

**17**

**DESARROLLO WEB**

v1.00 Abril 2023

**TEST PROJECT**

Cofinanciado por:

[1. Instrucciones generales 1](#_Toc507068089)

[2. Prueba 2](#_Toc507068090)

[3. Esquema de puntuaciones 6](#_Toc507068091)

# Instrucciones generales

**Hora límite de entrega: 13:30 horas**

Cada uno de los alumnos dispone de una máquina virtual con el sistema operativo Windows 10 y el entorno de desarrollo integrado NetBeans.

Esta prueba de programación tiene una puntuación, en su conjunto, de **80 puntos fijos** y un 25% extra en cada una de las preguntas. Está compuesta por 3 ejercicios distintos denominados Ejercicio 1, Ejercicio 2 y Ejercicio 3. Por lo tanto, las puntuaciones serían las siguientes:

* *Ejercicio 1. 15 puntos + 3,75 extra*
* *Ejercicio 2. 25 puntos + 6,25 extra*
* *Ejercicio 3. 40 puntos + 10 extra*

Esos **3 ejercicios** deberán desarrollarse en el lenguaje de programación Java utilizando el citado Entorno de Desarrollo Integrado **NetBeans**, en su versión 8.2.

Las propuestas serán enviadas a una tarea del curso, de la plataforma Moodle, donde se encuentran matriculados los alumnos participantes.

# Prueba

**1.- Ejercicio 1 de la prueba de programación**

**Introducción**

Palíndromo

El juego es parte fundamental del ser humano, algo inherente a las personas. Se juega en todas las culturas, en todos los rincones del mundo. Y desde tiempos inmemoriales, desde que existe la lengua y la escritura, se han practicado juegos con las palabras.

Uno de los juegos lingüísticos más famosos y antiguos que existen es el de los palíndromos, aquellas palabras o frases que se leen igual de izquierda a derecha que de derecha a izquierda. ¿Un ejemplo? **Aérea**: da igual que se lea empezando por delante que comenzando por detrás, el resultado es siempre el mismo. Y lo mismo ocurre con otros muchos vocablos como ojo, radar, orejero, reconocer…

La palabra palíndromo es de hecho de origen griego y viene de la construcción *"palin dromein"*, es decir, volver atrás, recorrer a la inversa, porque se trata precisamente de eso, de palabras o frases simétricas que se pueden transitar en ambos sentidos. Los palíndromos numéricos también existen, aunque se conocen como *capicúa*: 1331, 8008, 353.

**También frases**

Pero, para complicar el juego, también una frase puede ser un palíndromo. El ejemplo clásico, el que recoge el diccionario de la Real Academia de la Lengua Española es: *"Dábale arroz a la zorra el abad".* Pero hay muchas, muchísimas más: *Somos o no somos, Isaac no ronca así, Sé verlas al revés, Amó la paloma, Anita lava la tina, Luz azul, Yo hago yoga hoy, Ana lava lana…*

**Enunciado del ejercicio 1 (15 puntos)**

* **Apartado 1**

Realiza un método iterativo en Java, que reciba como parámetro un objeto de tipo *String* y devuelva, un valor booleano, para indicar si se trata de un palíndromo o no.

* **Apartado 2**

Realiza un método recursivo en Java, que reciba como parámetro un objeto de tipo *String* y devuelva, un valor booleano, para indicar si se trata de un palíndromo o no.

Posteriormente, debes usar ambos métodos en una clase que contenga el método **main**.

**NOTA IMPORTANTE:** no está permitido usar ninguna clase o métodos además de los incluidos en la clase **String** de la API de Java (por ejemplo, no está permitido usar la clase *StringBuilder* con el método reverse() ).

**2.- Ejercicio 2 de la prueba de programación**

Subset sum problem (SSP)

El **subset sum problem (SSP)** es un problema de decisión en la informática. En su formulación más general, hay un multiset S de integers y un target-sum K, y la cuestión es decidir si cualquier subconjunto de los enteros suma precisamente K.

El problema a plantear se puede abreviar con la siguiente frase:

*“Dado un conjunto de enteros positivos y un entero k, verifique si hay algún subconjunto no vacío que sume k”.*

**Enunciado del ejercicio 2 (25 puntos)**

Realice un ejercicio en Java que reciba por la entrada estándar una secuencia de 500 números positivos (el cero no está incluido, pero puede haber números repetidos).

