Ejercicios UT8 –Estructuras de datos: colecciones.

Programas en los que aparecen listas y conjuntos.

1. Desarrolle un programa que solicite números enteros al usuario y los almacene en una lista (ArrayList). Cuando el usuario introduzca el número cero, el programa debe mostrar la suma de todos los números, la media aritmética, la cantidad de números que superan la media y finalmente mostrar la lista de números ordenada.

2. Incorpore en el programa anterior el proceso de elaboración de una lista adicional con todos los números distintos introducidos por el usuario, es decir, si un número aparece en la lista original dos o más veces, en la nueva lista debe aparecer sólo una vez (HashSet) . Muestre la lista ordenada al final del programa.

3. Desarrolle un programa en el que participe una clase denominada Empleado cuyos atributos serán el nombre y el número de la seguridad social, ambos de tipo String, y el sueldo base de doble precisión. Incorpore dos clases adicionales que hereden de Empleado. La primera, llamada Comercial, dispondrá de un atributo comisión de tipo doble y la segunda llamada Consultor tendrá un atributo denominado precioHora de doble precisión y un atributo numHoras entero. Añada una clase repositorio que permita almacenar Empleados (plural) utilizando una lista (ArrayList ó LinkedList) como estructura de datos. Desarrolle en esta clase los métodos que le permitan añadir un empleado, eliminar un empleado, encontrar un empleado por número de la seguridad social, ordenar la lista de empleados alfabéticamente por nombre y obtener la información de todos los empleados, indicando delante de cada uno si se trata de un Comercial o de un Consultor.

4. Desarrolle un programa que maneje dos listas enlazadas (LinkedList). En la primera debe incorporar un conjunto de países, por ejemplo Japón, Sudáfrica, Canadá, Brasil, Australia, Argentina, India y Dinamarca. Y en la segunda sus capitales, por ejemplo Tokio, Pretoria, Ottawa, Brasilia, Camberra, Buenos Aires, Nueva Delhi y Copenhague. A continuación, el programa debe insertar los elementos de la lista de capitales en la lista de países, de manera que cada país estará seguido de su capital. La lista final de los países, siguiendo con el ejemplo, tendría que quedar así: Japón, Tokio, Sudáfrica, Pretoria, Canadá, Ottawa, Brasil, Brasilia, Australia, Camberra, Argentina, Buenos Aires, India, Nueva Delhi, Dinamarca, Copenhague. Finalmente solicite al usuario que introduzca una letra y elimine de la lista final de países (en la que ahora también hay capitales) todos aquellos y aquellas cuyo nombre comiencen por dicha letra. Utilice una versión sin iteradores y otra con ellos para observar las ventajas de estos últimos.

5. Una empresa de venta por internet de productos electrónicos ha encargado implementar un carrito de la compra. Cree la clase Carrito. Al carrito se le pueden ir agregando elementos que se guardarán en una lista, por tanto, deberá crear la clase Elemento. Cada elemento del carrito deberá contener el nombre del producto, su precio y la cantidad (número de unidades de dicho producto). Incorpore métodos para visualizar el contenido del carro de la compra y mantener actualizado su contenido.

6. Desarrolle una clase llamada EntidadTerritorial formada por dos campos. El primero es la denominación y el segundo será la naturaleza, atributo este último de tipo enumerado y cuyos valores serán Provincia y Comunidad Autónoma. Diseñe una clase EntidadesTerritoriales (plural) que tenga como atributo una lista de elementos EntidadTerritorial y que implemente la interface Iterable, de manera que el método iterator sólo se desplace por las CCAA de la lista. Pruebe el funcionamiento del iterador en un programa que muestre las CCAA de la lista.

7. Deseamos crear una aplicación que gestione una lista de personas con atributos nombre y edad. Queremos ordenar esta lista de dos maneras: Por edad, usando la interface Comparable para definir el orden natural y por nombre usando la interface Comparator para definir un orden externo. La clase Persona tendrá los atributos nombre y edad e implementará la interface Comparable<Persona> para definir el orden natural basado en la edad. Adicionalmente debe crear una clase llamada ComparadorNombre que implemente la interface Comparator<Persona> para definir en ella un orden basado en el nombre. El programa principal creará una lista de objetos Persona y ordenará la lista utilizando Collections.sort primero por el orden natural (Comparable) y luego por el orden externo del comparador definido (Comparator).

