Desarrollo de Aplicaciones Multiplataforma Programación
Examen 1º Trimestre
2025/2026

Nombre: Marcos Gracia García

1. ¿Cuál es la diferencia entre JRE y JDK? (Completa las siglas) 0.5 pt

JDK → (Java Development Kit) Es el kit de desarrollo de Java. Incluye un compilador (javac), la Máquina Virtual de Java (JVM) y unas librerías necesarias para ejecutar programas. Este se necesita para desarrollar, compilar y ejecutar programas Java.

JRE → (Java Runtime Environment) Es el entorno de ejecución de Java. Incluye la JVM y las librerías necesarias para ejecutar programas Java. El JRE permite ejecutar programas Java, pero no desarrollarlos.

// Dificultades: El JRE no aparecía en la teoría aunque seguro que se explicó en clase. La parte de JDK creo que lo hice casi perfecto.

2. ¿Qué es un Wrapper? Explica su utilidad, el concepto de autoboxing y el concepto de unboxing añadiendo ejemplos de todo. 1.5 pt

Un Wrapper (clase envolvente) es una clase que representa un tipo primitivo como un objeto. Cada tipo primitivo tiene su clase envolvente asociada en el paquete java.lang.Ejemplos de correspondencia: int → Integer, double → Double, boolean → Boolean, char → Character.

Los wrappers son útiles porque permiten tratar valores primitivos como objetos, son imprescindibles para colecciones genéricas (como ArrayList), proporcionan métodos útiles (por ejemplo, Integer.parseInt) y permiten usar null para representar ausencia de valor

El autoboxing es la conversión automática de un tipo primitivo a su clase envolvente. El unboxing es la conversión automática de un objeto wrapper a su tipo primitivo.

// Dificultades: Lo lei pero no lo supe con seguridad. No me acuerdo lo que puse exactamente, se que puse cosas que me sonaban y la esencia seguro que era. La parte de utilidad ahora ya no me acuerdo si la puse. Lo de autoboxing y unboxing ya asumí que estaba perdido.

3. ¿Cuál es la salida de este programa? 2 pt

```
public class EjercicioSalidaCorto {
	public static void main(String[] args) {
		int[] v = {3, -2, 4, 1, 0};
		int a = 2;
		int b = 1;
```

```
                for (int i = 0; i < v.length; i++) {
                        a += v[i];
                        if (v[i] < 0) {
                                b *= -1;
                                continue;
                        }
                        if (a % 2 == 0) {
                                b += i;
                        } else {
                                b -= a;
                        }
                }
                System.out.println(a);
                System.out.println(b);
        }
}
```

Salidas:
8
4

//Dificultades: Creo que el problema aquí podría estar en los despistes. Obtener el valor a era fácil pero la b necesitaba más cálculos que dependen en ocasiones de b. El truco era hacerlo poco a poco, apuntando los números como cambiaban durante el for. Aun así creo que falle en el b.

4. Diseña un programa de consola que simule el funcionamiento básico de un cajero automático para un único cliente fijo. El programa comienza siempre con un saldo inicial de 100 asociado a ese cliente. 2pt

Al iniciar la aplicación, debe mostrarse un menú dentro de un bucle para permitir operaciones sucesivas. El usuario introducirá una opción y el programa ejecutará la acción correspondiente.

El cajero debe permitir:
• Consultar el saldo actual del cliente.
• Ingresar una cantidad de dinero.
• Retirar una cantidad, validando que no exceda el saldo disponible.
• Registrar cada operación en un historial como texto descriptivo.
• Mostrar el historial completo acumulado.
• Salir del programa finalizando la sesión del cliente.

El programa debe rechazar cualquier opción no válida, haciendo que el menú se repita sin ejecutar acción.

```
--- CAJERO AUTOMÁTICO ---
1. Consultar saldo
2. Ingresar dinero
3. Retirar dinero
4. Ver historial
5. Salir
Elige una opción: 1

Saldo actual: 100€

--- CAJERO AUTOMÁTICO ---
1. Consultar saldo
2. Ingresar dinero
3. Retirar dinero
4. Ver historial
5. Salir
Elige una opción: 2

Cantidad a ingresar: 50
Ingreso realizado.
Nuevo saldo: 150€
```

```
--- CAJERO AUTOMÁTICO ---
1. Consultar saldo
2. Ingresar dinero
3. Retirar dinero
4. Ver historial
5. Salir
Elige una opción: 3

Cantidad a retirar: 40
Retirada realizada.
Nuevo saldo: 110€

--- CAJERO AUTOMÁTICO ---
1. Consultar saldo
2. Ingresar dinero
3. Retirar dinero
4. Ver historial
5. Salir
Elige una opción: 4

--- HISTORIAL ---
Ingreso de 50€
Retirada de 40€
```

```java
package Examen_Package;

import java.util.Scanner;

import java.util.ArrayList;

import java.util.List;

public class Ejercicio4_Marcos_Gracia {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Scanner sc = new Scanner(System.in);

        int saldo_inicial = 100;

        int opción;

        List<String> historial = new ArrayList<>();
        historial.add("Saldo inicial: " + saldo_inicial);

        do {

            System.out.println("--- CAJERO AUTOMÁTICO ---");
            System.out.println("1. Consultar saldo");
            System.out.println("2. Ingresar dinero");
            System.out.println("3. Retirar dinero");
```

```
            System.out.println("4. Ver historial");
            System.out.println("5. Salir");
            System.out.print("Elige una opción: ");
            opción = sc.nextInt();

            switch (opción) {

            case 1: {

                  System.out.println("Saldo actual: " + saldo_inicial +
"$");

                break;

            }

            case 2: {

                System.out.print("Cantidad a ingresar: ");
                int ingreso = sc.nextInt();

                saldo_inicial = saldo_inicial + ingreso;

                System.out.println("Ingreso realizado.");

                   System.out.println("Nuevo saldo: " + saldo_inicial +
"$");

                historial.add("Ingreso de " + ingreso + "$");

                break;

            }

            case 3: {

                System.out.print("Cantidad a retirar: ");
                int retiro = sc.nextInt();

                if ((saldo_inicial - retiro) < 0) {

                    System.out.println("Retirada no realizada.");
```

```
                    System.out.println("Saldo actual: " + saldo_inicial
+ "$");

                } else {

                    saldo_inicial = saldo_inicial - retiro;

                    System.out.println("Retirada realizada.");

                     System.out.println("Nuevo saldo: " + saldo_inicial
+ "$");

                    historial.add("Retirada de " + retiro + "$");

                }

                break;

            }

            case 4: {

                System.out.println("--- HISTORIAL ---");

                for (int i = 0; i < historial.size(); i++)
                    System.out.println(historial.get(i));

                break;

            }

            case 5: {

                System.out.println("Has finalizado tu sesión.");

                break;

            }

            }

            System.out.println();
```

```
        } while (opción != 5);

        sc.close();

    }

}
```

