EJERCICIOS Set Curso 2020 2021

1. Ejercicio Persona con HashSet y TreeSet resuelto en la página que se indica.Resolver el Ejercicio Hotel propuesto al final de la misma página. https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=610:interfaces-set-y-sortedset-del-api-java-clases-hashset-y-treeset-ordenado-ejemplo-diferencias-cu00924c&catid=58&Itemid=180
2. Generar 20 números, con rango entre 5 y 55, sin repeticiones.
3. Una empresa desea clasificar a sus clientes en función de su edad ([0,14),[14, 18),[18,..)). Dado un conjunto de personas, obtener un array con los tres conjuntos resultantes.
4. Sean s1 y s2, los conjuntos que respectivamente representan a todos los alumnos matriculados en una asignatura en el presente curso académico y en el anterior.
   Set permite simular operaciones sobre conjuntos: unión mediante el método addAll(Collection), intersección (sólo elementos comunes a los dos conjuntos) mediante el método retainAll(Collection) y la diferencia mediante el método removeAll(Collection). Tener en cuenta que algunos alumnos pueden estar en s1 y s2 al mismo tiempo.
   Se desea realizar:
   a. un método que a partir de ambos conjuntos, muestre un listado con los alumnos repetidores que **no** han pasado al siguiente curso. Los que están en s1 y no en s2.
   b. un método que a partir de ambos conjuntos, muestre un listado con cada alumno matriculado en uno u otro curso. Es decir, el grupo completo de alumnos del centro.
   c. Listado con los alumnos que han pasado de curso con alguna asignatura pendiente.
   d. un método que a partir de ambos conjuntos, muestre un listado con los alumnos NO repetidores en el presente curso. son los alumnos que no tienen pendientes.
5. Dados tres conjuntos con los alumnos aprobados respectivamente en un examen del idioma inglés, el francés y el alemán. Se pide implementar un método que devuelva un array con los tres conjuntos siguientes:
   a. candidatos que superaron la prueba de inglés y otra más.
   b. candidatos que superaron al menos dos idiomas.
   c. candidatos que superaron sólo un examen.
6. Dado un array de palabras, donde existirán repeticiones (yo, vengo, yo, voy, yo, soy), se propone crear dos conjuntos para listar por separado las palabras que están sin repetir (vengo, voy ,soy), y las palabras que sí lo están (yo).
7. Clase Persona (nombre, apellidos y edad); ordenación natural por apellido, nombre y edad; igualdad especificada por los tres campos). Establecer un segundo criterio de ordenación por edad. Se pide probar la creación de un SortedSet utilizando los 4 constructores siguientes:
   a. TreeSet()
   b. TreeSet(Collection c)
   c. Treset(Comparator c)
   d. TreeSet(SortetSet s)
8. Crear un conjunto ordenado de 10 números enteros (orden natural) y probar los métodos siguientes:

a. tailSet
b. headSet
c. subSet

9. Implemente una clase llamada Ejercicio10 que tenga los métodos siguientes:

a. Un método añadirPalabras que reciba como parámetro un conjunto ordenado y añada cadenas a este conjunto. Las cadenas se leerán por teclado y el bucle estará controlado por una variable booleana también leída por teclado.
b. Un método mostrarSubconjunto que reciba un conjunto ordenado de objetos de la clase String como parámetro y muestre en la pantalla el subconjunto de cadenas comprendidas entre dos cadenas leídas por el teclado. Para poder mostrar el subconjunto, la cadena primera tiene que ser menor que la segunda, si no, se eleva una excepción de tipo IllegalArgumentException que se debe capturar.
c. Un método mayoresOigualesQue que reciba un conjunto ordenado de objetos de la clase String como parámetro y muestre en la pantalla las cadenas del conjunto mayores o iguales a una cadena leída por el teclado.
d. Un método menoresQue que reciba un conjunto ordenado de objetos de la clase String como parámetro y muestre en la pantalla las cadenas del conjunto menores estrictamente que una cadena leída por el teclado.
e. Un método main en el que se cree un conjunto ordenado llamado diccionario y se prueben los métodos anteriores. Además muéstrese en la pantalla el tamaño del diccionario, la primera y última cadena del diccionario y el comparador con el que están ordenadas las cadenas que forman parte de él.

