SESIÓN 7: Librerías básicas: String y StringBuffer, Envoltorios, BigInteger y BigDecimal. Excepciones

Objetivos

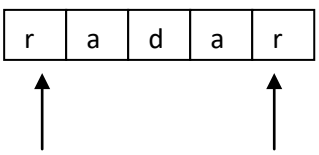
- Usar la clase *String* para tratar cadenas de caracteres inmutables.
- Usar la clase *StringBuffer* para tratar cadenas de caracteres modificables.
- Saber escribir una estructura *try – catch* para el manejo de excepciones.
- Saber utilizar las clases envoltorios (*Wrappers).*
- Saber utilizar las clases BigInteger y BigDecimal.

Nota importante: Siga el esquema de nombrado de paquetes que se indicó en la sesión 01 es decir: **org.ip.sesion07**. En ese paquete se crearán todos los programas que se proponen en la sesión dándoles un nombre al programa y que se indica en cada ejercicio entre **paréntesis y en negrita**.
Al final de la sesión, el alumno deberá cargar el trabajo realizado a su repositorio personal indicando la clave correspondiente a la sesión.

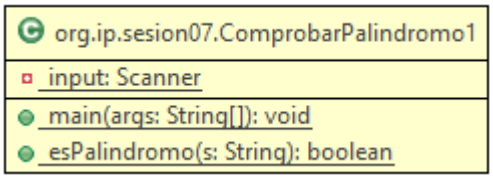
Ejercicios propuestos

1. Un palíndromo es una palabra o frase que se lee igual de izquierda a derecha, que de derecha a izquierda. Por ejemplo: seres, radar. Implemente un programa **(ComprobarPalindromo1)** que lea de teclado en el método *main* un string (utiliza para ello la clase *Scanner)* y compruebe a partir del método *esPalindromo* si dicha cadena es o no palíndromo.

Una posible solución para el método *esPalindromo* es ir comparando el primer carácter de la cadena con el último para ver si coinciden. El segundo carácter con el penúltimo carácter y así sucesivamente. Para implementar esta idea, usa dos variables, *inicio* y *fin* para indicar las posiciones de los caracteres por el principio y el final del string *s*. Por tanto, *inicio* deberás inicializarla a 0 (posición del primer carácter) y *fin* a *s.length()*-1. Si los caracteres en las posiciones *inicio* y *fin* coinciden compararemos los siguientes caracteres de *s* (incrementando *inicio* y decrementando *fin*). El proceso continuará hasta que *(inicio >= fin)* o encontramos caracteres distintos.



inicio = 0 *fin* = *s.length()*-1



Ejemplo de ejecución:

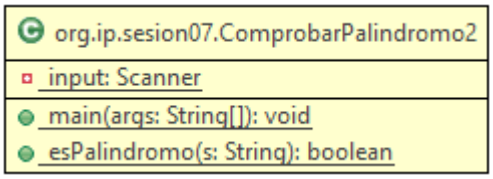
```
Introduzca una cadena radar
radar es un palindromo
```

Implemente otra solución **(ComprobarPalindromo2)** que permita comprobar si una frase es palíndroma. Por ejemplo, *A cavar a Caravaca* es palíndroma.

No se tendrán en cuenta los caracteres blancos y no se distinguirán mayúsculas de minúsculas. Para ello, sobre el algoritmo anterior, incluye en el cuerpo de la estructura repetitiva (**while (inicio < fin)**) otros dos bucles, uno que permita saltar blancos por la izquierda y otro que lo haga por la derecha.

Ejemplo de ejecución:

```
Introduzca una cadena A CAVAR A Caravaca
A CAVAR A Caravaca es un palindromo
```

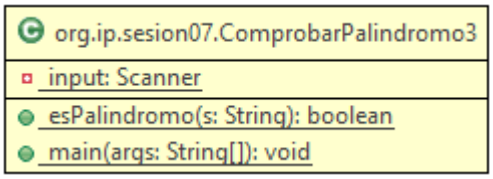


2. Repite el ejercicio anterior dando una tercera solución. **(ComprobarPalindromo3)**. El método *main* será el mismo que la primera solución (es decir leeremos de teclado un string) y el nuevo método *esPalindromo* utilizará:

- ✓ La clase *StringBuffer* y el método *reverse* para invertir la cadena.
- ✓ El método *substring* de la clase *StringBuffer* para extraer una cadena (devuelve un *String)*.
- ✓ El método *equals* para comparar objetos de la clase *String.*

Ejemplo de ejecución:

```
Introduzca una cadena radar
La cadena radar es un palindromo
```



Trabajo autónomo

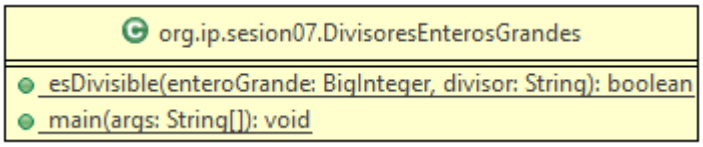
3. Implemente un programa (**DivisoresEnterosGrandes**) que muestre los 5 primeros números divisibles entre 5 y 6 mayores que Long.MAX_VALUE (9223372036854775807). El método de objeto *remainder* de la clase **BigInteger** puede ayudarte. Dicho método es el equivalente al operador **%** utilizado con datos de tipo primitivo. Consúltalo.