Programas que utilizar Pilas

8. Se desea desarrollar un método que permita verificar si las expresiones matemáticas ingresadas por los usuarios tienen todos los paréntesis correctamente balanceados. Una expresión está balanceada si cada paréntesis de apertura tiene un paréntesis de cierre correspondiente y en el orden correcto. Para ello puede utilizar una pila en la que cada vez que aparezca un paréntesis de apertura lo guardará en la pila y cada vez que reciba un paréntesis de cierre, si no hay uno de apertura en la pila entonces puede afirmar que los paréntesis de la expresión no están balanceados. Si lo hay, extrae el de apertura de la pila y continúa el análisis.
9. Desarrolle una calculadora que permita evaluar expresiones en notación polaca inversa. La calculadora sólo contará con las cuatro operaciones aritméticas básicas. Dispone de información sobre la notación polaca inversa y el algoritmo de evaluación en este enlace: https://es.wikipedia.org/wiki/Notaci%C3%B3n_polaca_inversa

Programas que utilizan estructuras Hash

10. Elabora un programa que permita llevar a cabo la gestión y el seguimiento de su horario académico semanal, La aplicación debe permitir dar de alta el horario semanal, visualizar el horario íntegro semanal y consultar el mismo de manera que, a partir de una fecha y una hora concretas, el sistema indicará qué módulo le corresponde.

11. Crear una clase Futbolista con los siguientes atributos y métodos. Los atributos serán: Nombre: nombre del futbolista, Edad: edad del futbolista, Posición: Posición que ocupa en el campo el futbolista. Será un tipo enumerado. Podrá ser cualquiera de las siguientes: Portero, Defensa, Centrocampista ó Delantero. Cada posición tendrá asociada un salario máximo y un salario mínimo. Portero entre 2000 y 3000. Defensa entre 2500 y 3500.

Centrocampista, entre 3500 y 4000 y Delantero, entre 4000 y 5000. Incorpore los métodos getter y setter así como el método toString.

Además, Futbolista implementará la interfaz Deportista. Esta interfaz poseerá los siguientes métodos: getAnyosProfesional. Retorna el número de años que un deportista ha estado compitiendo en nivel profesional. getListadoEquipos. Devuelve un listado con el nombre de todos los equipos en los que ha estado el deportista. getTotalTrofeos. Obtiene el total de trofeos conseguidos por el deportista.

También se debe crear una clase Equipo que tenga como atributos: Nombre: nombre del equipo, Listado de Futbolistas. Listado de futbolistas del equipo, Y se deberá controlar el número máximo de futbolistas por posición. Que serán los que a continuación se definen: Portero: 2 como máximo, Defensa: 5 como máximo, Centrocampista: 5 como máximo, Delantero: 4 como máximo.

Además, constará de los siguientes métodos: agregarFutbolista. Añadirá un futbolista al equipo siempre que la posición de este no esté completa. En caso contrario lanzará una excepción de tipo RegistroFutbolistaException. listarFormaciónDelEquipo. Listará todos los miembros del equipo ordenados por posición. Se mostrarán en el siguiente orden: Porteros, Defensas, Centrocampistas, Delanteros.

La gestión se hará en dos pasos: Primero generará el listado ordenado, para ello debe usar el método sort asociado al ArrayList, y deberá sobreescribir el método compare. Luego recorrerá este con un iterator y mostrará la información de cada elemento.

Respecto a la clase Main, simplemente crear un equipo y añadir futbolistas de diferente tipo, generar una solución dónde se pruebe todo lo implementado. Es decir, el hecho de superar el número máximo de futbolistas en cada posición en cada equipo y también que se vea el funcionamiento del método que ordena.

12. Una pequeña tienda de barrio le ha solicitado desarrollar un programa sencillo para vender su mercancía. En esta primera versión del programa se tendrán en cuenta los productos que se indican en la tabla junto con su precio. Los productos se venden en bote, brick, etc. Cuando se realiza la compra, hay que indicar el producto y el número de unidades que se compran, por ejemplo “guisantes” si se quiere comprar un bote de guisantes y la cantidad, por ejemplo “3” si se quieren comprar 3 botes. La compra se termina con la palabra “fin. Suponemos que el usuario no va a intentar comprar un producto que no existe. Utiliza un diccionario para almacenar los nombres y precios de los productos y una o varias listas para almacenar la compra que realiza el usuario.

Avena	Garbanzos	Tomate	Jengibre	Quinoa	Guisantes
2,21	2,39	1,59	3,13	4,50	1,60

13. Realice una nueva versión del ejercicio anterior con las siguientes mejoras: Si algún producto se repite en diferentes líneas, se deben agrupar en una sola. Por ejemplo, si se

pide primero 1 bote de tomate y luego 3 botes de tomate, en el extracto se debe mostrar que se han pedido 4 botes de tomate. Después de teclear “fin”, el programa pide un código de descuento. Si el usuario introduce el código “ECODTO”, se aplica un 10% de descuento en la compra.