Programación Orientada a Objetos

Práctica Nº 5 – 2019

Para cada una de los siguientes problemas plantee una solución identificando los **objetos que intervienen**, su **estado**, su **comportamiento** y las **relaciones** entre ellos.

1. ***Centro de Cómputos***
   Un centro de cómputos se encarga de ejecutar procesos utilizando algunas de las computadoras que dispone. Si no hay computadoras disponibles los procesos a ejecutar deben esperar en una cola de espera que los ordena teniendo en cuenta sus requerimientos de memoria (los procesos con mayor requerimiento de memoria serán atendidos en primer lugar). Las computadoras disponibles para ejecutar procesos se ordenan en una cola que prioriza la selección de las computadoras más rápidas.

2. ***Puerto de Cereales***
   Un puerto debe coordinar las actividades de carga de los barcos con cereal traído por camiones desde el campo. Sólo se puede cargar un barco a la vez. Los barcos que esperan ser cargados se ubican en una cola de espera que los ordena teniendo en cuenta su capacidad (los barcos con mayor capacidad serán atendidos en primer lugar). Sólo se puede descargar un camión a la vez. Los camiones que esperan ser descargados se ubican en una cola de espera que los ordena teniendo en cuenta la fecha en que fueron cargados (los camiones que fueron cargados primero serán atendidos en primer lugar)

3. ***Abstracción***
   Abstraer el comportamiento en común de los ejercicios 1 y 2 e implementar la solución. Implementar como quedan las dos versiones utilizando la abstracción.

4. ***Vocabulario***
   Se desea llevar la cuenta de las estadísticas del vocabulario de un texto. El constructor de esta clase recibe como parámetro un String y crea los objetos necesarios para saber qué palabras aparecen en el mismo y cuántas veces. Se debe permitir:

   1. Conocer la cantidad de palabras diferentes que contiene el texto.
   2. Recorrer las palabras alfabéticamente.
   3. Recorrer las palabras por frecuencia de ocurrencia.
   4. Retornar las N palabras más frecuentes.
   5. Retornar las N palabras menos frecuentes.
   5. Obtener la frecuencia de ocurrencia de una palabra.

6. ***Pila***
   Implementar en Java una colección para almacenar una pila de elementos. A esta colección se le pueden agregar elementos utilizando el método `push(Object o)`. Para retirar elementos de la colección se utiliza el método `pop()`, que retorna el último elemento agregado a la colección y lo elimina de la misma. Es posible consultar el tope de la pila sin eliminarlo utilizando el método `top()`. La mencionada anteriormente es la única forma de consultar y retirar elementos de la colección, es decir, no se pueden obtener ni consultar elementos de otra posición que no sea el tope de la pila. Definir también un método para conocer el tamaño de la pila.

Programación Orientada a Objetos

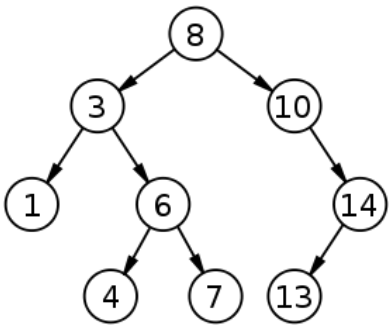
Práctica Nº 5 – 2019

7. Mazo de cartas

Implementar un juego de cartas en el cual se dispone de un mazo y se reparten, en cada ronda, dos cartas por jugador. Un jugador gana la mano cuando la suma de sus cartas es mayor que la de su oponente. El jugador que gana coloca todas las cartas (las suyas y las del oponente) en su pila de cartas ganadas. En caso de empate, se reparte una carta más a cada jugador hasta que uno de los dos gane. Implementar los mazos como pilas de cartas.

8. ***Árbol binario de búsqueda***

Un árbol binario es una estructura de datos formada por nodos que contienen un determinado valor. El primer elemento agregado a la estructura se conoce con el nombre de "raíz" y es el único punto de acceso a la misma. Cada nodo, puede tener un nodo "hijo" a su izquierda y un nodo hijo a su derecha cumpliendo con la restricción que los valores a su izquierda son valores menores que su propio valor, y los valores a su derecha son valores mayores (no se almacenan valores repetidos). Los nodos sin hijos se conocen como "hojas". Normalmente, para facilitar el recorrido de la estructura, cada nodo tiene una referencia a su nodo "padre".



a) Implementar la funcionalidad para agregar un nuevo objeto a la estructura. Para poder trabajar con cualquier objeto es necesario que el mismo pueda ser comparable, es decir, implementar la interfaz `Comparable` de Java.

b) Implementar un método que permita recorrer la estructura en orden, es decir, todos los elementos a la izquierda, luego la raíz y después todos los elementos a la derecha. Al recorrer los elementos es necesario que se defina una acción que se va a ejecutar con cada nodo visitado. Para poder trabajar de forma transparente y que se pueda extender la funcionalidad, definir una interface `AccionEjecutable`.

c) Crear una acción que permita incorporar los elementos de forma ordenada a un Vector.

d) Crear una acción que los agregue de forma ordenada inversa, en todo momento del proceso del recorrido el menor elemento visitado es el último del Vector.

e) Crear una acción que cuente la cantidad de elementos visitados.