Entornos de desarrollo
Bloque 2
Tema 3: Manejo de arrays y cadenas

Soluciones

Ejercicios propuestos

2.3.1.1. Crea un programa llamado *MatrixAddition* que pida al usuario que introduzca 2 matrices bidimensionales de 3x3 y muestre el resultado de sumarlas. Recordad que para sumar dos matrices se deben sumar sus celdas una a una:

```
result[i][j] = matrixA[i][j] + matrixB[i][j]
```

```
import java.util.Scanner;

public class MatrixAddition
{
    static final int MATRIX_SIZE = 3;

    public static void main(String[] args)
    {
        int[][] matrixA, matrixB, matrixResult;

        Scanner sc = new Scanner(System.in);

        matrixA = new int[MATRIX_SIZE][MATRIX_SIZE];
        matrixB = new int[MATRIX_SIZE][MATRIX_SIZE];
        matrixResult = new int[MATRIX_SIZE][MATRIX_SIZE];

        System.out.println("Matrix A:");

        for (int i = 0; i < matrixA.length; i++)
        {
            for (int j = 0; j < matrixA[i].length; j++)
            {
                System.out.println("Enter row " + (i + 1) +
                ", column " + (j + 1));
                matrixA[i][j] = sc.nextInt();
            }
        }

        System.out.println("Matrix B:");

        for (int i = 0; i < matrixB.length; i++)
```

```
        {
            for (int j = 0; j < matrixB[i].length; j++)
            {
                System.out.println("Enter row " + (i + 1) +
                ", column " + (j + 1));
                matrixB[i][j] = sc.nextInt();
            }
        }

        for (int i = 0; i < matrixResult.length; i++)
        {
            for (int j = 0; j < matrixResult[i].length; j++)
            {
                matrixResult[i][j] = matrixA[i][j] + matrixB[i][j];
            }
        }

        System.out.println("Result:");

        for (int i = 0; i < matrixResult.length; i++)
        {
            for (int j = 0; j < matrixResult[i].length; j++)
            {
                System.out.print(matrixResult[i][j] + " ");
            }

            System.out.println();
        }
    }
}
```

2.3.1.2. Crea un programa llamado *MarkCount* que pida al usuario que introduzca 10 notas (enteros entre 0 y 10). El programa deberá mostrar cuantas notas de cada tipo se han introducido. Por ejemplo si se introducen estas notas: 1, 7, 5, 7, 2, 6, 7, 3, 5, 8, entonces el programa deberá mostrar:

```
Notas por categoría:
1: 1 notas
2: 1 notas
3: 1 notas
4: 0 notas
5: 2 notas
6: 1 notas
7: 3 notas
8: 1 notas
9: 0 notas
10: 0 notas
```

```java
import java.util.Scanner;

public class MarkCount
{
    /* Solution 1
     *
     * We use an array to store the marks, and another array to store
     * the count of each mark

    public static void main(String[] args)
    {
        int[] marks, markCounter;
        Scanner sc = new Scanner(System.in);

        marks = new int[10];
        markCounter = new int[11];

        System.out.println("Enter 10 marks:");
        for(int i = 0; i < marks.length; i++)
        {
            marks[i] = sc.nextInt();
        }

        for(int i = 0; i < marks.length; i++)
        {
            markCounter[marks[i]]++;
        }

        for(int i = 0; i < markCounter.length; i++)
        {
```

```
            System.out.println(i + ": " + markCounter[i] + " marks");
        }
    }
    */

    /* Solution 2
     *
     * We use just an array to store the marks, and then we use a nested
     * for to explore the marks from 0 to 10 and see how many times they
     * appear in the array
     *
    public static void main(String[] args)
    {
        int[] marks;
        Scanner sc = new Scanner(System.in);

        marks = new int[10];

        System.out.println("Enter 10 marks:");
        for(int i = 0; i < marks.length; i++)
        {
            marks[i] = sc.nextInt();
        }

        for (int i = 0; i <= 10; i++)
        {
            int counter = 0;
            for (int j = 0; j < marks.length; j++)
            {
                if (marks[j] == i)
                    counter++;
            }
            System.out.println(i + ": " + counter + " marks");
        }

    }
    */

    /* Solution 3
     *
     * We just use the array of counters, and every time the user types
     * a new mark, we just increase the counter of the corresponding
     * mark
     */

    public static void main(String[] args)
    {
        int[] markCounter;
        Scanner sc = new Scanner(System.in);

        markCounter = new int[11];
```

```
        System.out.println("Enter 10 marks:");
        for(int i = 0; i < 10; i++)
        {
            int mark = sc.nextInt();
            markCounter[mark]++;
        }

        for(int i = 0; i <= 10; i++)
        {
            System.out.println(i + ": " + markCounter[i] + " marks");
        }
    }
}
```

