Ejercicios prácticos

Unidades: 9

Guía

- Se plantean diferentes ejercicios para realizarlos en java
- Se recomienda utilizar un IDE como Eclipse o Netbeans.
- Cada ejercicio está pensado a utilizar una colección concreta.
- Puede haber más de una solución en un ejercicio, buscar la solución óptima, aquella que requiera menos líneas de código y no se abuse de estructuras de control.

Ejemplo

Guardar el abecedario en una lista.

```
import java.util.ArrayList;
import java.util.List;

public class Ejemplo {

    public static void main(String[] args) {
        List<Character> alfabeto = new ArrayList<>();

        for(char ch = 'A'; ch <= 'Z'; ++ch) {
            alfabeto.add(ch);
        }

        System.out.println(alfabeto);
    }

}
```

Ejercicio 1.

Crea un programa que permita al usuario introducir nombres de personas y guardarlos en una colección. Se quiere almacenar solo los nombres únicos y no repetidos (No hay que realizar ninguna lógica para verificar si se introducen nombres repetidos). Una vez que se han introducido todos los nombres, el programa debe imprimir todos los nombres únicos.

Ejercicio 2.

Crea un programa que gestione una lista de tareas pendientes. El usuario debe poder agregar tareas, eliminar tareas y ver todas las tareas. Piensa adecuadamente que colección corresponde según la funcionalidad descrita.

Ejercicio 3.

Crea un simulador de una cola de un banco. Los clientes deben ser añadidos a la cola y procesados en orden (FIFO, primero en entrar primero en salir). Asegúrate de que la cola funciona correctamente cuando llegan nuevos clientes y cuando un cliente es atendido, para ello, haz un menú para poder agregar nuevos clientes a la cola, atender a clientes y mostrar los clientes que actualmente están esperando.

Ejercicio 4.

Implementa un sencillo diccionario español-inglés. El usuario puede introducir palabras en español y el programa traducirá al inglés. Piensa adecuadamente como almacenar el diccionario para que sea lo más eficiente posible.

Ejercicio 5.

Desarrollar un programa para solicitar números enteros, guardarlos en una lista hasta que se decida finalizar. Una vez finalizado, ordenar la lista de números de menor a mayor utilizando el algoritmo de ordenación burbuja y mostrar el resultado por pantalla.

El algoritmo de ordenación burbuja tiene el siguiente pseudocódigo:

```
procedimiento ordenacionBurbuja(lista)
   para i desde 0 hasta tamaño de lista - 1
       para j desde 0 hasta tamaño de lista - i - 1
           // Si el elemento actual es mayor que el siguiente
           si lista[j] > lista[j + 1] entonces
               // Intercambiar los elementos
               temp = lista[j]
               lista[j] = lista[j + 1]
               lista[j + 1] = temp
           fin si
       fin para
   fin para
fin procedimiento
```

Ejercicio 6.

Implementa un programa que dado dos conjuntos de números enteros, calcule la intersección y la unión de estos. Cada conjunto puede tener un número indeterminado de elementos y no puede haber repetidos. El programa debe imprimir ambos conjuntos, la intersección y la unión.

La intersección de dos conjuntos es un nuevo conjunto que contiene todos los elementos que son comunes a ambos conjuntos originales

La unión de dos conjuntos es un nuevo conjunto que contiene todos los elementos que están en cualquiera de los dos conjuntos originales

Por ejemplo, si hay dos conjuntos:

- Conjunto A: {1, 2, 3, 4, 5}
- Conjunto B: {4, 5, 6, 7, 8, 9}

La intersección de A y B es: {4, 5} y la unión de A y B es: {1, 2, 3, 4, 5, 6, 7, 8, 9}

Ejercicio 7.

Implementa un programa que simule la gestión de procesos en un sistema operativo utilizando una cola. Los procesos tienen un ID único y un tiempo de ejecución en milisegundos. El programa debe permitir añadir procesos, ejecutar los procesos en orden de llegada y mostrar los procesos que están en cola. No es posible ejecutar el siguiente proceso, si el anterior aún no ha acabado su tiempo de ejecución.

Ejercicio 8.

Escribe un programa que cuente el número de apariciones de cada carácter en una cadena de texto. La cadena de texto será introducida por teclado y finalmente se debe de mostrar por cada carácter la cantidad que aparece en la cadena de texto.

Ejercicio 9.

Crea un programa que maneje un conjunto de estudiantes y sus notas. Cada estudiante tiene un nombre y una nota media. El programa debe permitir al usuario añadir estudiantes sin que existan repetidos y ver todos los estudiantes ordenados ascendentemente por su nombre alfabéticamente. Además, también se debe poder buscar estudiantes por su nombre.

Ejercicio 10.

Implementa un sistema de votación para una elección política. El programa debe contemplar los siguientes requisitos:

- Los candidatos disponen de su nombre y el nombre del partido político que representa. Se debe de inicializar los candidatos al inicio del sistema de votación hasta un máximo de 5 y no puede existir candidatos repetidos.
- Los votantes son añadidos a una cola. Cada votante se identifica por un ID único y vota por un candidato. Aún que el voto debe ser anónimo, es necesario que se quede guardado el voto que ha realizado. Se debe poder ingresar nuevos votantes y ejercer su voto según el orden de la cola.
- Los votos de cada candidato se almacenan en una lista para facilitar la cuenta final. Un candidato solo puede ejercer un único voto.
- Al finalizar la votación, el programa debe mostrar el candidato ganador y el recuento total de votos. En el caso de empate en votos el ganador será el primer candidato que se le hizo el recuento.