10. Implemente una clase llamada comparadorLongitud donde se defina como criterio de comparación entre dos cadenas su longitud. En el caso de que dos cadenas tengan la misma longitud el orden será el orden natural de los objetos de la clase String. En el método main del ejercicio anterior, cree un nuevo conjunto ordenado según este comparador llamado nuevodiccionario donde se añaden todas las palabras del diccionario anterior y algunas nuevas pedidas mediante teclado, utilizando añadirPalabras. Imprimir en la pantalla el nuevo diccionario para ver como queda ordenado según el nuevo orden. Imprima además el comparador usado.
11. Siga la traza escribiendo en las líneas numeradas, la salida que se produciría por pantalla. En las líneas en blanco, implemente el código necesario para realizar lo que se indique en los comentarios correspondientes. Los prototipos de los métodos disponibles en la clase Entero son: Entero(int), int get(), void set(int), Entero suma(Entero,Entero), String toString(), int compareTo(Object).

```
import java.util.*;
class prueba {
public static void main (String args[]) {
List l1 = new ArrayList();
List l2 = new ArrayList();
Entero e1 = new Entero(0);
l1.add(e1);
for (int i=1; i<=6; i++) {
l1.add(new Entero(i));
l2.add(new Entero(i+3));
}
e1.set(5);
l2.add(e1);
```

```
((Entero)(l1.get(0))).set(10);
l1.set(5,new Entero(4));
System.out.println("l1: "+l1.toString()); (1)
System.out.println("l2: "+l2.toString());(2)
System.out.println("l1: "+l1.subList(0,3).toString());(3)
System.out.println("l2: "+(l2.indexOf(new Entero(3)))); (4)
// Escriba el código que muestre los tres últimos elementos de la lista l2,
// utilizando iteradores específicos para listas
```

_____
_____
_____
_____

```
SortedSet s1 = new TreeSet(l1);
SortedSet s2 = new TreeSet();
Set s3 = s1;
s3.add(new Entero(12));
s2.addAll(l2);
s2.add(l1.get(1));
s2.retainAll(l1);
System.out.println("s1: "+s1.toString());(5)
System.out.println("s2: "+s2.toString());(6)
// Escriba el código que sume 10 a todos los elementos del conjnto s1
// y mostrar los elementos de s1 superiores a 15
```

_____
_____
_____
_____
_____
_____

```
System.out.println("s1: "+s1.tailSet(new Entero(15)).toString());
System.out.println("s1: "+s1.toString()); (7)
System.out.println("s2: "+s2.toString()); (8)
System.out.println("l1: "+l1.toString());(9)
System.out.println("l2: "+l2.toString());(10)
}
}
```

(1)_____
(2)_____
(3)_____
(4)_____
(5)_____
(6)_____
(7)_____
(8)_____
(9)_____
(10)_____

b) Escriba el código necesario para que funcione correctamente la clase Set utilizada en la clase prueba.

12. Se dispone de la clase ConjuntoS para implementar conjuntos de cadenas de tipo String. Los prototipos de los métodos de esta clase son:

int tamaño(); // devuelve el nº de cadenas que contiene el conjunto.
boolean esVacio(); // devuelve true si el tamaño es 0.
boolean añadir(String s); // añade una nueva cadena al conjunto.
int comunes(ConjuntoS c);
// devuelve el nº de elementos en común con el conjunto pasado por parámetro.
List aLista(); // devuelve una lista con los elementos del conjunto.
ConjuntoS(); // crea un conjunto vacío.
ConjuntoS(ConjuntoS c); // crea un conjunto a partir de otro.

Implemente un método estático examen tal que dado un ConjuntoS c devuelva otro formado por aquellas cadenas que no tengan en común el primer carácter con respecto a alguna otra cadena de c. Es decir, si por ejemplo c = {“d”, “abad”, “bcd”, “xabcd”, “cd”, “xb2”, “bcd2”, “ybcd”} entonces el conjunto resultante debe ser {“d”, “abad”, “cd”, “ybcd”} ya que el resto de cadenas de c tienen el primer carácter en común con alguna otra cadena de c. Hay que tener en cuenta que cada cadena se debe comparar con el resto de cadenas del conjunto, es decir, sin contar ella misma porque si no siempre se devolvería el conjunto vacío.

Para ello deberá construir y utilizar el método public static boolean encomúnprimero(String p, List ls) tal que dado un String s y una lista de String ls devuelve un valor lógico que es cierto si p tiene el primer carácter en común con alguna cadena de ls y falso en caso contrario. Debe utilizar el método char charAt(int) de String.