2.3.2.1. Crea un programa llamado *SortJoin* que pida al usuario que introduzca una lista de nombres separados por espacios en blanco. El programa deberá separar la cadena, ordenar los nombres alfabéticamente y los mostrará separados por comas. Por ejemplo, si el usuario introduce esta lista de nombres: `Susan Kailey William John`, se mostrará `John, Kailey, Susan, William`.

```
import java.util.Scanner;

public class SortJoin
{
    public static void main(String[] args)
    {
        String input;
        String[] names;

        Scanner sc = new Scanner(System.in);

        // 1. Read names

        System.out.println("Enter the names separated by whitespaces:");
        input = sc.nextLine();

        // 2. Split names

        names = input.split(" ");

        // 3. Sort names

        for(int i = 0; i < names.length - 1; i++)
        {
            for (int j = i + 1; j < names.length; j++)
            {
                if(names[i].compareTo(names[j]) > 0)
                {
                    String aux = names[i];
                    names[i] = names[j];
                    names[j] = aux;
                }
            }
        }

        // 4. Show result

        for (int i = 0; i < names.length; i++)
        {
            System.out.print(names[i]);
            if (i < names.length - 1)
                System.out.print(",");
        }
```

```
        // Alternatively, we can sort the names with
        // Arrays.sort(names);
        // And then join them with commas with:
        // String output = String.join(",", names);
    }
}
```

2.3.2.2. Crea un programa llamado *CheckMessages* que pida 10 cadenas al usuario y las guarde en un array. Después de esto, se deberá reemplazar cada aparición de la palabra "Eclipse" por "NetBeans". El programa deberá mostrar las cadenas almacenadas en el array ya actualizadas.

```
import java.util.Scanner;

public class CheckMessages
{
    static final int SIZE = 10;

    public static void main(String[] args)
    {
        String[] texts = new String[SIZE];

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter 10 texts:");

        for(int i = 0; i < SIZE; i++)
        {
            texts[i] = sc.nextLine();
        }

        for(int i = 0; i < SIZE; i++)
        {
            if (texts[i].contains("Eclipse"))
                texts[i] = texts[i].replace("Eclipse", "NetBeans");

            System.out.println(texts[i]);
        }
    }
}
```

2.3.2.3 Crea un programa llamado *LispChecker*. LISP es un lenguaje de programación donde cada instrucción está dentro de paréntesis. Esta podría ser una instrucción en LISP:

```
(let ((new (x-point a y))))
```

Se debe implementar un programa que coja una cadena con instrucciones LISP (una sola cadena) y comprobar que los paréntesis son correctos (o sea, que el número de paréntesis que se abren sea el mismo que los que se cierran)

```
import java.util.Scanner;

public class LispChecker
{
    public static void main(String[] args)
    {
        String lispText;
        int index = 0, brackets = 0;
        boolean ok = true;
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter LISP text:");
        lispText = sc.nextLine();

        while(index < lispText.length() && ok)
        {
            if(lispText.charAt(index) == '(')
                brackets++;
            else if(lispText.charAt(index) == ')')
                brackets--;

            // If brackets < 0, then we have closed a bracket without
            // opening it before
            if (brackets < 0)
                ok = false;

            index++;
        }

        if (brackets == 0 && ok)
            System.out.println("OK");
        else
            System.out.println("Error");
    }
}
```