Ejercicios: Programación orientada a objetos

Nota: en cada ejercicio puede crear atributos, métodos y clases auxiliares en caso lo considere necesario.

1. Crear una clase llamada ListaEnlazada que contenga una lista enlazada y los siguientes métodos:
    - agregarElemento: agrega un elemento en la posición indicada.
    - eliminarElemento: elimina un elemento en la posición indicada.
    - verValor: retorna el valor de la posición indicada.
    - cantidadElementos: muestra la cantidad de elementos que posee la lista enlazada.

2. Crear una clase llamada Pila que contenga una lista y posea solo los siguientes métodos:
    - push: agrega un elemento al final de la lista.
    - peek: muestra el último elemento de la lista.
    - pop: elimina el último elemento de la lista.

    La lista de la Pila debe ser privada, es decir no se podrá acceder a ella fuera de la clase.

3. Crear una clase llamada Cola que contenga una lista y posea solo los siguientes métodos:
    - enqueue: agrega un elemento al final de la lista.
    - first: muestra el primer elemento de la lista.
    - dequeue: elimina el primer elemento de la lista.

    La lista de la Cola debe ser privada, es decir no se podrá acceder a ella fuera de la clase.

4. A partir de la clase Cola, crear una clase hija llamada ColaDePrioridad la cual deberá mantener sus elementos siempre en orden ascendente.

5. A partir de la clase Cola, crear una clase hija llamada LazyColaDePrioridad la cual deberá ordenar sus elementos en orden ascendente solo cuando se desee mostrar o eliminar un elemento.

6. Crear una clase llamada HojaCalculo que represente una hoja de calculo similar a la de un excel. Para esto la clase deberá contener una lista de filas, cada fila debe contener una lista de celdas. Todas las filas de la hoja de calculo deben contener la misma cantidad de celdas. HojaCalculo debe contener los siguientes métodos:
    - agregarFila: agrega una fila a la hoja de cálculo.
    - agregarColumna: agrega una columna a la hoja de cálculo, es decir agrega una celda a todas las filas.
    - verValor: retorna el valor ubicado en la fila y columna indicada.
    - asignarValor: asigna el valor ingresado en la celda indicada.
    - mostrarHoja: muestra la hoja de cálculo en la consola.

    Recordar qué puede crear las clases auxiliares que considere necesarias.

7. Crear una clase llamada Caja que contenga los siguientes atributos:
    - capacidad: volumen que puede caber en la caja en cm3
    - volumen: volumen de la caja
    - contenido: lista de objetos (cajas o productos)
    - peso: peso de la caja

   Y los siguientes métodos:
    - agregarObjeto: agrega un objeto a la lista de objetos en caso la caja aun tenga capacidad, en caso contrario debe verificar si alguna caja de la lista posee el espacio suficiente para guardar el objeto. En caso ninguna caja o sub caja pueda guardar el objeto imprime en consola que no hay espacio.
    - calcularPeso: retorna el peso de la caja.

   La clase Producto debe contener los siguientes atributos:
    - volumen: volumen del producto
    - peso: peso del producto

8. Agregar a la clase HojaCalculo los siguientes métodos:
    - asignarValorSuma: asigna a la celda indicada la suma de los valores de otras dos celdas indicadas. Si el valor de alguna de esas dos celdas cambia, la suma se deberá actualizar.
    - asignarValorResta: asigna a la celda indicada la resta de los valores de otras dos celdas indicadas. Si el valor de alguna de esas dos celdas cambia, la resta se deberá actualizar.

   Modificar las clases que considere necesarias.

9. Crear una clase llamada Función que contenga un atributo llamado nombre y una lista de parámetros, un parámetro puede ser un valor o un objeto de tipo Función. La clase debe poseer los siguientes métodos:
    - formarLlamadaPython: retorna un string con la llamada a la función hecha con la sintaxis de Python.
    - formarLlamadaRacket: retorna un string con la llamada a la función hecha con la sintaxis de Racket.

10. Agregar un constructor a la clase Función que reciba como parámetro un string que contenga una llamada a una función con la sintaxis de Racket, extraiga de este string el nombre y los parámetros de la función, y asigne estos datos a los atributos nombre y parámetros respectivamente.

11. Crear una clase PartidaAjedrez que represente una partida de ajedrez y contenga los siguientes métodos:
    - verTurno: muestra en la consola que color de piezas se deben mover en este turno.
    - mostrarPartida: muestra en la consola el tablero y las fichas en su respectiva ubicación. Imprimir también los ejes del tablero.

- mostrarPosiblesMovimientos: muestra en consola las casillas a las que se podría mover la pieza de la casilla indicada. En caso no haya una pieza en dicha casilla indicarlo en la consola.
- moverFicha: recibe de parámetros las coordenadas de la ficha a mover y las coordenadas del destino. Mueve la ficha a la casilla indicada en caso sea posible, en caso contrario muestra en la consola que la jugada es incorrecta.
- reiniciarPartida: devuelve todos los valores a su estado inicial.

Sugerencia: crear una clase abstracta que representa a las piezas y luego crear clases hija a partir de esta para cada una de las piezas en particular (peón, caballo, etc)

12. Modificar la clase PartidaAjedrez para que muestre en consola el resultado de la partida cuando esta acabe. Puede modificar las clases, métodos y atributos que considere necesarios.