Hoja 14

Ejercicio 1:

1. **Se almacenarán los candidatos a jefe de la sección 1 en una lista doblemente enlazada llamada listado.**
2. Los candidatos serán seis:
   “ana", 53 "bea", 57 "oliver", 59 “leo", 63 "lia", 55 “anais", 51
3. Añado al listado los 4 primeros.
4. Me doy cuenta de que el quinto candidato iría mejor en la segunda posición.
5. El primer candidato llama y deja su puesto al sexto candidato, con lo cual este será sustituido
6. El que inscribí como tercer candidato cand3 abandona.
7. Me llaman de personal y me piden reducir el número de entrevistas a tres. He de eliminar el que ahora ocupa el último lugar (ya no recuerdo la longitud de mi listado)
8. ¿Qué lugar ocupa cand5?

9. Quiero ver el contenido del listado. (No valdría aplicar toString directamente, pero sí sobreescrito).
   Además, hay tres métodos diferentes para recorrer el Arraylist
   - Bucle for que ya conocemos
   - Bucle for-each
   - Iterador

10. Hacemos una sublista desde la posición 2 (índice 1) hasta el final de la lista. Y vemos el contenido de esta nueva lista con un iterador.

11. **Se almacenarán los candidatos a jefe de la sección 2 en una lista de tipo ArrayList listado2.**
    "tomás", 40 "teresa", 43

12. Se añaden estos candidatos al listado2 y vemos el contenido de los objetos, esta vez creando para ello un iterador.

13. Uno las dos colecciones indicando la posición de listado donde quiero introducir el listado2, por ejemplo, al principio.

14. A partir de aquí listado lleva a todos los candidatos. Lo vemos con el bucle for-each. Y podrá ser tratado como LinkedList.

Ponemos a prueba métodos de colas y pilas:

15. Consultar el primer elemento de la cola
16. Consultar y eliminar el primer elemento de la cola
17. Volver a consultar el primer elemento de la cola
18. Añadir un elemento a la cola.
19. Ver el listado.
20. Consultar el primer elemento de la pila
21. Consultar y eliminar el primer elemento de la pila
22. Volver a consultar el primer elemento de la pila
23. Añadir un elemento a la pila.
24. Ver el listado.

Ejercicio 2: Realiza una aplicación que pida diferentes nombres con el fin de visualizarlos en el orden que se insertaron y en orden contrario. Para saber si el usuario quiere insertar más nombres se pregunta
¿Quiere insertar otro nombre? s/n

Ejercicio 3: Vamos a guardar cadenas en una colección con el fin de borrar posteriormente la que empiecen por b
"hola"
"buenos días"
"buenas tardes"
"hasta luego"

Ejercicio 4: Escribe un programa que cargue dos colecciones distintas con datos de diferentes candidatos. Inserta datos, alguno en común en las dos colecciones. Realizar las siguientes operaciones:

Añade a la primera colección todos los candidatos que tiene la segunda
Borra de la primera todos los candidatos de la otra
Borra de la segunda colección todos los candidatos que estén en la otra
Borra todos los candidatos de la primera colección

Ejercicio 5: Escribe un programa que pida los datos de una serie de candidatos (nombre y edad). El programa debe estar pidiendo datos de candidatos hasta que se inserte como edad un dato no numérico.
Posteriormente, una vez almacenado lo anterior, se le pedirá al usuario que inserte una edad con el fin de borrar todos los elementos que tengan una edad inferior a la indicada.

Ejercicio 6: En el ejercicio Candidatos vamos a sobreescribir el método equals en la clase Candidato. De esta forma haremos que compare valores de los atributos en lugar de direcciones de memoria.

Ejercicio 7:
Idea un menú para trabajar sobre una colección de candidatos.