   Ejemplo de ejecución:

```
Introduce el primer divisor => 5
Introduce el segundo divisor => 6

Los 5 primeros números mayores de 9223372036854775807 divisibles entre 5 y 6 son:

9223372036854775830
9223372036854775860
9223372036854775890
9223372036854775920
9223372036854775950
```



4. Ejecuta el programa **(EntradaEnterosSinExcepcion)** que se proporciona con la sesión, introduciendo como valores de entrada los caracteres: **u v**, en lugar de dos valores enteros.

```
1  package org.ip.sesion07;
2
3  import java.util.Scanner;
4
5  public class EntradaEnterosSinExcepcion {
6
7      /**
8       * @param args
9       */
10     public static void main(String[] args) {
11         Scanner entrada = new Scanner(System.in);
12
13         // Introducir dos enteros
14         System.out.print("Introduzca dos valores enteros: ");
15         int number1 = entrada.nextInt();
16         int number2 = entrada.nextInt();
17
18         System.out.println("La suma es  " + (number1 + number2));
19     }
20 }
```

Comprobarás que la salida genera la excepción:

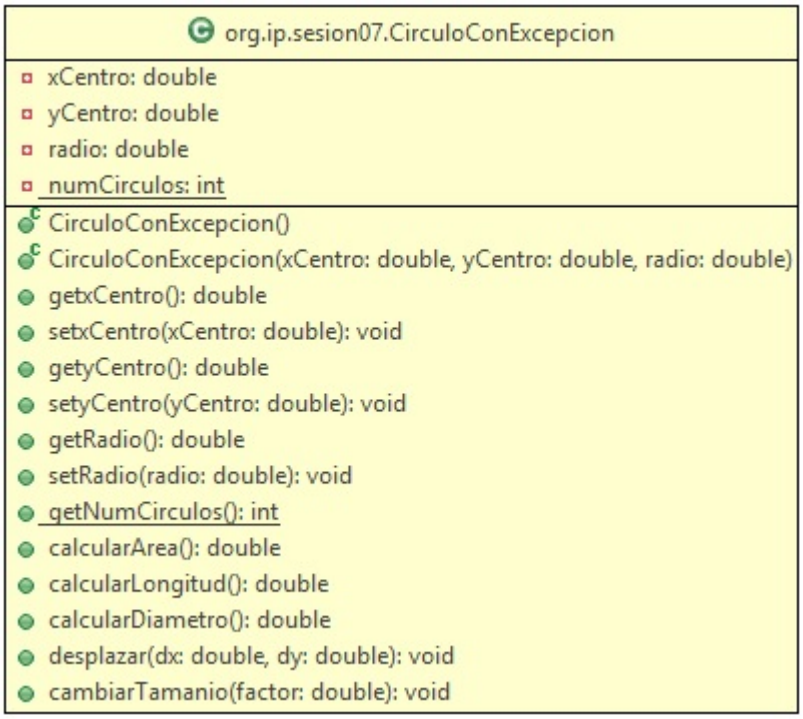
```
Introduzca dos valores enteros: u v
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:864)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nextInt(Scanner.java:2117)
        at java.util.Scanner.nextInt(Scanner.java:2076)
        at org.ip.sesion07.EntradaEnterosSinExcepcion.main(EntradaEnterosSinExcepcion.java:15)
```

Reescriba el programa anterior **(EntradaEnterosConExcepcion)** utilizando la estructura **try-catch** para tratar y manejar la excepción y así impedir que se interrumpa la ejecución ante una entrada de datos errónea. El programa estará pidiendo la entrada de los dos números mientras dicha entrada no sea correcta.

Ejemplo de ejecución:

```
Introduzca dos valores enteros: u v
Incorrecto, introduzca dos valores enteros: 4 j
Incorrecto, introduzca dos valores enteros: k 8
Incorrecto, introduzca dos valores enteros: 4 9
La suma es  13
```

5. Implemente una clase Java para la creación y uso de círculos con gestión excepciones **(CirculoConExcepcion)**. La gestión de excepciones (`IllegalArgumentException`) debe estar en los métodos setter *setRadio*, que a su vez debe utilizarse en uno de los constructores (el *radio* no debe ser negativo) y en *cambiarTamanio* que es donde el *radio* se ve modificado y el valor del factor tampoco debe ser negativo. Finalmente, implemente también un test (**TestCirculoConExcepcion**) para dicha clase en la que se compruebe la gestión de excepciones, tal y como se muestra a continuación.



Ejemplo de ejecución:

Problems | Javadoc | Declaration | Search | Console | History

```
<terminated> TestCirculoConExcepcion [Java Application] C:\ECLIPSES\GIPP2015EclipseLuna\JDKS\jdk1.8.0_31\bin\javaw.exe (27/11/2015 0:32:33)
java.lang.IllegalArgumentException: El radio no puede ser negativo
Number of objects created: 1
java.lang.IllegalArgumentException: El factor no puede ser negativo
Number of objects created: 2
```