**Java War: ¡Desata la masacre de vehículos de guerra!**

Queremos simular una “guerra” en Java, donde cada **Vehículo de Guerra** puede atacar y defenderse siguiendo ciertas reglas, y además cuenta con un conjunto de **Guerreros** que participan en el ataque y la defensa. Para ello, deberás diseñar un conjunto de clases e interfaces que representen tanto la lógica de combate como los atributos específicos de cada tipo de vehículo.

Diagrama

El contenido generado por IA puede ser incorrecto.

1. **Interfaz Tripulable**
   * **iAtacar**: declara el método int atacar().
   * **iDefender**: declara el método int defender(int).
2. **Clase padre: VehiculoGuerra (No se pueden crear objetos)**
   * Atributos sugeridos (puedes agregar otros que consideres necesarios):
     + Puntos de Vida (int).
     + nombre (String): nombre del vehículo.
     + tipo (String): tipo o categoría del vehículo (por ejemplo, “Tanque”, “NaveDestructora”).
     + fuerzaResistencia (int): nivel de resistencia o puntos de vida del vehículo.
     + listaGuerreros (colección de objetos “Guerrero”): guerreros que viajan en el vehículo.
   * Métodos sugeridos:
     + **Constructor/es**: debe validar si la suma de fuera y resistencia son mayores de 10).
   * Método embarcarGuerrero() Añadirá nuevos guerreros a la lista de guerreros de la nave. NO puede haber más de 10 guerreros en una nave
   * **Consideraciones**:
     + Los puntos de ataque y defensa por defecto son de 5 si se asignaran otros valores la suma de ambos no puede mayor de 10 si fuera así se quedaría con los valores por defecto.
     + Los puntos de vida serán de 1000 para todas las naves
3. **Clase padre Guerrero (no se pueden generar objetos de esta clase)**
   * Atributos sugeridos:
     + nombre (String): nombre o identificador del guerrero.
     + tipo (String): tipo de guerrero (infantería, artillero, piloto, etc.).
     + fuerza (int): fuerza base del guerrero
     + resistencia (int): resistencia base del guerrero.
   * Comportamiento sugerido:
     + Cada guerrero aporta puntos extra de ataque y/o defensa al vehículo.
     + La suma de la fuerza y la resistencia de un guerrero no puede ser mayor de 10
4. **Clases hijas de Vehículos** (ejemplos de implementación de las clases bases):
   * **Tanque**
     + Atributos iniciales recomendados:
       - puntosVida = 1000
       - ataque = 7
       - defensa = 3
     + Implementa las interfaces iAtacar e iDefender.
     + **Métodos**:
       - atacar(): devuelve un valor calculado con la fórmula del diagrama (p. ej. ataque base + suma del ataque de los guerreros).
       - defender(): devuelve un valor calculado con la fórmula del diagrama (p. ej. defensa base + suma de la defensa de los guerreros). Con el valor devuelto por el método defender se quitarán los puntos de vida de la nave
   * **NaveDestructora**
     + Atributos iniciales recomendados:
       - puntosVida = 1000
       - ataque = 2
       - defensa = 8
     + Implementa las interfaces iAtacar e iDefender.
     + **Métodos**:
       - atacar(): similar a Tanque, pero con su propia lógica de ataque (p. ej. ataque base + bonificación de guerreros).
       - defender(): lógica de defensa particular.
5. **Fórmulas de ataque y defensa** (según se observa en el diagrama):
   * **Ataque**: ataque del vehiculo×random(0,1) + ∑ ( ataques de los guerreros)×random(0,0.5)
   * **Defensa**: defensa del vehículo × random (0,1) + ∑ (defensas de los guerreros) ×random(0,0.5)
   * Se puede incluir cualquier variación o ampliación de las formulas
   * Se asume que random(a, b) devuelve un número aleatorio entre a y b. Puedes usar Math.random() u otra función de tu preferencia.
6. **Requerimientos del ejercicio**
   * Implementar las clases e interfaces descritas.
   * En la clase principal (o en un método de prueba), crear al menos un objeto Tanque y un objeto NaveDestructora o cualquier otra creada
   * Crear guerreros con distintos valores de fuerza resistencia.
   * Embarcar a los guerreros en sus naves correspondientes (Ejemplo: Un soldado solo podría embarcar en un tanque)
   * Simular un combates donde cada vehículo invoque sus métodos atacar() y defender() en orden, primero un vehículo ataca y el otro se defiende y luego al revés así hasta que una de las dos naves se quede sin puntos de vida
   * Mostrar por consola los resultados de cada ataque y defensa, y actualizar puntos de vida del vehículo atacado en función del daño recibido (ataque – defensa).

Ejemplo:

* + - Empieza la guerra
    - El vehiculo1 ataca con x puntos
    - El vehículo 2 se defiende con y puntos
    - El vehículo 2 pierde z puntos
    - …
    - El vehículo 2 se queda sin puntos
    - El vehículo1 gana
  + Generar una librería con el proyecto y permitir hacer una llamada a un método del proyecto donde se le pase un Vehículo de Guerra y se inicie una batalla con un vehículo interno.
  + El proyecto se desarrollará en grupos que integrarán todo el código en Github

**Objetivo final**  
El objetivo es crear una estructura orientada a objetos que permita simular combates entre diferentes vehículos de guerra, cada uno con sus propias características y acompañados por guerreros que añaden variaciones a sus capacidades de ataque y defensa. Deberás utilizar correctamente la herencia, la implementación de interfaces y el manejo de excepciones para modelar la lógica de la guerra de vehículos.