Ejercicios

Unidad 11

# Colecciones



1. Realizar un programa que contenga una LinkedList para almacenar las matrículas de los coches aparcados en un parking. El parking es un poco raro, mide solo 3 metros de ancho y caben 10 coches, pero uno detrás de otro por lo que el último en entrar debe ser el primero en salir (esta estructura se llama pila LIFO – Last Input, First Output). El programa tendrá un menú para:
   - Aparcar: se le pasará el número de matrícula y lo almacenará a no ser que esté lleno.
   - Desaparcar: Muestra la matrícula del coche a desaparcar o bien un mensaje informando cadena vacía si el parking está vacío.
   - Mostrar la lista de las matrículas de los coches que hay en el parking, por orden inverso al de llegada, es decir primero el último en llegar

   Nota: Usa los métodos que meten y sacan por el principio de la lista: addFirst, removeFirst
2. Haz una versión del ejercicio anterior, pero con los métodos que tiene LinkedList referidos específicamente a pilas (peek, poll,pop, push, etc.)
3. Haz un programa que cree un conjunto HashSet que almacene la lista de personas que van a una fiesta (de una persona sabemos su nombre, teléfono, email y fecha de nacimiento). Crea en un archivo aparte la clase Persona con los atributos y métodos que necesites. En el programa introduce “a mano” unas cuantas personas, y luego muestre la edad media y el nombre del mayor.

   Intenta hacer una inserción de una persona repetida ¿Qué ocurre? (Dos personas son iguales si tienen exactamente el mismo nombre).

   Muestra todos los valores almacenados en el HashSet ¿Tienen algún orden?

   Nota: Uso de fechas en Java
   https://www.campusmvp.es/recursos/post/como-manejar-correctamente-fechas-en-java-el-paquete-java-time.aspx
4. Repite el ejercicio anterior con un LinkedHashSet ¿Qué ha cambiado?
5. Modifica la clase Persona (llámala Persona_v2) para que dos personas sean iguales si tienen el mismo nombre sin tener en cuenta mayúsculas y minúsculas. Repite el ejercicio anterior intentando incluir en el conjunto valores que no sean permitidos por esta nueva condición (por ejemplo: Ana y ANA generarían un duplicado).
6. Realizar un programa que tenga un HashMap que almacene una plantilla de jugadores de un equipo (Nombre, salario). El programa dispondrá de un menú que permita: añadir jugador, eliminar jugador, consultar el salario de un jugador e incrementar el salario un 10% a un empleado.
7. Realizar un programa que tenga un HashMap que almacene una plantilla de jugadores de

un equipo (Nombre, salario). La plantilla tiene un máximo de 25 jugadores. El programa dispondrá de menú que permita: añadir jugador, eliminar jugador, lista jugadores con su salario y que tenga también opción que permita introducir un salario y muestre los jugadores que tiene un salario parecido al introducido (por "parecido" entendemos que el salario del jugador esté en un rango de 6000 euros arriba o abajo respecto al introducido).

8. Realizar un programa al que se le introduzca un año y genera un array con las temperaturas medias de una ciudad para todos los días de ese año (365 ó 366). La temperatura será un entero entre 10 y 30 grados. Almacena en un Mapa la distribución de temperaturas, es decir, para cada temperatura, cuantos días del año la hubo y muestra dicha distribución.

9. Realizar un programa que genera un array con las temperaturas medias de una ciudad para todos los días de un año no bisiesto. La temperatura será un entero entre 10 y 30 grados. Ayudándote de un TreeMap, muestra la temperatura mínima y cuantas ocurrencias tuvo y la temperatura máxima y cuantas ocurrencias tuvo.

10. Haz un programa con un TreeSet de Persona_v2 (es un conjunto ordenado), que introduzca "a mano" unas cuantas personas en el conjunto y a continuación las muestre con un for...each. ¿Qué ocurre? Crea una nueva versión de Persona_v3 que resuelva los problemas encontrados y que ordene y no permita duplicados por nombre sin distinguir mayúsculas de minúsculas. Si eliminases de la clase Persona_v3 los métodos equals() y hashCode(). ¿Seguiría funcionando correctamente?

11. Realiza un programa que defina un Arraylist de Persona_v3. Añade 5 personas distintas al mismo y luego muéstralas por pantalla ordenadas por email. A continuación, vuelve a mostrarlas, pero esta vez ordenadas por nombre. ¿Debes usar un Comparator en ambos casos?

12. Realiza un programa que defina un Arraylist de Persona_v3. Añade 5 personas al mismo, pero una de ellas que esté repetida. Utilizando conversiones entre colecciones (sin iterar sobre el ArrayList) elimina los repetidos, preservando el orden de los elementos del ArrayList.

13. (Avanzado) Modifica el ejercicio 11.8 para que muestre la distribución de temperaturas ordenadas por el número de veces que ha aparecido la temperatura. La solución se verá en clase.