Ejercicios sobre Collection y Map

Ejercicio 1

Implementar el tipo abstracto de datos `pila` de enteros (se podrán almacenar un número indeterminado de números enteros y a la hora de extraer información el último elemento que se haya añadido será el primero en salir) con las operaciones de:

- Apilar en la cima
- Desapilar de la cima
- Número de elementos apilados
- Visualización de elementos de la pila. Tener en cuenta que al mostrar un elemento de una pila lo borramos.

Ejercicio 2

Este ejercicio pretende desarrollar el programa de cálculo de escaños en unas elecciones, teniendo como datos de entrada: El número de partidos que se presentan (N), el número de escaños a repartir (M) y los votos que ha obtenido cada partido

A partir de aquí, hay que hacer lo siguiente se dividen los votos de cada partido entre 1, 2, 3....M y se obtienen M divisiones y así sucesivamente para el partido 2,3,...N es decir se obtendrán NxM cocientes, y de esos cocientes se eligen los M mayores, puesto que hay M diputados a repartir

Un ejemplo sería el siguiente:

PARTIDOS: 4
ESCAÑOS: 5

	Escaños				
Partidos	1	2	3	4	5
Part1	**10000**	**5000**	**3333**	**2500**	**2000**
Part2	**25000**	**12500**	**8333**	**6250**	**5000**
Part3	**30000**	**15000**	**10000**	**7500**	**6000**
Part4	**12000**	**6000**	**4000**	**3000**	**2400**

Una vez hechos los cálculos habrá que ver a qué partidos le corresponden los **5 mayores cocientes.**

El listado de resultados debe hacerse ordenado por número de votos.

Los partidos que no obtengan un 5% de los votos emitidos deben rechazarse, y no entrarán en el reparto de escaños.

NOTA: Obligatorio el uso de colecciones para resolverlo, se puede usar un TreeMap con clave el cociente y valor asociado un TreeSet que en cada elemento guarda el nombre del partido a quien pertenece ese cociente. El mapa debe estar ordenado por los valores de los cocientes de mayor a menor. Si hay M escaños los M primeros elementos de los TreeSet que se guardan en el mapa nos indican a quien corresponden los escaños. En este ejercicio nos interesa recorrer los primeros nodos del mapa y llevar la cuenta del número de elementos recorridos de los TreeSet que van asociados a cada nodo porque cuando tengamos 5 en total se termina.

Se pueden utilizar otros enfoques con colecciones para resolverlo.

Ejercicio 3.

En este ejercicio se generan los datos a partir de dos vectores de 10 nombres y 10 apellidos combinándolos al azar. Hay que generar un ArrayList de 50 alumnos para guardar el nombre, el apellido y las notas obtenidas en tres exámenes parciales que se puntúan entre 0 y 100.

El programa debe:

1. Generar al azar los datos de 50 alumnos. Cada alumno se almacenará en un elemento del ArrayList. Los datos que queremos guardar de cada alumno son:

   string nombre;
   string apellido;
   int nota1;
   int nota2;
   int nota3;
2. Después generar tres mapas (TreeMap), uno para cada nota, donde la key será la nota por la que ordenamos y el dato asociado un ArrayList con los índices del vector donde aparece esa nota (puede haber varios alumnos que tengan la misma nota).
3. Presentar un menú para poder listar la clase por cualquiera de las tres notas en orden ascendente, es decir ascen1ª, ascen2ª, ascen3ª.
4. Presentar en pantalla el listado requerido

De esta forma no hay que reordenar el ArrayList cada vez que se pide un tipo de listado pues el ArrayList, aunque este desordenado, es accesible en el orden de las notas según el mapa que se recorra.

Ejercicio 4.

Deseamos realizar una aplicación para la gestión de usuarios y claves de acceso a un sistema (red, base de datos, programa de gestión etc.)

Para ello vamos a utilizar mapas (TreeMap) donde el par de datos será:

- <usuario, clave de acceso>, los usuarios no pueden repetirse (TreeMap).
- Además para mayor seguridad las claves no se guardan tal y como la tecleamos sino que se guardan codificadas con un sencillo algoritmo que consiste en desplazar los caracteres un número fijo entero, p. ej. si la clave es ABC y el número fijo es 2 la clave será ABC pero en el mapa guardamos CDE, de forma que si nos pillan el fichero de claves, dicho fichero no sería el real sino el codificado.
- Las claves deben ser de tal forma que solo admitan mayúsculas, minúsculas y números y ningún otro carácter debe ser admitido.

Crear la aplicación para que se pueda:

- ✓ Insertar usuario, clave
- ✓ Borrar usuarios
- ✓ Modificar claves de un usuario
- ✓ Validar a un usuario si su clave es correcta

Ejercicio 5.

Simulación de colas de espera en una oficina de atención a clientes.

En una oficina de atención a clientes se disponen de 5 ventanillas para los mismos, cada una de ellas genera una cola de espera.

Cada nodo de información que se inserta en la cola tiene una tarea a despachar que puede ser de duración 10 min, 20 min ó 30 min , se ha estimado estadísticamente que llegan un 60% de tareas de duración 10 un 25% de tareas de duración 20 y un 15% de tareas de duración 30.

El programa debe generar con los porcentajes descritos un nodo de tipo 15, 20 o 30. Se sugiere generar números aleatorios entre 1 y 100 de manera que si el número generado está entre 1 y 60 la tarea será de duración 15, si está entre 61 y 85 de duración 20 y si está entre 86 y 100 de duración 30.

Cuando se genera un cliente(es decir nodo ó tarea), este se coloca en la cola cuya longitud es mínima.

Se ha estimado estadísticamente también que cada 5 minutos llega un nuevo cliente (nodo/tarea).

El programa debe simular el comportamiento de las 5 colas durante 5 horas es decir se genera un bucle de 1 a 300 porque son trescientos minutos los que transcurren en ese intervalo. Para cada minuto, es decir para cada paso del bucle hay que hacer dos cosas:

1. Si el minuto es múltiplo de 5 generar un cliente (tarea/nodo) y ponerlo en su cola correspondiente.
2. Dar servicio a las 5 ventanillas esto es las tareas que estén en el frente de cada cola descontarles un minuto que ha transcurrido y si quedan en cero suprimirlas de la cola.

El programa debe darnos las longitudes de cada cola de espera (ventanilla) cada 15 minutos. Y al final la longitud media de cada cola, obtenidas como la suma de las longitudes de dicha cola en cada instante dividida por el número de instantes totales

Los métodos que se proponen para este ejercicio de simulación son las siguientes:

ServirColas : su tarea es dar servicio a las 5 colas decrementando en uno los minutos de su gestión , a las cabeceras de dichas colas y en el caso de que los minutos de ese nodo queden a cero suprimirlos de la cola.

MedirCola : su función es darnos el número de nodos de la cola en un momento dado.

AsignarCola : su función es indicar donde debe insertarse la tarea que se genera cada 5 minutos.

InsertarTarea : insertar una tarea en la cola que le corresponda.

ExtraerTarea : eliminar una tarea cuyo número de minutos haya terminado.

Estos métodos serán indispensables para la buena ejecución del programa, pero si se considera oportuno, se pueden crear otros métodos auxiliares.