + cypher);
44      }
45  }
```