UNIDAD 6: ESTRUCTURAS DE DATOS DINÁMICAS

Ejercicios

LISTAS

Ejercicio 1. Practicando listas dinámicas

Escribe un programa que pida al usuario números enteros positivos hasta introducir un número negativo. Los valores introducidos, menos el negativo, los irá guardando en una lista dinámica. A continuación, se indican los diferentes apartados de funcionalidades que irás añadiendo a este programa:

a) Muestra la lista por defecto.

b) Inserta el 101 en la primera posición.

c) Comprueba que el 101 y el -1 estén en la lista. Si está muestra "X: Encontrado" o en caso contrario "X: No encontrado".

d) Obtén la media aritmética de los valores introducidos.

e) Lee el elemento que está en la posición X, siendo X un valor introducido por el usuario. Comprueba que esté dentro del rango de índices de la lista.

f) Actualiza el valor que está en la última posición incrementándolo en 1.

g) Elimina el elemento que está en la primera posición.

h) Ordena de forma ascendente la lista.

i) Ordena de forma descendente la lista.

j) Desordena aleatoriamente los elementos de la lista.

k) Crea otra lista dinámica (ArrayList). Añade tantos números enteros como tenga la lista anterior, pero estos deben ser generados aleatoriamente entre 5 y 100. Compara esta nueva lista con la anterior e indica si son o no iguales.

l) Ahora que tienes dos listas únelas en una tercera lista.

Ejercicio 2. Practicando de estructura Pila

Escribe un programa que pida al usuario números enteros y los vaya apilando en el tipo de lista más adecuada. Estas las siguientes tareas a hacer:

a) Obtener la cima de la pila.

b) Eliminar la cima de la pila.

c) Imprimir el contenido de la pila. Mostrarlo al usuario en formato vertical como se vería en una pila lógica, es decir, el elemento de más abajo es el primero en entrar y el de arriba el último.

d) Eliminar todos los elementos de la pila, de golpe.

e) Comprobar que la pila está vacía.

Ejercicio 3. Practicando de estructura Cola

Escribe un programa que pida al usuario números enteros y los vaya encolando en el tipo de lista más adecuada. Estas las siguientes tareas a hacer:

a) Obtener la cabeza de la cola.

b) Eliminar la cabeza de la cola.

c) Imprimir el contenido original de la cola.

d) Eliminar todos los elementos de la cola, de golpe.

e) Comprobar que la cola está vacía.

Ejercicio 4. Practicando de estructura Conjunto

Escribe un programa que pida al usuario que agregue dorsales (números) de jugadores a un equipo de fútbol y estos no pueden estar repetidos. Estas las siguientes tareas a hacer:

a) Comprobar si el dorsal 10 existe en el equipo.

b) Muestra todos los dorsales del equipo.

c) Elimina el dorsal 13 del equipo.

d) Elimina todos los dorsales del equipo.

e) Comprueba que no hay dorsales en el equipo.

Ejercicio 5. Gestión de colas procesos

Escribe un programa para que la CPU atienda procesos. Para ello la CPU necesita una lista de PID (process ID) que serán números enteros. El programa empezará pidiendo al usuario que introduzca PIDs, hasta que se introduzca un valor negativo. Los PID se almacenarán en una Cola de números enteros.

Ahora, el programa mostrará un menú con las siguientes opciones:

1. Atender proceso.
2. Eliminar proceso.
3. Mostrar procesos restantes
4. Mostrar total de procesos atendidos.
5. Atender a todos los procesos.
6. Salir

Las opciones harán los siguiente:

- **Atender proceso:** mostrará el proceso a atender y se eliminará de la cola. Se indicará que el proceso con el PID X ha sido atendido.
- **Eliminar proceso:** se eliminar un proceso sin ser atendido por la CPU.
- **Mostrar procesos restantes:** muestra la lista de PIDs que quedan por atender en formato vertical.
- **Mostrar total de procesos atendidos:** muestra el total de procesos atendidos. Puede que no todos hayan sido atendidos por la CPU.
- **Atender a todos los procesos:** la CPU atiende de golpe a todos los procesos y por tanto la cola de procesos se vacía.

Para cada una de las opciones escribir una función estática que la solucione.

Ejercicio 6. Implementar un filtro de fechas

Escribe un programa que pida fechas en formato DD/MM/AAAA hasta que introduzcas la palabra “fin”. Luego, el programa deberá mostrar cuantos de los días introducidos son únicos. Por ejemplo, si tenemos las siguientes fechas:

- 01/01/2023
- 02/01/2023
- 01/01/2023
- 03/01/2013

En total hay tres fechas únicas (ignora la que está repetida).

Por cada fecha introducida por el usuario debes comprobar si es válida (formato DD/MM/AAAA, rango de valores, etc). Deberás almacenar las fechas introducidas en un array de cadenas (ArrayList). Luego, por cada fecha (String) deberás tomar el día, mes y año por separado (subcadenas), con el objetivo de convertir cada fecha (String) en un objeto Calendar. Una vez ya tienes creado el objeto Calendar de una fecha, añadirlo a una lista de calendarios. Esta lista será una colección de tipo HashSet que almacenará objetos de tipo Calendar. Y así hasta que no queden fechas en el array de cadenas.

Cuando ya hayas insertado todas las fechas correctamente en el HashSet, el programa mostrará un número que será el de fechas únicas. Lo puedes obtener a partir del tamaño del HashSet que contiene estas fechas (ya que en un HashSet no puede haber elementos duplicados).

Por tanto, en este programa deberás usar la clase String, Calendar, ArrayList y HashSet.

Ejercicio 7. Implementa el control de acceso al área restringida de un programa.

Lo primero que pedirá el programa será usuario/contraseña. En caso de que sea el usuario administrador el sistema mostrará el mensaje “Ha accedido al área restringida como administrador” y mostrará el siguiente menú:

1. Registrar nuevo usuario
2. Listar usuarios del sistema
3. Eliminar usuario del sistema
4. Actualizar contraseña de usuario
5. Salir

En caso de que sea un usuario no administrador el sistema le mostrará “Ha accedido al área restrigida” y terminará el programa.

En cualquier caso, el usuario tendrá un máximo de 3 oportunidades. Si se agotan las oportunidades el programa dirá “Lo siento, no tiene acceso al área restringida”.

El usuario (admin) y contraseña (1234) del administrador se debe de introducir en el código fuente del programa. Cada una de las opciones del menú del administrador:

- **Registrar nuevo usuario:** pedirá usuario y contraseña nuevos. En caso de que ya exista el usuario el sistema debe advertirlo. La contraseña debe ser numérica y no puede tener más de 4 caracteres.
- **Listar usuarios:** muestra los usuarios que hay en el sistema, pero no sus contraseñas. Por ejemplo, en caso de tener los usuario usu1, usu2 y usu3, el formato de salida será: (usu1:usu2:usu3)
- **Eliminar usuario:** el sistema pedirá el nombre del usuario y deberá confirmar si se ha podido eliminar del sistema.
- **Actualizar usuario:** el sistema pedirá el nombre de un usuario del sistema y contraseña. En caso de no existir el usuario lo advertirá.