Ejercicio 1

Implemente una clase llamada CajaDeAhorro, cuyo estado interno esté representado por un número de caja de ahorro y un saldo monetario. Se debeen proveer operaciones para averiguar el saldo y para realizar depósitos y extracciones.

Ejercicio 2

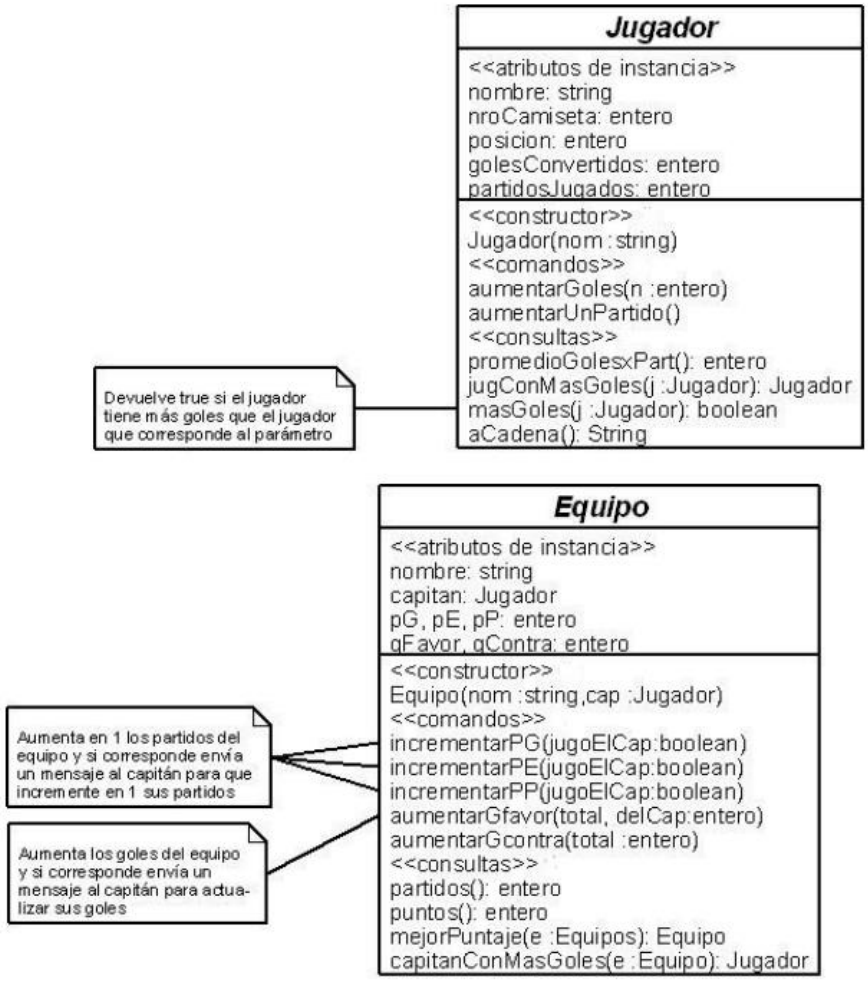
Implemente una clase Java llamada Flor que tenga tres atributos de tipo String, int y float, que respectivamente representen el nombre de la flor, el número de pétalos y precio. Se debe incluir un constructor que permita inicializar cada uno de los atributos y un constructor que inicialice la flor por defecto con nombre Margarita, número de pétalos igual a 10 y precio igual a 34.3. También se deben proveer operaciones para modificar los atributos y obtener sus valores en

forma individual.

Ejercicio 3

En un campeonato de fútbol por cada partido ganado se obtienen 3 puntos y por cada empate se logra 1. Cada equipo tiene un nombre, un capitán, una cantidad de partidos ganados, otra de empatados y otra de perdidos, una cantidad de goles a favor y otra de goles en contra. El capitán es un jugador que tiene un nombre, un año de nacimiento, un número de camiseta, un número que representa la posición en la que juega, la cantidad de partidos jugados y la cantidad de goles convertidos en el campeonato.

Para un jugador se desea calcular el promedio de goles de un jugador por partido y dado otro jugador, cuál es el que hizo más goles. Para un equipo se desea calcular los partidos jugados y los puntos obtenidos. Además, para otro equipo dado es necesario decidir cuál es el equipo con mayor puntaje y cuál es el capitán con más goles. Si dos equipos tienen los mismos puntos, se devuelve el que tiene mayor cantidad de goles a favor y si también hay coincidencia se consideran los goles en contra. Si hay coincidencia se devuelve uno cualquiera.



