**Pregunta 1:**

Un miembro de instancia puede ser un atributo o método que pertenece solamente a un objeto (instancia de una clase), mientras que los miembros de clases corresponden a aquellos atributos o métodos que son compartidos entre todos los objetos de una misma clase; además los miembros de clases son almacenados en una ubicación de memoria fija común para todos los miembros de esta clase, en cambio los miembros de instancias son almacenados en diferentes ubicaciones.

**Pregunta 2:**

Las agrupaciones lógicas sirven para agrupar ordenadamente clases o interfaces que cumplen una función parecida, de esta manera es más fácil acceder a aspectos específicos de nuestro programa (modelos, vista, controladores) y a la vez son protegidos de otras agrupaciones, ya que, para utilizar la agrupación esta debe ser anteriormente importada.

La manera de crear una agrupación lógica en java es escribir al comienzo del documento la palabra “package ” seguido del nombre del paquete al que pertenece, en este caso será el nombre que le dará a este nuevo paquete, además los archivos de las clases que pertenecen a este paquete deben estar almacenados en una misma carpeta que debe tener el nombre del paquete.

Ejemplo: package models;

**Pregunta 3:**

Este concepto hace referencia a que una interfaz define la estructura “externa” para un método que implementarán las clases que utilicen esta interfaz, en otras palabras, en una interfaz establecemos solamente el modificador de acceso, el retorno, el nombre y los argumentos de un método, la manera en que se utilizaran estos, es definida dentro de cada clase que utilice esta interfaz.

En el siguiente ejemplo podemos ver como se crea la interfaz saludable, haciendo referencia a alguien o algo que podemos saludar y una vez dentro de la clase Persona, definimos el método saludar diciendo “Hola”, mientras que, en la clase Perro, saludamos diciendo “Guau guau”

public interface Saludable {

public void saludar();

}

public class Persona {

protected String nombre;

protected int edad;

public void saludar() {

System.out.println("Hola ");

}

}

public class Perro {

protected String nombre;

protected String raza;

public void saludar() {

System.out.println("Guau guau");

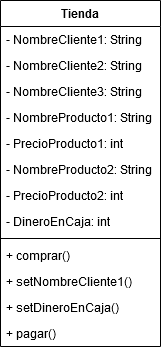
}

}

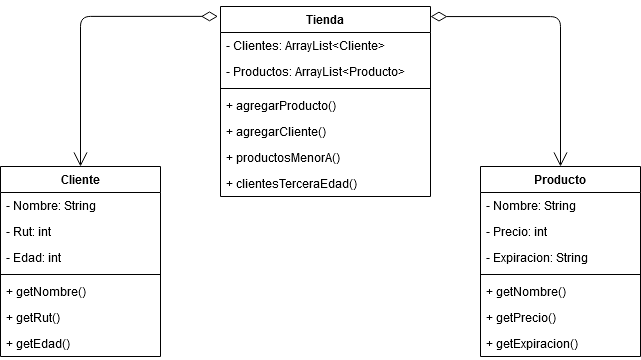
**Pregunta 4:**

La cohesión representa el nivel en el que cada módulo de nuestro programa haga referencia a un proceso o entidad en específico, mientras más específico sean nuestros módulos, mayor cohesión tendrá nuestro programa.

En el primer diagrama podemos observar baja cohesión, ya que la clase tienda contiene todos los elementos que la componen y sus métodos lo hacen todo, esto quiere decir que los procesos no están separados en entidades separadas, por lo que cada elemento será modificado desde la clase tienda.



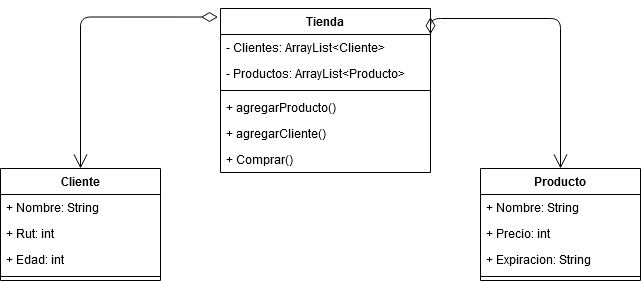
En este segundo diagrama podemos notar una alta cohesión, ya que cada entidad está separada, de esta manera cada componente de la tienda está separado con sus respectivos atributos y métodos, así si queremos modificar un componente, utilizamos los métodos de su correspondiente clase.



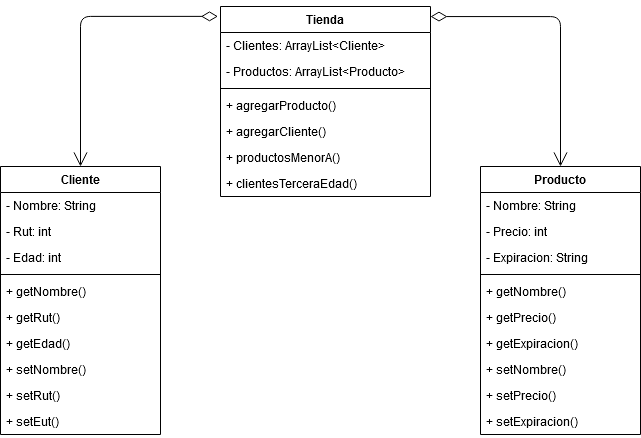
**Pregunta 5:**

El acoplamiento hace referencia al nivel de interdependencia de los módulos que componen nuestro programa, lo ideal es que una clase sepa de otra solamente lo necesario, si modificamos algo en una clase, no debe perjudicar el funcionamiento de otra clase.

En este diagrama podemos ver un ejemplo de alto acoplamiento, ya que las clases Cliente y Producto no poseen métodos getter y setter, por lo que, si queremos modificar o acceder a un atributo de la clase debemos hacerlo “manualmente” (tienda.productos.get(0)……), esto provocará que, si cambiamos la estructura de la clase Cliente o Producto, también tendremos que cambiar la manera en la que accedemos a cada elemento.



En el segundo diagrama podemos notar que tanto la clase Cliente como Producto, poseen getter y setter para cada uno de sus atributos, de esta manera si modificamos la estructura de estas clases, solamente tendremos que modificar el getter y setter de dicho atributo dentro de la clase, no tendremos que cambiar el código que, por ejemplo, se utiliza en la clase Tienda para sus métodos.



**Pregunta 6:**

1. Dentro del diagrama están presentes los conceptos de clases, agregación, herencia, atributos y métodos públicos.
2. La sobreescritura la podemos utilizar para métodos de las clases Armadura con sus clases hijas o Arma con sus clases hijas: equipar, desequipar, descartar, estadísticas, etc.

Un ejemplo para sobrecarga podría ser un constructor para la clase personaje, de esta manera podemos tener un constructor que solamente reciba un nombre, de esta manera crearíamos un nuevo personaje por ejemplo en nivel 1 y a la vez tener un constructor que reciba como argumentos el nombre y su nivel, de esta manera podremos crear un personaje con el nivel que queramos.

1. Para ilustrar el concepto de polimorfismo podríamos crear clases hijas de la clase personaje, tales como mago, arquero, luchador, sanador; de esta manera cuando utilicemos el método atacar, la ejecución de este dependerá del objeto que este reciba y así por ejemplo si utilizamos el método atacar con la clase Personaje este atacará con el arma equipada, si lo utilizamos con un mago este lanzará un hechizo o con un sanador este curara a otro personaje.