Ejercicios "COLAS"

Ejercicio nº 1.-

Una cola medieval se comporta como una cola ordinaria, con la única diferencia que los elementos almacenados en ella se dividen en dos estamentos: nobles y plebeyos. Dentro de cada estamento, los elementos deben ser atendidos en orden de llegada; pero siempre que haya nobles en la cola, éstos deben ser atendidos antes que los plebeyos. Realiza una aplicación en Java que nos permita realizar la tarea que se describe a continuación.

Especificar un tipo abstracto de datos para las colas medievales que disponga de operaciones para:

- Crear una cola medieval vacía.
- Añadir un elemento nuevo a una cola.
- Consultar el primer elemento de una cola.
- Quitar el primer elemento de una cola
- Consultar el número de nobles en una cola.
- Consultar el número de plebeyos de una cola
- Consultar si una cola es vacía o no.

Crear un programa principal, que nos permita mediante un menú de opciones, comprobar el funcionamiento de la clase, para ello ten en cuenta de que se guarda la siguiente información de los nobles y de los plebeyos, cuando llegan a ser atendidos.

Noble:	Plebeyo:
- Nombre - Cargo - Tema que quiere tratar	- Nombre - Tema que quiere tratar

Ejercicio nº 2.-

Se desea implementar una estructura de datos que permita gestionar las tareas que se ejecutan en una CPU multitarea. Dentro del conjunto de tareas se pueden diferenciar cinco niveles de prioridad, de 1 a 5, siendo el nivel más prioritario el «1». Cuando hay tareas de nivel «1» éstas serán las primeras en ejecutarse, si no hay de este nivel se ejecutarán las de nivel 2 y así sucesivamente hasta que no haya tareas. El orden de ejecución dentro de un mismo nivel de prioridad es por orden de llegada. Realizar un programa en Java que simule la selección de tareas de la CPU, seleccionar las estructuras a utilizar acorde al planteamiento.

Ejercicio nº 3.-

Un concesionario de vehículos tiene un número limitado de modelos, todos en un número limitado de colores distintos. Cuando un cliente quiere comprar un automóvil, lo solicita de un modelo y color determinados. Si el vehículo de ese modelo y color no está disponible en el concesionario, se toman los datos del cliente (Nombre y teléfono), y se almacenan para ser atendida su petición cuando el vehículo esté disponible. Si hay más de una petición con las mismas características, se atienden las peticiones por orden cronológico.

a) Definir la estructura de datos más adecuada capaz de contener las peticiones de un modelo de y color de vehículo
b) Definir una operación que, dado un cliente (nombre y teléfono) que desea comprar un modelo y color determinado, coloque sus datos como última petición de ese modelo y color.

Ejercicio nº 4.-

Un restaurante dispone de «n» número de mesas. De cada mesa se sabe su código de identificación. Hay una cola de espera para ir ocupando las mesas, de forma que, para cada petición se sabe el nombre de la persona que ha hecho la reserva y el número de comensales que no debe exceder de 5. Definir una operación que, dado un identificador de mesa libre, devuelve el nombre de la persona que se encuentra de primera de la lista de clientes que hayan hecho una reserva indicándole que puede pasar al comedor. La estructura debe quedar convenientemente actualizada una vez la persona pase al comedor saliendo así de la lista de espera. No se permiten reservas que excedan el número de comensales por mesa. Realizar el programa que nos permita representar la reserva de mesas del restaurante.

Ejercicio nº 5.-

En una tienda hay sólo 1 caja registradora, en la cual se colocan los clientes con sus carros de la compra en orden de llegada. De la caja registradora se guarda el número identificador de la caja, la recaudación acumulada y los carros en espera. Por otro lado, en cada carro se amontonan los distintos productos, de modo que tan sólo puede añadirse o extraerse el situado en la parte superior. Por cada producto guardamos su nombre y precio.

- Definir la estructura más adecuada para guardar una caja registradora explicar ¿por qué?
- Definir la estructura más adecuada para guardar un carro de la compra y explicar ¿por qué?
- Escribir una función denominada «***atenderCliente***» que, dado un carro de la compra, pase los productos que contiene por caja y calcule el costo a pagar.
- Desarrollar una función que calcule la «***recaudación***» de la caja después de pasar los primeros «n» carros. «n» será un argumento de entrada que no puede ser mayor que el número de carros en espera de la caja

Utilizar la estructura de datos más adecuada para devolver el resultado pedido en cada uno de los casos.