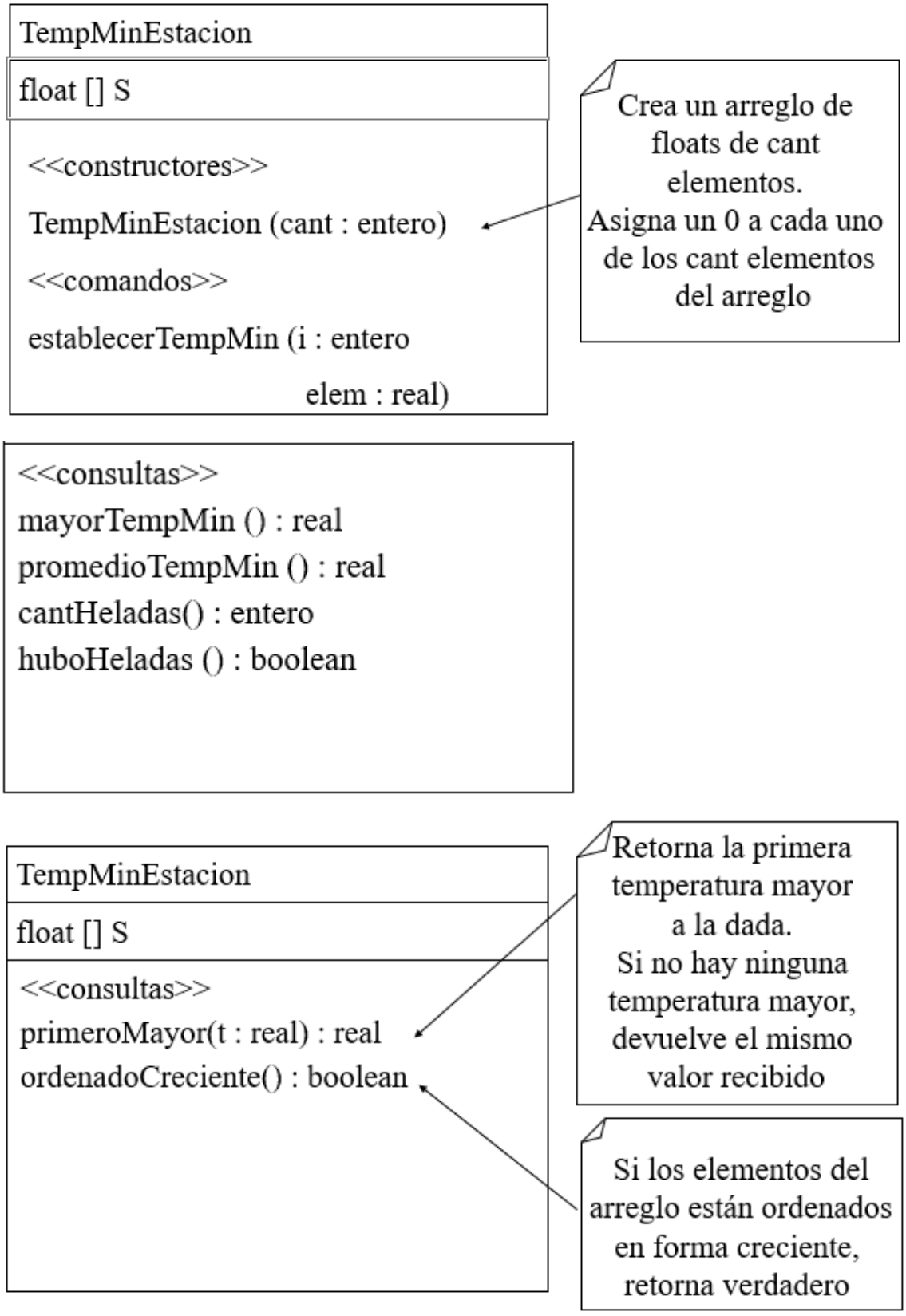
1. A partir del diagrama de clases en UML y la especificación de requerimientos implemente dos clases en Java encapsulando atributos y comportamiento para modelar la situación planteada. Incluya dentro del comportamiento los comandos para modificar cada atributo y las consultas para devolver cada atributo.
2. Elabore una clase que permita testear el modelo.

Ejercicio 4

Defina una clase TempMinEstacion que encapsule la representación de las temperaturas mínimas registradas en una estación meteorológica y brinde operaciones para:

- Retornar la mayor temperatura registrada
- Calcular el promedio de las temperaturas
- Contar cuántos días heló
- Decidir si hubo heladas

- Retornar la primera temperatura mayor a una dada; si no existe devolver el mismo valor recibido.
- Decidir si las temperaturas registradas se produjeron en orden creciente



Ejercicio 5

Extienda la definición de la clase TemperaturasEstacion presentada en clase

incorporando los siguientes servicios:

- float menorTemperatura(): Retornar la menor temperatura registrada
- int posMenorTemperatura(): Retornar la posición de la menor temperatura
- boolean todosPositivos(): Determinar si todas las temperaturas son positivas
- int contarCoincidencias(TemperaturasEstacion s): Calcular en cuántos días dos estaciones meteorológicas coincidieron en la temperatura mínima, es decir cuántas componentes coinciden en valor y posición respecto a un objeto de la misma clase recibido como parámetro. Se asume que se han registrado los valores de una misma cantidad de días.
- TemperaturasEstacion invertir(): Generar un objeto de clase TemperaturasEstacion con las mismas componentes que la tabla que recibe el mensaje, pero almacenadas en orden inverso.
- void reemplazar(float val1,val2): Reemplazar toda aparición del valor val1 por val2
- String intercambiar(int pos1, pos2): Intercambiar las componentes de las posiciones pos1 y pos2, verificando que ambas existan. Si pudo hacer el intercambio devuelve una cadenavacía, sino un mensaje de error.
- void invertirMe(): invierte los elementos de la tabla receptora del mensaje
- **OPCIONAL** void ordenar(): Ordena las temperaturas de mayor a menor. Investigar métodos de ordenamiento Bubble Sort, Quick Sort, Merge Sort y Búsqueda Binaria (implementar este último).

Ejercicio 6

Implementar los ejercicios 4 y 5 con LinkedList en lugar de arreglos.

Ejercicio 7

Implementar el TDA LinkedList pero guardando una referencia al final de la lista (tail).

Ejercicio 8

Implementar el TDA DLinkedList donde cada Nodo tiene un puntero también al anterior.