

Examen — Gestor de Juegos (Herencia + Interfaz + List/Set/Map)

1) Modelo (Herencia)

- Clase abstracta: `Juego`
 - Propiedades (**no pueden ser nulas/vacías**):
 - `id: String` (único, no null/blank)
 - `titulo: String` (no null/blank)
 - `complemento: double` (porcentaje en rango **0..100**)
 - Método abstracto:
 - `String tipoJuego()` → devuelve `"ARCADE"` o `"MESA"` según la subclase
- Subclases:
 - `Arcade` extends `Juego`
 - `JuegoMesa` extends `Juego`

2) Interfaz

Implementa `TarificableJuego` en `Arcade` y `JuegoMesa` con:

- `double getComplemento()`
- `double precio()`

3) Cálculo de precios (muy importante)

Para calcular el precio usa esta fórmula:

- `baseArcade = 20.0`
- `baseMesa = 35.0`

Fórmula común:

```
precio = base + (base * complemento / 100)
```

Ejemplos:

- Arcade con 10% → $20 + 2 = 22$
- Mesa con 10% → $35 + 3.5 = 38.5$

4) Gestores de colecciones (CRUD + lógica)

Implementa 3 clases:

- `ListaJuegos` (usa `List`)

- `SetJuegos` (usa `Set`)
- `MapJuegos` (usa `Map<String, Juego>`)

En cada una debes implementar:

CRUD

- `boolean crearJuego(Juego juego)`
- `Juego buscarId(String id)`
- `Juego buscarJuego(Juego juego)` (busca por igualdad/equals)
- `boolean actualizarJuego(Juego juego)`
- `boolean eliminarId(String id)`
- `boolean existeJuego(String id)`

Lógica (3 funciones)

- `double calcularPrecio(String tipo)`
Suma el precio de todos los juegos cuyo `tipoJuego()` coincide con `tipo`.
- `double calcularPrecioTotal()`
Suma el precio de todos los juegos del gestor.
- `int contarPorTipo(String tipo)`
Cuenta cuántos juegos hay de un tipo.

5) Reglas de validación (EXCEPCIONES)

En TODOS los métodos anteriores, si recibes valores inválidos, debes lanzar:

- `IllegalArgumentException` si:
 - `id` es `null` o `blank`
 - `tipo` es `null` o `blank`
 - `juego` es `null`
 - `juego.getId()` es `null` o `blank`

6) Cómo se evalúa

La corrección se realiza con tests:

```
mvn test
```

Nota automática (orientativa): $(\text{testsPasados}/\text{testsTotales}) * 10$

¡Suerte!