\*Dudas Hoy\*

1. por ejemplo las clases abstractas se deben crear siempre o no necesariamente, por ejemplo para producto es una clase abstracta por que no se instancia no crea objeto pero a partir de esa clase existen clases hijas como pan amasado pan pita, y otro que se podrán crear en el futuro y hereda clase

Dudas hoy

1. Por ejemplo en el proceso descontar stock como habilito una compra efectiva porque esta será posible si y solo si hay stock disponible. Por ejemplo, si para el día martes todos reservaron un stock de 100 panes y al momento de publicar el día martes ya no hay mas stock, como permito o no permito la compra para ese día? Eso se pone en diagramas?.
2. De una decisión deben salir las líneas de un mismo punto y ahí separarse en si o en no o pueden salir de 2 puntos con el si por uno y el no por otro?
3. En los diagramas de flujo debo poner los return o eso es en el de clases. Por ejemplo en el proceso de reservar stock voy a tener fechas de reserva y al final del proceso de reserva voy a tener una cantidad reservada para cada fecha, esto lo va a utilizar el proceso o la función de descontar stock, que va a ver la fecha y de acuerdo a ella si son iguales va a proceder a descontar el stock reservado y una vez hecho eso si queda aun stock se procede a la compra efectiva.
4. Qué son los receptores? Y como se diagraman? Son diferentes a los puertos?
5. Si hay herencia puede existir polimorfismo? Sin herencia no se puede aplicar polimorfismo??
6. Entiendo el encapsulamiento, pero como se aplica en código como “oculto” la estructura o soporte de mi código. Ejemplo la fecha? En las funciones hay encapsulamiento, no se puede acceder a la variables fuera de ellas.

El jueves cuando creía que tenia claro como comenzar los diagramas de clases me puse a ver mas info respecto a como pasarlo al código,

1. “Todos los atributos van a tener getters y setters” Se debe utilizar getters y setters para pasar de diagrama de clases a código, es una buena práctica? Los Setters y Getters son métodos de acceso a los campos/atributos de una clase.   
   **Setters**: Del Inglés Set, que significa establecer, sirven para asignar un valor a un campo/atributo. Si lo dejo abierto lo modifican.  
   **Getters**: Del Inglés Get, que significa obtener, sirven para recuperar el valor de un campo/atributo. Se puede leer pero no se puede modificar.

**Violar el Principio de Encapsulamiento debate 2003)**Un objeto puede ser destruido por otros objetos, dado que estos últimos pueden infectar información al primero, a través de setters. El objeto no puede encapsular su propio estado de manera segura, debido a que cualquiera lo puede alterar.

<https://arcanefoam.medium.com/getters-setters-perversos-punto-fe889b690332>

Esto es valido se esta cuestionando a que se refiere o es un terraplanista?