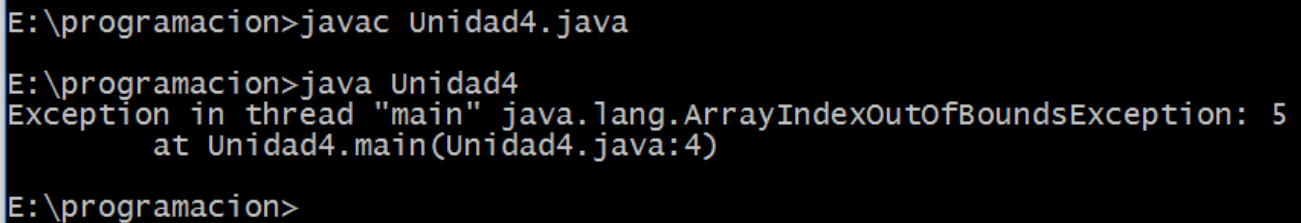


## Ejercicio U5\_B9A\_E1:

```
class App{
    public static void main(String[] args){
        int[] x= {0,1,2,3,4};
        x[5]=5;
        System.out.println("El programa se recupera de la excepción y continua");
    }
}
```

ArrayIndexOutOfBoundsException es de tipo no verificado (unchecked) y comprobamos que la excepción salta en tiempo de ejecución, no en tiempo de compilación. El println() no se llega a ejecutar ya que el programa rompe antes.



```
E:\programacion>javac Unidad4.java
E:\programacion>java Unidad4
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
    at Unidad4.main(Unidad4.java:4)
E:\programacion>
```

```
class App{
    public static void main(String[] args){
        int[] x= {0,1,2,3,4};
        try{
            x[5]=5;
        }catch(ArrayIndexOutOfBoundsException e){
            System.out.println("No es posible meter nada en x[5]");
        }
        System.out.println("El programa se recupera de la excepción y continua");
    }
}
```

Recuerda que el código anterior no es de buena calidad ya que todo programador debe controlar los límites de un array expresamente y no recurrir al sistema de excepciones cuando ya sabemos de antemano lo que puede pasar.

## Ejercicio U5\_B9A\_E2:

### SOLUCIÓN CON InputMismatch

Hasta ahora siempre que usamos la clase Scanner supusimos que el usuario metía los datos correctamente, nunca nos ocurrían excepciones InputMismatch. Observa ahora el nextLine() del catch, es para limpiar el contenido del buffer de la variable teclado, si comentas el nextLine() hay bucle infinito ya que nextInt() genera una excepción y como no encontró token no avanza en la lectura del buffer de Scanner, de forma que cuando volvamos a hacer un nextInt() volvemos a leer lo mismo.

```
import java.util.InputMismatchException;
import java.util.Scanner;
class App{
    public static void main(String[] args) {
        Scanner teclado= new Scanner(System.in);
        int n=0;

        //obtenemos un entero correcto
        boolean numeroEnteroCorrecto=false;
```

```

do{
    try{
        System.out.print("Teclea número entero para calcular factorial: ");
        n=teclado.nextInt();

        numeroEnteroCorrecto=true;
    }catch(InputMismatchException e){
        teclado.nextLine();//sin esto se vuelve a leer el mismo token
        System.out.println("La entrada no es un número entero correcto");

    }

}while(!numeroEnteroCorrecto);

//calculamos el factorial
int factorial=n;
String salida=n+"! = "+n;
n--;
while(n>=1){
    salida+="*"+n;
    factorial*=n;
    n--;
}
salida+=" = "+factorial;
System.out.println(salida);
}
}

```

## SOLUCIÓN CON NumberFormatException

```

class App{
    public static void main(String[] args) {
        Scanner teclado= new Scanner(System.in);

        int n=0; //Numero entero introducido por teclado.

        //obtenemos un número entero correcto
        boolean numeroEnteroCorrecto=false;
        do{
            try{
                System.out.print("Teclea número entero para calcular factorial: ");
                n=Integer.parseInt(teclado.nextLine());
                numeroEnteroCorrecto=true;
            }catch(NumberFormatException e){
                System.out.println("La entrada no es un número entero correcto");
            }
        }while(!numeroEnteroCorrecto);

        //calculamos factorial
        int factorial=n;
        String salida=n+"! = "+n;
        for(int i=n-1;i>0;i--){
            salida+="*"+i;
            factorial*=i;
        }
        salida+=" = "+factorial;
        System.out.println(salida);
    }
}

```

En general, el mecanismo apropiado para depurar los datos de entrada es leer dichos datos como un String y luego analizar dicho String con los métodos de la clase String y expresiones regulares (lo veremos más adelante). De todas formas siempre puede haber un contexto apropiado para trabajar como en los dos ejemplos anteriores