A continuación, el programa solicitará el valor de la variable K.

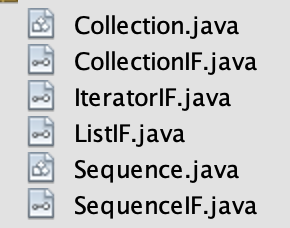
El programa debe mostrar por pantalla, todos los subconjuntos de 1, 2 y 3 elementos (de la secuencia de 500 números) que suman el valor K.

**3.- Ejercicio 3 de la prueba de programación**

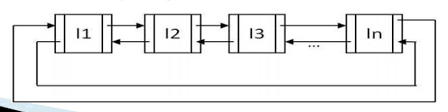
**Enunciado del ejercicio 3 (40 puntos)**

Lista doblemente enlazada circular y ordenada (DECO)

Se dispone de una carpeta que incluye clases e interfaces. Las interfaces están definidas en ficheros cuyos nombres tienen el sufijo IF.



Usando los ficheros .*java* incluidos, se debe crear una nueva estructura de datos, que represente a una lista doblemente enlazada circular y ordenada.



Esta estructura será definida en un fichero llamado **ListaDECO.java** y debe cumplir lo siguiente:

**Atributos**

* Un atributo para llevar la cuenta de cuántos elementos hay incluidos en la estructura.
* Un atributo para “apuntar” a la cabeza o primer elemento de la lista.
* Un atributo para “apuntar” a la cola o último elemento de la lista.

**Métodos**

* Constructor por defecto
* Constructor de copia
* Un método para obtener el elemento almacenado en una posición dada.
* Un método para buscar la posición que ocupa un elemento E.
* Un método para insertar un nuevo elemento en la posición que corresponda, para que la lista esté siempre ordenada.

Todos los métodos de la **ListaDECO** deben implementar la genericidad, esto es, que se podrán almacenar objetos de cualquier tipo en el campo *value* de un nodo.

*Está permitido hacer las modificaciones que sean necesarias en cualquier fichero, pero se debe indicar con un comentario.*

# Esquema de puntuaciones

La corrección de los ejercicios se realizará probando 4 casos de prueba, para cada uno de los programas.

Para el caso especial en el que dos o más soluciones, cumplan con la salida esperada en los 4 casos de prueba, se le asignará una nota extra del 25% de la nota total del ejercicio, a aquella solución que realice las operaciones necesarias, en el menor tiempo de ejecución.

Vamos a verlo mejor con un ejemplo:

* Tenemos dos alumnos, llamados Juan y Pepe, cuya solución al ejercicio 1 es 100% correcta. Por lo tanto, Juan y Pepe suman 15 puntos en ese ejercicio.

* A continuación, al comprobar el tiempo de ejecución de sus soluciones, se observa que la solución de Juan ha tardado 120 milisegundos y la de Pepe 150 milisegundos. Ante tal situación, Juan obtiene 3,75 puntos extra.

**NOTA IMPORTANTE**

Para optar al 25% extra de puntuación de cada uno de los ejercicios, el programa debe obtener la salida esperada en los 4 casos de prueba.

En caso de que solo un alumno haya resuelto 100% correctamente un ejercicio, este alumno obtendrá el 25% extra, independientemente del tiempo de ejecución de su programa.

Si la ejecución de cualquier programa terminase bruscamente por una **excepción** no controlada, la nota de ese ejercicio será de 0 puntos, independientemente de que, para alguno de los casos de prueba, la salida obtenida coincidiese con la salida esperada.

**TABLA DE CALIFICACIONES**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ítem** | **%** | **Ejercicio 1** | **Ejercicio 2** | **Ejercicio 3** | **Total** |
| **Entrada / Salida** | 10 | 2 | 2 | 4 | 8 |
| **Estructuras de control** | 25 | 4 | 8 | 8 | 20 |
| **Excepciones** | 10 | 3 | 2 | 3 | 8 |
| **Estructuras de datos** | 25 | 0 | 7 | 13 | 20 |
| **Programación Orientada a Objetos** | 30 | 6 | 6 | 12 | 24 |
| **Total** | **100** | **15** | **25** | **40** | **80** |