Boletín 1. Estructura de Datos Avanzados:

Ejercicio 1. Genéricos:

Crear un método genérico ***reverse*** que devuelva un array de ***objetos*** con la misma información, pero en orden inverso, es decir donde en la primera posición esté la información del último objeto, en la segunda la penúltima y así sucesivamente.

protected static <T> T[] reverse (T[] arrayOriginal)

Ejercicio 2. Equipos deportivos:

Escribir un programa para gestionar grupos de alumnos apuntados a equipos deportivos. El programa dispondrá de:

- La Clase Alumno, con los atributos: nombre y DNI de tipo String. Implementar constructor y los métodos toString() y equals.
- La Clase Equipo. Cada equipo deportivo se representa en una clase que contiene el nombre del equipo y un grupo de alumnos.

El equipo tiene operaciones para:

- Añadir un alumno (recibe como parámetro el objeto alumno a insertar). Si el alumno ya existe en el equipo debe saltar una excepción.
- Borrar un alumno (recibe como parámetro el objeto alumno a borrar). Si el alumno no existe en el equipo debe saltar una excepción.
- Saber si un alumno pertenece al equipo. Recibe como parámetro el objeto alumno a buscar y devuelve null si no lo encuentra y el objeto alumno si existe.
- Mostrar en pantalla la lista de personas del equipo.
- Unión de equipos. El método lo llamará un equipo y se le pasará por parámetro el otro equipo con el que se quiere unir. Devuelve un equipo formado por los alumnos de ambos equipos.
- Intersección de equipos. Idem al anterior pero devuelve un equipo formado por los alumnos que están en los dos equipos

Se pide:

1. Decidir la estructura de almacenamiento más indicada para este ejercicio. No debe establecerse ningún límite de jugadores en los equipos.
2. Realizar las clases Alumno, Equipo así como un programa de prueba que cree varios equipos y prueba todas las operaciones mostrando en pantalla los resultados.

Ejercicio 3. Ampliación con genéricos:

Modificar la clase anterior para que Equipo sea una clase genérica y se pueda aplicar a cualquier tipo de equipo.

Realizar dos programas que prueben la clase Equipo parametrizada con la clase Alumno y otro que lo pruebe con la clase Integer.

Ejercicio 4. Historial de navegación:

Realizar un programa que permita almacenar un historial de páginas web consultadas por un usuario. De cada página web se almacenará su url y la fecha y hora (LocalDateTime) en la que se visitó.

Se debe realizar un menú de este tipo

1. Nueva página consultada: Se solicitará el nombre de la página y se tomará la fecha y hora del sistema añadiendo ésta al historial. No se permitirá introducir una fecha y hora anterior a la última almacenada.
2. Consultar historial completo.
3. Consultar historial de un día.
4. Borrar todo el historial.
5. Borrar visitas a una página.
6. Salir.

Se pide:

a. Decidir la estructura de almacenamiento más indicada para este ejercicio. No debe establecerse ningún límite de entradas en el historial.
b. Implementar la clase PaginaWeb, así como la clase Historial y el programa Principal.

Ejercicio 5. Chat de instituto:

Se desea desarrollar una aplicación que permita enviar mensajes entre los profesores y los alumnos de un centro de enseñanza.
Se utilizará la clase abstracta Persona, así como Profesor y Alumno vistos anteriormente.
Se debe incluir las opciones para que las personas puedan enviarse mensajes. De cada mensaje se guardará el remitente, el texto del mensaje y la fecha y hora en la que se envió.
Todas las personas deben tener:

