Entornos de Desarrollo
Bloque 2
Tema 2: Estructuras de control. Condiciones y bucles

Soluciones

> **Ejercicios propuestos**
>
> **2.2.1.1.** Crea un programa llamado *MarkCheck* que pida al usuario que introduzca 3 notas. El programa mostrará por pantalla uno de estos mensajes, dependiendo de las notas:
>
> - *Todas las notas son mayores o iguales a 4*
> - *Algunas notas no son mayores o iguales a 4*
> - *Ninguna nota es mayor o igual a 4*

```
import java.util.Scanner;

public class MarkCheck
{
    public static void main(String[] args)
    {
        int mark1, mark2, mark3;

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter 3 marks:");
        mark1 = sc.nextInt();
        mark2 = sc.nextInt();
        mark3 = sc.nextInt();

        if (mark1 >= 4 && mark2 >= 4 && mark3 >= 4)
        {
            System.out.println("All marks are greater or equal than 4");
        } else if (mark1 < 4 && mark2 < 4 && mark3 < 4) {
            System.out.println("No mark is greater or equal than 4");
        } else {
              System.out.println("Some marks are not greater or equal than
4");
        }
    }
}
```

Versión utilizando un Array

```
import java.util.Scanner;

public class MarkCheckArrays
{
    public static void main(String[] args)
    {
        int marks[] = new int[3];
        boolean allGreater, allLower;

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter 3 marks:");
        for(int i = 0; i < 3; i++)
        {
            marks[i] = sc.nextInt();
        }

        /* We use these two boolean values to determine if all the
         * elements of the array are greater or lower than 4,
         * respectively */

        allGreater = true;
        allLower = true;

        for(int i = 0; i < 3; i++)
        {
            if (marks[i] < 4)
                allGreater = false;
            else if (marks[i] >= 4)
                allLower = false;
        }

        if (allGreater)
        {
            System.out.println("All marks are greater or equal than 4");
        } else if (allLower) {
            System.out.println("No mark is greater or equal than 4");
        } else {
              System.out.println("Some marks are not greater or equal than
4");
        }
    }
}
```

2.2.1.2. Crea un programa llamado *GramOunceConverter* que mejorará el ejercicio 2.1.6.2 del tema anterior. En este caso, el usuario introducirá un peso (float), y una unidad de medida (`g` para gramos, `o` para libras *ounces*). Entonces, dependiendo de la unidad elegida, el programa convertirá el peso en la otra unidad de medida. Por ejemplo, si el usuario introduce un peso de 33 y elige `o`, entonces el programa convertirá 33 libras a gramos. Se debe resolver usando la estructura `switch`. Si la unidad introducida es diferente a `g` o `o`, entonces el programa deberá mostrar un mensaje de error: "Unidad no reconocida" y ningún resultado.

```
import java.util.Scanner;

public class GramOunceConverter2
{
    static final float GRAMS_OUNCES = 28.3495f;

    public static void main(String[] args)
    {
        float weight, result;
        String unit;

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter weight:");
        weight = sc.nextFloat();

        System.out.println("Enter unit (g or o):");
        unit = sc.next();

        switch(unit)
        {
            case "g":
                result = weight / GRAMS_OUNCES;
                System.out.printf("%.2f ounces", result);
                break;
            case "o":
                result = weight * GRAMS_OUNCES;
                System.out.printf("%.2f grams", result);
                break;
            default:
                System.out.println("Unexpected unit");
        }
    }
}
```

2.2.2.1. Crea un programa llamado *GroupPeople* que pida al usuario que introduzca cuanta gente va a acudir a una conferencia. El programa debe crear grupos de preferiblemente 50 personas. Cuando esto no sea posible, se probará a crear grupos de 10 y finalmente grupos de 1 persona. El programa deberá mostrar cuantos grupos de cada tipo serían necesarios. Por ejemplo, si van a ir 78 personas a la conferencia, entonces necesitaremos 1 grupo de 50 personas, 2 grupos de 10 personas y 8 grupos de 1 persona.

```
import java.util.Scanner;

public class GroupPeople
{
    public static void main(String[] args)
    {
        int people, groups50, groups10, groups1;

        Scanner sc = new Scanner(System.in);

         System.out.println("Enter how many people are going to attend the
conference:");
        people = sc.nextInt();

        groups50 = groups10 = groups1 = 0;

        /*
         * We can just substract 50 or 10 people to the total amount
         * as long as we can, and then increment the number of
         * respective groups

        while(people >= 50)
        {
            people -= 50;
            groups50++;
        }

        while (people >= 10)
        {
            people -= 10;
            groups10++;
        }
        */


        /*
         * And we can just divide the total amount into 50, and then
         * into 10 to determine how many groups can be made of each
         * category
         */
```

```
        groups50 = people / 50;
        people = people % 50;

        groups10 = people / 10;
        groups1 = people % 10;

        System.out.println(groups50 + " groups of 50");
        System.out.println(groups10 + " groups of 10");
        System.out.println(groups1 + " groups of 1");
    }
}
```

> **2.2.2.2.** Crea un programa llamado *SumDigits* que pida al usuario que introduzca números enteros hasta que se introduzca un 0. El programa deberá sumar todos los números introducidos y mostrar el resultado final, así como indicar el número de dígitos que tiene. Por ejemplo, si el usuario introduce 12,20,60,33,99 y 0, entonces el programa deberá mostrar: "El resultado es 224 y tiene 3 dígitos".

```
import java.util.Scanner;

public class SumDigits
{
    public static void main(String[] args)
    {
        int number, sum = 0, sumCopy, digitCount;

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter numbers. Enter 0 to finish:");

        do
        {
            number = sc.nextInt();
            sum += number;
        } while(number != 0);

        /*
         * In order to determine how many digits the sum has, we can
         * divide by 10 iteratively, and then check how many times we
         * have divided...
         *
        digitCount = 0;
        sumCopy = sum;

        while (sumCopy > 0)
        {
            digitCount++;
            sumCopy /= 10;
        }
        */

        /*
         * ... or we can convert the sum into a string and check its
         * length
         */

        digitCount = ("" + sum).length();


           System.out.println("The result is " + sum + " and it has " +
digitCount + " digits");
```

```
        }
    }
```