EXAMEN (2h 30min) — Programación

"MiniLiga de Criaturas" (POO: herencia, abstractas, interfaces y polimorfismo)

Contexto

Vas a programar un juego simple de "criaturas" que luchan por turnos. Cada criatura tiene un **tipo elemental**, pero aquí el tipo será un **String** con uno de estos valores (en MAYÚSCULAS):

- `"FUEGO"`, `"AGUA"`, `"PLANTA"`, `"ELECTRICO"`

El tipo afecta al daño con un multiplicador (x2 fuerte, x0.5 débil, x1 normal).

Tabla de efectividad (String vs String)

- `"FUEGO"` → fuerte vs `"PLANTA"` (x2), débil vs `"AGUA"` (x0.5)
- `"AGUA"` → fuerte vs `"FUEGO"` (x2), débil vs `"PLANTA"` (x0.5)
- `"PLANTA"` → fuerte vs `"AGUA"` (x2), débil vs `"FUEGO"` (x0.5)
- `"ELECTRICO"` → fuerte vs `"AGUA"` (x2), débil vs `"PLANTA"` (x0.5)
- Resto: x1

Calificación (total 10 puntos)

Parte 1 — Clase base `Criatura` (2,5 puntos)

Requisitos

Crear clase `Criatura` con **atributos privados**:

- `nombre` (String)
- `nivel` (int)
- `psMax` (int)
- `psActual` (int)
- `ataque` (int)
- `defensa` (int)

Constructor con campos mínimos.

Métodos obligatorios:

- `boolean estaDebilitada()` → true si `psActual <= 0`
- `void curarCompleto()` → `psActual = psMax`
- `void recibirDanio(int cantidad)` → resta, sin bajar de 0

- `String estado()` → devuelve:
  `"Pika [ELECTRICO] Nv.5 PS 18/18 ATK 7 DEF 4"`

Métodos:

- `String getTipo()` → devuelve `"FUEGO"`, `"AGUA"`, etc.
- `int atacar(Criatura objetivo)` → devuelve daño infligido

Ejemplo

Si `getTipo()` devuelve `"AGUA"`, entonces `estado()` debe contener `"[AGUA]"`.

Puntuación Parte 1 (2,5)

- (1,0) atributos privados + constructor + getters coherentes
- (0,5) métodos `curarCompleto`, `recibirDanio`, `estaDebilitada` correctos
- (0,5) `estado()` correcto con el tipo (String)
- (0,5) abstractos bien definidos
- Se valora el uso de clases abstractas adecuadamente.

Parte 2 — Interfaces (1,5 puntos)

Crear:

- `interface Entrenable { void entrenar(); }`
- `interface Evolucionable { boolean puedeEvolucionar(); Criatura evolucionar(); }`

Regla mínima:

- `entrenar()` sube el nivel +1 y mejora stats
- Se puede evolucionar desde nivel **8**
- `evolucionar()` devuelve una **nueva** criatura (p.ej. nombre + `" Evo"`) con stats mejores

Ejemplo

- Criatura nivel 7: `puedeEvolucionar()` → false
- Tras `entrenar()`: nivel 8 → `puedeEvolucionar()` → true

Puntuación Parte 2 (1,5)

- (0,5) interfaces correctas
- (0,5) se usan realmente en clases hijas
- (0,5) lógica coherente (entreno y evolución)

Parte 3 — Subclases por tipo (4,0 puntos)

Crea 4 clases que heredan de `Criatura` e implementan `Entrenable` y `Evolucionable`:

- `FuegoMon`, `AguaMon`, `PlantaMon`, `ElectricoMon`

Cada una:

- `getTipo()` devuelve el String fijo: `"FUEGO"`, `"AGUA"`, etc.
- `atacar(objetivo)` aplica el daño con efectividad usando Strings

Regla de daño (obligatoria)

1. `dañoBase = ataque - objetivo.defensa/2`
2. si `dañoBase < 1` ⇒ `dañoBase = 1`
3. multiplicador por tipos (0.5, 1, 2)
4. `dañoFinal = round(dañoBase * multiplicador)`
5. aplicar `objetivo.recibirDanio(dañoFinal)`
6. devolver `dañoFinal`

IMPORTANTE: compara tipos con `.equals(...)` (no con ==).

Ejemplo de ataque

Atacante `"FUEGO"` ATK=10
Defensor `"PLANTA"` DEF=6, PS=20

- dañoBase=10-(6/2)=7
- mult=2
- dañoFinal=14
- defensor queda con PS=6

Puntuación Parte 3 (4,0)

- (1,0) herencia + `getTipo()` String correcto (las 4 clases)
- (1,5) `atacar()` aplica fórmula correcta y devuelve daño
- (1,0) efectividad correcta (x2/x0.5/x1) usando Strings
- (0,5) entrenar + evolucionar bien en las subclases

Parte 4 — Main con menú y combate (2,0 puntos)

En `Main`:

- `ArrayList<Criatura>`
- Menú:
    1. Crear criatura (nombre, tipo, nivel 1–10)
    2. Listar criaturas
    3. Combatir (elige atacante y defensor por índice, turnos hasta que una quede debilitada)
    4. Entrenar criatura
    5. Evolucionar criatura (sustituye en la lista)
    6. Curar criatura
    7. Salir

Validaciones mínimas:

- lista vacía
- índices inválidos
- no combatir si una está debilitada

Ejemplo de sesión

- Crear “Pika” tipo ELECTRICO nivel 5
- Crear “Aqua” tipo AGUA nivel 4
- Combatir: Pika inflige más daño (eléctrico fuerte vs agua)

Puntuación Parte 4 (2,0)

- (0,5) menú completo
- (1,0) combate por turnos y fin correcto
- (0,5) validaciones mínimas