# Análisis profundo del enunciado de la Práctica 1

Basado en el documento oficial de **Práctica 1 – IPC1 2026**

by: **kik3.h**

## ¿Qué evalúa realmente esta práctica?

Aunque parece “hacer jueguitos”, NO evalúa juegos, evalúa:

- Comprensión de flujo de control
- Capacidad de dividir un problema grande en subproblemas
- Uso correcto de:
    - Condicionales
    - Ciclos
    - Arreglos
    - Métodos
- Lógica **sin ayudas avanzadas**
- Orden, claridad y legibilidad del código
- Capacidad de seguir instrucciones estrictas

En resumen:

> *“No es qué tan bonito queda, es qué tan bien pensaste el problema.”*

## Idea central del proyecto

Un programa de consola en Java que actúa como una librería de juegos, con:

- Menú principal
- 3 juegos independientes
- Reportes acumulados
- Flujo correcto entre menús
- Uso opcional de parámetro por consola

Nada gráfico, nada POO avanzada, nada estructuras dinámicas.

## Restricciones implícitas (muy importantes)

Aunque no lo dicen explícitamente, por el **nivel del curso** y por experiencia:

**NO deben usar**:

- `ArrayList`, `LinkedList`, `HashMap`, etc.
- `try-catch`
- Streams
- Archivos
- Clases complejas
- Herencia
- Interfaces
- Librerías externas
- Lógica “copiada” de internet/IA

**SÍ deben usar**:

- `Scanner`
- `Math.random()` o `Random`
- Arreglos simples (`int[]`, `String[]`)
- `if`, `else`, `switch`
- `for`, `while`, `do-while`
- Métodos bien definidos

Esto encaja perfectamente con el contenido del curso IPC1 y laboratorio

# 2. Desglose lógico de cada parte del proyecto

## Menú principal (columna vertebral)

Debe permitir:

1. Elegir juego

2. Ver reportes
3. Salir

Conceptos clave aquí:

- Ciclo infinito controlado (`while(opcion != salir)`)
- `switch` para opciones
- Validación de entrada (SIN try-catch)

---

# Inicio por parámetro (extra de nota)

> "Al ejecutar el programa desde la terminal…"

Esto evalúa:

- `main(String[] args)`
- `args.length`
- Comparación de `String`

No debe romper el flujo normal:

- Si el parámetro es inválido → menú
- Si no hay parámetro → menú

# Juego 1: Batalla

### Qué evalúa realmente:

- Uso de `char`
- Comparación sin importar mayúsculas
- Contadores acumulativos
- Lógica de niveles
- Ciclos con múltiples condiciones

### Variables clave:

- Vida jugador
- Vida computadora
- Contador de aciertos

- Contador de fallos
- Nivel
- Puntaje máximo alcanzado

Error típico del estudiante:

> Mezclar **lógica del juego** con **menús**

---

## Juego 2: Wordle

Este es el **más importante** conceptualmente.

### Evalúa:

- Arreglos
- Comparación carácter por carácter
- Ciclos anidados
- Persistencia de intentos
- Validación estricta

Puntos críticos:

- Palabra de exactamente 5 letras
- Máximo 6 intentos
- Mostrar intentos anteriores
- Colores (opcional, pero suma puntos)

Aquí NO se espera perfección, se espera:

- Lógica clara
- Que funcione siempre
- Que no se rompa

## Juego 3: Basketball

Evalúa:

- Simulación con probabilidades
- Uso correcto de `Math.random()`

- Condicionales encadenados
- Control de turnos

Importante:
No necesitan estadística avanzada, solo:

if (Math.random() <= 0.65) { ... }

# Reportes

Esto evalúa **memoria en ejecución**, no archivos.

- Variables globales
- Acumuladores
- Incrementos bien ubicados

Error clásico:

Incrementar contadores en el lugar incorrecto

# 3. Ruta recomendada para empezar desde CERO

Esta parte es CLAVE para la tutoría.

## Paso 1: Leer y subrayar el enunciado

Antes de escribir código:

- ¿Qué entra?
- ¿Qué sale?
- ¿Qué se repite?
- ¿Qué se guarda?

---

## Paso 2: Diseñar el menú en papel

Sin código:

- Menú principal
- Menú después de cada juego
- Flujo de regreso

---

## Paso 3: Esqueleto del programa

Solo métodos vacíos:

main()
menuPrincipal()
menuPostJuego()
juegoBatalla()
juegoWordle()
juegoBasketball()
reportes()

## Paso 4: Implementar UN juego primero

Recomendado:

1. Wordle (arreglos + ciclos)
2. Batalla

3. Basketball

---

## Paso 5: Agregar reportes

Cuando todo funcione.

---

## Paso 6: Limpiar y comentar

Aquí se gana nota:

- Nombres claros
- Métodos cortos
- Comentarios útiles

# 4. Buenas prácticas que SÍ aplican en IPC1

- Métodos pequeños
  Nombres descriptivos
  Evitar código duplicado
- Usar constantes
  No abusar de variables globales
  Un `Scanner` bien controlado

---

# 5. Herramientas y consejos para sacar MUY buena nota

### Técnicas clave:

- Probar cada juego por separado
- Simular entradas “malas”
- Pensar como el catedrático corrigiendo
- Código **entendible > código sofisticado**

### Manual técnico:

- Pseudocódigo de:

    - Menú
    - Wordle
    - Batalla
- No UML avanzado, solo flujo lógico

# Esta práctica no se trata de ser genios, se trata de pensar ordenado.