UF2404 - Ejercicios sobre la Estructura de la Información y Gestión de Memoria

A continuación se presentan una serie de ejercicios divididos en distintas secciones para practicar conceptos clave de la estructura de la información y la gestión de memoria en Java. Cada ejercicio está diseñado para abordar un tema específico de la lista proporcionada. Resuelve los ejercicios en el orden que prefieras y trata de escribir el código de cada uno en tu entorno de desarrollo.

Ejercicio 1: Datos Simples

1. Declara variables en Java para los siguientes tipos de datos simples:
    - Un número entero (int) con valor 100.
    - Un número real (double) con valor 3.14.
    - Un valor lógico (boolean) que represente verdadero.
    - Un carácter (char) que contenga la letra 'A'.
    - Una cadena de caracteres (String) con el texto "Hola Mundo".
    - Declara una referencia (puntero) a un objeto `String` y asigna una nueva cadena "Referencia de Memoria".
2. Imprime los valores de todas las variables en la consola.

Ejercicio 2: Arrays

1. Declara un array de números enteros con tamaño 5 y asigna los valores 10, 20, 30, 40 y 50.
2. Imprime cada valor del array utilizando un bucle `for`.
3. Crea un array bidimensional de enteros de tamaño 2x2 e inicializa sus elementos con los valores de tu elección. Imprime sus valores.

Ejercicio 3: Listas Enlazadas, Pilas y Colas

1. Crea una **lista enlazada** de objetos `String` y añade tres nombres de personas. Luego, imprime cada nombre en la consola.
2. Implementa una **pila** utilizando la clase `Stack` de Java. Añade tres números a la pila, realiza una operación de "pop" y luego muestra el número en la parte superior de la pila.

3. Implementa una **cola** utilizando la clase `Queue` (por ejemplo, con `LinkedList`). Añade tres cadenas de texto a la cola, elimina el primer elemento y muestra el siguiente.

Ejercicio 4: Estructuras

1. Crea una clase `Coche` que contenga los siguientes atributos: `marca` (String), `modelo` (String), `año` (int).
2. Implementa un constructor que inicialice estos valores.
3. Escribe métodos `getter` y `setter` para cada atributo.
4. Crea un objeto de la clase `Coche`, inicialízalo con tus valores, y muestra la información del coche en la consola.

Ejercicio 5: Ficheros

1. Crea un programa que cree un fichero de texto llamado `datos.txt` y escribe en él la cadena "Este es un archivo de prueba".
2. Lee el contenido del fichero y muéstralo en la consola.
3. Asegúrate de manejar posibles excepciones relacionadas con la lectura y escritura de archivos.

Ejercicio 6: Otras Estructuras Complejas

1. Implementa una **tabla hash** utilizando `HashMap` en Java. Añade tres pares clave-valor, donde las claves sean nombres de personas y los valores sean sus edades. Imprime todos los pares clave-valor en la consola.
2. Crea un **árbol binario** básico. Implementa una clase `Nodo` con un valor entero y referencias a nodos izquierdo y derecho. Inserta manualmente valores en el árbol y luego imprime el valor de la raíz, el nodo izquierdo y el nodo derecho.
3. Crea un **grafo** utilizando una lista de adyacencia en Java. Añade cuatro nodos y algunas aristas entre ellos. Implementa un recorrido BFS (búsqueda en amplitud) y muestra el orden de los nodos visitados.

Ejercicio 7: Mecanismos de Gestión de Memoria

1. Crea una clase `Persona` que tenga un constructor que imprima "Objeto creado" cuando se construya un objeto.

2. Implementa un método `finalize()` que imprima "Recolección de basura" cuando el objeto sea destruido por el Garbage Collector.
3. Crea una instancia de la clase `Persona` y luego establece la referencia a `null`. Llama a `System.gc()` para sugerir al Garbage Collector que recoja el objeto. Imprime un mensaje indicando cuándo se ha terminado de ejecutar el programa.
4. Explica qué ocurre con los objetos inalcanzables y cómo funciona la recolección de basura en Java.

Ejercicio 8: Uso de Constructores y Destructores

1. Modifica la clase `Persona` para añadir un constructor que acepte un nombre y una edad como parámetros, e imprima esos valores al crear un objeto.
2. Crea varios objetos `Persona` en el método `main()` y muestra sus valores.
3. Explica por qué Java no tiene destructores explícitos y cómo la recolección de basura se encarga de liberar los recursos cuando un objeto ya no es accesible.