Entornos de desarrollo
Bloque 2 Tema 4: Funciones y manejo de excepciones

Soluciones

> **Ejercicios propuestos**
>
> **2.4.1.1.** Crea un programa llamado *Palindrome* con una función llamada *isPalindrome*. Esta función recibirá una cadena como parámetro y devolverá un booleano indicando si dicha cadena es palíndromo (o sea, una cadena que se lee igual de adelante hacia atrás que de atrás hacia adelante, sin tener en cuenta mayúsculas ni minúsculas). Prueba esta función desde la función *main* con los textos *Hannah*, *Too hot to hoot* and *Java is the best language* (este último no es palíndromo).

```
public class Palindrome
{
    public static boolean isPalindrome(String text)
    {
        int i, j;
        boolean result;

        text = text.toUpperCase();
        text = text.replace(" ", "");

        i = 0;
        j = text.length() - 1;
        result = true;

        while(i <= j && result)
        {
            if (text.charAt(i) != text.charAt(j))
                result = false;
            i++;
            j--;
        }
        return result;
    }

    public static void main(String[] args)
    {
        System.out.println(isPalindrome("Hannah"));
        System.out.println(isPalindrome("Too hot to hoot"));
        System.out.println(isPalindrome("Java is the best language"));
    }
}
```

2.4.1.2 Crea un programa llamado *CountOcurrences* con una función llamada *countString*. Esta función recibirá dos cadenas *a* y *b*, y un entero *n* como parámetros, y devolverá un booleano indicando si la cadena *b* está contenida al menos *n* veces en la cadena *a*. Pruébalo desde la función main con la cadena a= `This string is just a sample string`, la subcadena b= `string` y el número n= 2 (debería devolver `true`).

```
public class CountOccurrences
{
    public static boolean countString(String a, String b, int n)
    {
        int pos, times = 0;

        do
        {
            pos = a.indexOf(b);
            if (pos >= 0)
            {
                times++;
                a = a.substring(pos + b.length());
            }
        }
        while(times < n && pos >= 0);

        return times == n;
    }

    public static void main(String[] args)
    {
             System.out.println(countString("This string is just a sample
string", "string", 2));
    }
}
```

2.4.2.1. Crea un programa llamado *CalculateDensity* que pida al usuario un peso (en gramos) y un volumen (en litros) y muestre la densidad. La densidad se calcula dividiendo peso / volumen. El programa debe capturar las excepciones que se puedan producir : `NumberFormatException` y `ArithmeticException` donde se puedan generar. Solo se puede usar el método `Scanner.nextLine` para leer los datos en este ejercicio.

```
import java.util.Scanner;

public class CalculateDensity
{
    public static void main(String[] args)
    {
        int weight, volume, density;

        Scanner sc = new Scanner(System.in);

        try
        {
            System.out.println("Enter weight:");
            weight = Integer.parseInt(sc.nextLine());

            System.out.println("Enter volume:");
            volume = Integer.parseInt(sc.nextLine());

            density = weight / volume;
            System.out.println("Density = " + density);

        } catch (NumberFormatException e) {
            System.err.println("Wrong number:" + e.getMessage());
        } catch (ArithmeticException e) {
            System.err.println("Error dividing by zero");
        }
    }
}
```

2.4.2.2. Crea un programa llamado *WaitApp* con una función llamada *waitSeconds* que recibirá un número de segundos (entero) como parámetro. Esta función deberá llamar al método `Thread.sleep` para pausar el programa el número de segundos pasados como parámetro (este método funciona con milisegundos, por lo que se deberá convertir los segundos recibidos a milisegundos). Como el método `sleep` puede producir una excepción del tipo `InterruptedException`, se necesitará manejar. En este caso se deberá lanzar la excepción desde la función *waitSeconds* y capturarla en la función *main* que llamará a la función *waitSeconds* con los segundos que habrá recibido como parámetro (en `String[] args`) Después de esperar el número de segundos especificados, el programa mostrará un mensaje de finalización antes de salir.

```
import java.util.Scanner;

public class WaitApp
{
    public static void waitSeconds(int seconds) throws InterruptedException
    {
        Thread.sleep(seconds * 1000);
    }

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

        System.out.println("Enter seconds:");
        seconds = sc.nextInt();

        try
        {
            waitSeconds(seconds);
        } catch (InterruptedException e) {
            System.err.println("Error: " + e.getMessage());
        }

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