EJERCICIOS DE APRENDIZAJE

En este módulo vamos a continuar modelando los objetos con el lenguaje de programación Java, pero ahora vamos a utilizar las colecciones para poder manejarlas de manera más sencilla y ordenada.

VIDEOS: Te sugerimos ver los videos relacionados con este tema, antes de empezar los ejercicios, los podrás encontrar en tu aula virtual o en nuestro canal de YouTube.

1. Diseñar un programa que lea y guarde razas de perros en un ArrayList de tipo String. El programa pedirá una raza de perro en un bucle, el mismo se guardará en la lista y después se le preguntará al usuario si quiere guardar otro perro o si quiere salir. Si decide salir, se mostrará todos los perros guardados en el ArrayList.

2. Continuando el ejercicio anterior, después de mostrar los perros, al usuario se le pedirá un perro y se recorrerá la lista con un Iterator, se buscará el perro en la lista. Si el perro está en la lista, se eliminará el perro que ingresó el usuario y se mostrará la lista ordenada. Si el perro no se encuentra en la lista, se le informará al usuario y se mostrará la lista ordenada.

3. Crear una clase llamada Alumno que mantenga información sobre las notas de distintos alumnos. La clase Alumno tendrá como atributos, su nombre y una lista de tipo Integer con sus 3 notas.

   En el servicio de Alumno deberemos tener un bucle que crea un objeto Alumno. Se pide toda la información al usuario y ese Alumno se guarda en una lista de tipo Alumno y se le pregunta al usuario si quiere crear otro Alumno o no.

   Después de ese bucle tendremos el siguiente método en el servicio de Alumno:

   Método notaFinal(): El usuario ingresa el nombre del alumno que quiere calcular su nota final y se lo busca en la lista de Alumnos. Si está en la lista, se llama al método. Dentro del método se usará la lista notas para calcular el promedio final de alumno. Siendo este promedio final, devuelto por el método y mostrado en el main.

Pueden encontrar un ejemplo de Colección como Atributo en tu Aula Virtual.

4. Un cine necesita implementar un sistema en el que se puedan cargar peliculas. Para esto, tendremos una clase Pelicula con el título, director y duración de la película (en horas). Implemente las clases y métodos necesarios para esta situación, teniendo en cuenta lo que se pide a continuación:

   En el servicio deberemos tener un bucle que crea un objeto Pelicula pidiéndole al usuario todos sus datos y guardándolos en el objeto Pelicula.

   Después, esa Pelicula se guarda una lista de Peliculas y se le pregunta al usuario si quiere crear otra Pelicula o no.

   Después de ese bucle realizaremos las siguientes acciones:

- Mostrar en pantalla todas las películas.
- Mostrar en pantalla todas las películas con una duración mayor a 1 hora.
- Ordenar las películas de acuerdo a su duración (de mayor a menor) y mostrarlo en pantalla.
- Ordenar las películas de acuerdo a su duración (de menor a mayor) y mostrarlo en pantalla.
- Ordenar las películas por título, alfabéticamente y mostrarlo en pantalla.
- Ordenar las películas por director, alfabéticamente y mostrarlo en pantalla.

5. Se requiere un programa que lea y guarde países, y para evitar que se ingresen repetidos usaremos un conjunto. El programa pedirá un país en un bucle, se guardará el país en el conjunto y después se le preguntará al usuario si quiere guardar otro país o si quiere salir, si decide salir, se mostrará todos los países guardados en el conjunto. (Recordemos hacer los servicios en la clase correspondiente)

   Después deberemos mostrar el conjunto ordenado alfabéticamente: para esto recordar cómo se ordena un conjunto.

   Por último, al usuario se le pedirá un país y se recorrerá el conjunto con un Iterator, se buscará el país en el conjunto y si está en el conjunto se eliminará el país que ingresó el usuario y se mostrará el conjunto. Si el país no se encuentra en el conjunto se le informará al usuario.

6. Se necesita una aplicación para una tienda en la cual queremos almacenar los distintos productos que venderemos y el precio que tendrán. Además, se necesita que la aplicación cuente con las funciones básicas.

   Estas las realizaremos en el servicio. Como, introducir un elemento, modificar su precio, eliminar un producto y mostrar los productos que tenemos con su precio (Utilizar Hashmap). El HashMap tendrá de llave el nombre del producto y de valor el precio. Realizar un menú para lograr todas las acciones previamente mencionadas.