13. Con motivo de la celebración del cuarto centenario del Quijote, un conocido grupo editorial planea lanzar un DVD con contenido multimedia sobre la obra y vida de Cervantes. Ante las opiniones encontradas que ha generado tal iniciativa, se ha decido desarrollar un prototipo con el siguiente contenido:

• Los dos tomos del Quijote con ilustraciones en alta resolución y funciones de narración por audio, acceso directo y zoom.
• Un glosario sobre el castellano antiguo.
• Un índice de todos los personajes de ambos tomos con artículos y relaciones entre los mismos.

Tras encargar el proyecto a tres empresas de servicios informáticos y siendo cada módulo desarrollado por una empresa distinta, en la última reunión mantenida por los jefes de proyecto
de cada empresa se acordó respetar, en función de los requisitos actuales y de las futuras ampliaciones, las siguientes interfaces:

public interface ITomo {
List getCapitulos(); // lista ordenada de capítulos

```
// pertenecientes al tomo
List getPersonajes();// lista ordenada de personajes que aparecen
// en el tomo
}
public interface ICapitulo {
String getTitulo(); // título del capítulo
List getPersonajes(); // lista ordenada de personajes que
//aparecen en el capítulo
}
public interface IPersonaje {
String getApodo(); // p.e. “Don Quijote”, “Dulcinea del Toboso” String getNombre(); // p.e. “Alonso Quijano”, “Aldonza Lorenzo”
Set getPersonajes(); // conjunto de personajes vinculados
List getCapitulos(); // lista ordenada de capítulos donde aparece
}
```

Se pide:

Un método que a partir de un tomo t y un personaje p muestre en pantalla, ordenados ascendentemente según el número del capítulo donde aparecen por primera vez, el nombre de cada uno de los personajes vinculados a p en t, junto al nombre de dicho capítulo. En la implementación se utilizara un Set para evitar la repetición de personajes.

• Un método que devuelva el personaje, distinto a “Don Quijote”, con mayor número de personajes vinculados (véase el método getPersonajes de la interfaz Ipersonaje) en un mismo capítulo c de un tomo t.

Ambos métodos se incluirán en una clase donde uno de sus atributos, de nombre tomos, es un array de ITomo de tamaño 2. El orden natural de los capítulos y de los personajes es, respectivamente, el de su aparición en la novela y el orden alfabético.

14. Tenemos una cadena de montaje de robots de distintos tipos. Hemos modelado las distintas piezas de los robots a partir de la clase PiezaRobot:

```
public class PiezaRobot {
private int referencia; // número de fabricación único para cada pieza
private double claveRobot; // tipo de robot al que pertenece la pieza
private String pieza; // tipo de pieza
private String testCalidad; //resultado de test de calidad sobre la pieza
// Implementación de métodos set y get.
}
```

Las piezas son sometidas a un test de calidad, cuyos posibles resultados son las siguientes cadenas:

“tr1”: test OK

“tr0”: test NO OK

“tr10”: pieza no correspondiente a la cadena de montaje

Existe un método ya implementado llamado:

```
public static PiezaRobot damePiezaValida(PiezaRobot p, SortedSet piezas)
```

el cual recibe un objeto PiezaRobot y un conjunto ordenado de objetos PiezaRobot. Este método busca en el conjunto de piezas un objeto de las mismas características (mismos atributos claveRobot y pieza) que el objeto que se pasa como parámetro y además con calidad válida (testCalidad=”tr1”) y lo devuelve. Si no encuentra un objeto de dichas características devuelve null.

Se pide:

a) Implemente un método que reciba como parámetros una lista de PiezaRobot y un conjunto ordenado de PiezaRobot para repuesto. Este método deberá recorrer la lista y revisar la calidad de cada una de las piezas de tal forma que si la calidad es:

- “tr1”: la pieza es válida. No hay nada que hacer.
- “tr0”: la pieza es errónea. Deberá sustituirla por una nueva pieza válida. Para ello utilice el método damePiezaValida y el conjunto de piezas de repuesto.
- “tr10”: la pieza es de otro robot. Deberá eliminarla de la lista.

En el caso de que no encontremos una pieza de sustitución habrá que elevar y propagar la excepción CadenaIncompletaException con la información de las características del robot (claveRobot y pieza) que no se ha podido sustituir. Suponga que la excepción CadenaIncompletaException ya está implementada.

b) Implemente los cambios necesarios en la clase PiezaRobot para ordenar las piezas de los robots primero por el atributo claveRobot y después por el atributo pieza.