a. Un método para poder enviar un mensaje a otra persona. Recibirá como parámetro la persona destinataria y el texto del mensaje.
   Si la persona que envía el mensaje es un alumno y es menor de edad sólo puede enviar mensajes a profesores, es decir, si un alumno menor de edad intenta enviar un mensaje a otro alumno se debe provocar un error.
b. Un método para poder leer los mensajes del buzón. Este método devolverá una cadena de texgto con todos los mensajes que tiene. Si no tiene mensajes para leer saltará el correspondiente mensaje de error. Cada mensaje irá precedido de un número 1 en adelante, es decir se mostrará con este formato.
   “Mensaje X: De: nombreRemitente Texto: textoMensaje Fecha y hora: DD-MM-AAAA HH:MM”
c. Un método para poder leer los mensajes del buzón ordenados alfabéticamente por el nombre del remitente.
d. Un método para poder borrar un mensaje del buzón. Al método se le pasará el número de mensaje que se desea borrar. Si no existe ese número de mensaje saltará una excepción.
e. Un método que realice una búsqueda para poder encontrar los mensajes de su buzón que contienen una frase Este método devolverá una cadena de texto con todos los mensajes que tienen esa frase o saltará la excepción si no encuentra ningún mensaje con esa frase.

Realiza pruebas o tests de cada uno de estos métodos.

Ejercicio 6. Diccionario:

Realizar una clase que implemente un diccionario. Cada palabra puede tener varios significados. Se presentará un menú de esta forma

1. Añadir palabra. Se solicitará la palabra y su significado. Si la palabra ya tenía un significado se guardará este nuevo significado con los anteriores.
2. Buscar palabra en diccionario: Se solicitará la palabra y se mostrarán todos sus significados.
3. Borrar una palabra del diccionario: Se solicitará la palabra y se borrará, con todos sus significados.
4. Listado de palabras que empiecen por "…" Se solicitará una cadena y se mostrará un listado de palabras ordenadas alfabéticamente que comiencen por esa cadena.
5. Salir.

No debe existir límite de palabras y debes decidir el contenedor más adecuado teniendo en cuenta que sobre todo se realizarán búsquedas.

Ejercicio 7. Gran almacén:

Un gran almacén necesita crear una aplicación para gestionar las cajas (dispone de 20 cajas) y asigne el turno a los clientes (cuando llegue un cliente se le indica aquella que tenga menos gente esperando). Para ello, debes desarrollar una aplicación que ofrezca e implemente el siguiente menú de acciones:

1. Abrir caja. Se solicitará el número de caja y si no está abierta se abrirá. Si estuviese abierta se debe mostrar un mensaje de error.
2. Cerrar caja. Se solicitará el número de caja. Sólo se puede cerrar, si estaba abierta y además no hay clientes, si no se mostrará el mensaje de error oportuno.
3. Nuevo cliente: Se generará un número automáticamente, de 1 en adelante, que identificará a este cliente. Se informará "Es usted el cliente número xx y debe ir a la caja número xx". El número de caja que se le asigna es la que tenga menos clientes esperando. En caso de empate debe ir a ser la caja de menor número.
4. Atender cliente: Se solicitará el número de caja en la que está y se mostrará un mensaje indicando "Se ha atendido al cliente con número XX". El cliente abandona la caja. Si no existen clientes en esta caja o bien estuviese cerrada se mostrarán los errores oportunos.
5. Salir .

Ejercicio 8. Ampliación equipos deportivos:

Añade los campos edad, sexo (char ‘H’ ‘M’) y ciudad a los jugadores e incluye los siguientes métodos en la clase Equipo (puedes usar lambda y stream)

a. Método que devuelva un listado de los jugadores de sexo masculino que son mayores de edad ordenado por edad.
b. Método que diga si un equipo es exclusivamente femenino.
c. Método que devuelve el número de jugadoras mayores de edad que tiene un equipo.
d. Método que devuelva la edad mayor de entre todas las jugadoras mayores de edad.
e. Método que devuelva un Set de los dni de los jugadores de sexo masculino que son menores de edad.
f. Método que devuelve una lista de los nombres de las jugadoras ordenados ascendentemente.
g. Método que diga si existe alguna jugadora mayor de edad en un equipo.
h. Método que cuente cuántas ciudades diferentes hay entre los jugadores de un equipo.