EJERCICIOS DE APRENDIZAJE EXTRA

Estos van a ser ejercicios para reforzar los conocimientos previamente vistos. Estos pueden realizarse cuando hayas terminado la guía y tengas una buena base sobre lo que venimos trabajando. Además, si ya terminaste la guía y te queda tiempo libre en las mesas, podes continuar con estos ejercicios extra, recordando siempre que no es necesario que los termines para continuar con el tema siguiente. Por ultimo, recordá que la prioridad es ayudar a los compañeros de la mesa y que cuando tengas que ayudar, lo más valioso es que puedas explicar el ejercicio con la intención de que tu compañero lo comprenda, y no sólo mostrarlo. ¡Muchas gracias!

1. Diseñar un programa que lea una serie de valores numéricos enteros desde el teclado y los guarde en un ArrayList de tipo Integer. La lectura de números termina cuando se introduzca el valor -99. Este valor no se guarda en el ArrayList. A continuación, el programa mostrará por pantalla el número de valores que se han leído, su suma y su media (promedio).

2. Crear una clase llamada CantanteFamoso. Esta clase guardará cantantes famosos y tendrá como atributos el nombre y discoConMasVentas.
   Se debe, además, en el main, crear una lista de tipo CantanteFamoso y agregar 5 objetos de tipo CantanteFamoso a la lista. Luego, se debe mostrar los nombres de cada cantante y su disco con más ventas por pantalla.
   Una vez agregado los 5, en otro bucle, crear un menú que le de la opción al usuario de agregar un cantante más, mostrar todos los cantantes, eliminar un cantante que el usuario elija o de salir del programa. Al final se deberá mostrar la lista con todos los cambios

3. Implemente un programa para una Librería haciendo uso de un HashSet para evitar datos repetidos. Para ello, se debe crear una clase llamada Libro que guarde la información de cada uno de los libros de una Biblioteca. La clase Libro debe guardar
   el título del libro, autor, número de ejemplares y número de ejemplares prestados.
   También se creará en el main un HashSet de tipo Libro que guardará todos los libros creados.
   En el main tendremos un bucle que crea un objeto Libro pidiéndole al usuario todos sus datos y los seteandolos en el Libro. Despues, ese Libro se guarda en el conjunto y se le pregunta al usuario si quiere crear otro Libro o no.
   La clase Librería contendrá además los siguientes métodos:

   - Constructor por defecto.
   - Constructor con parámetros.
   - Métodos Setters/getters
   - Método prestamo(): El usuario ingresa el titulo del libro que quiere prestar y se lo busca en el conjunto. Si está en el conjunto, se llama con ese objeto Libro al método. El método se incrementa de a uno, como un carrito de compras, el atributo ejemplares prestados, del libro que ingresó el usuario. Esto sucederá cada vez que se realice un préstamo del libro. No se podrán prestar libros de los que no queden ejemplares disponibles para prestar. Devuelve true si se ha podido realizar la operación y false en caso contrario.
   - Método devolucion(): El usuario ingresa el titulo del libro que quiere devolver y se lo busca en el conjunto. Si está en el conjunto, se llama con ese objeto al método. El método decrementa de a uno, como un carrito de compras, el atributo ejemplares prestados, del libro seleccionado por el usuario. Esto sucederá cada vez que se produzca la devolución de un libro. No se podrán devolver libros donde que no tengan ejemplares prestados. Devuelve true si se ha podido realizar la operación y false en caso contrario.
   - Método toString para mostrar los datos de los libros.

4. Almacena en un HashMap los códigos postales de 10 ciudades a elección de esta página: https://mapanet.eu/index.htm. Nota: Poner el código postal sin la letra, solo el número.

- Pedirle al usuario que ingrese 10 códigos postales y sus ciudades.
- Muestra por pantalla los datos introducidos
- Pide un código postal y muestra la ciudad asociada si existe sino avisa al usuario.
- Muestra por pantalla los datos
- Agregar una ciudad con su código postal correspondiente más al HashMap.
- Elimina 3 ciudades existentes dentro del HashMap, que pida el usuario.
- Muestra por pantalla los datos