//Dificultades: Así es como lo hice literalmente y creo que no tiene ningún defecto, se lo he pasado a una IA y hay cosas que se podrían mejorar pero no son requeridas. Es cierto que para hacer la lista para el historial me costó un poco más, pero sin problema. Salí muy contento de hacer esto.

5. Implementa un programa en Java que simule un tablero de ajedrez muy simplificado, utilizando únicamente una matriz bidimensional y operaciones básicas con variables y bucles. 4 pt

Requisitos:

• Representa el tablero como una matriz de 8×8 de caracteres.
• El carácter "." indicará una casilla vacía.
• Coloca dos piezas en posiciones iniciales fijas: Un caballo ("C") y un peón ("P")
• Muestra el tablero por consola en formato legible, indicando filas y columnas.
• Pregunte al usuario qué pieza desea mover: «caballo» o «peón».
• Según la pieza elegida, calcula todas las posiciones a las que esa pieza puede moverse de forma válida:
    ‣ El caballo debe generar sus 8 posibles movimientos en L y aceptar solo los que estén dentro del tablero y en casillas vacías.
    ‣ El peón se moverá únicamente una casilla hacia arriba si dicha casilla está libre.
• Muestra al usuario los movimientos válidos en coordenadas numéricas (por ejemplo: «(1,1)», «(2,2)», etc).
• El usuario elegirá uno de esos movimientos indicando su índice.
• El programa actualizará la matriz del tablero moviendo la pieza elegida.
• Tras cada movimiento, vuelve a mostrar el tablero completo actualizado.

Condiciones adicionales:

• Todo el comportamiento debe implementarse en un único programa con instrucciones simples (for, if, while, arrays).
• El tablero debe actualizarse correctamente después de cada jugada.
• El usuario podrá repetir el proceso indefinidamente mientras el programa esté en ejecución.

```
  Tablero actual:
     0 1 2 3 4 5 6 7
  ------------------
  0 | . C . . . . . .
  1 | . . . . . . . .
  2 | . . . . . . . .
  3 | . . . . . . . .
  4 | . . . . . . . .
  5 | . . . . . . . .
  6 | . P . . . . . .
  7 | . . . . . . . .
  ------------------

  ¿Qué pieza quieres mover? (caballo/peon/
salir): peon

  Movimientos válidos del peón:
  [0] (5,1)

  Elige el índice del movimiento: 0

  Moviendo peón a (5,1)...

  Tablero actualizado:
     0 1 2 3 4 5 6 7
  ------------------
  0 | . C . . . . . .
  1 | . . . . . . . .
  2 | . . . . . . . .
  3 | . . . . . . . .
  4 | . . . . . . . .
  5 | . P . . . . . .
  6 | . . . . . . . .
  7 | . . . . . . . .
  ------------------
```

```
  ¿Qué pieza quieres mover? (caballo/peon/
salir): caballo

  Movimientos válidos del caballo:
  [0] (1,3)
  [1] (2,0)
  [2] (2,2)

  Elige el índice del movimiento: 2

  Moviendo caballo a (2,2)...

  Tablero actualizado:
     0 1 2 3 4 5 6 7
  ------------------
  0 | . . . . . . . .
  1 | . . . . . . . .
  2 | . . C . . . . .
  3 | . . . . . . . .
  4 | . . . . . . . .
  5 | . P . . . . . .
  6 | . . . . . . . .
  7 | . . . . . . . .
  ------------------

  ¿Qué pieza quieres mover? (caballo/peon/
salir): salir

  Fin del programa.
```

```java
package Tema_3;
import java.util.Scanner;
public class Ejercicio5_AjedrezSimplificado {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Tablero actual:");
		char[][] tablero = new char[8][8];
		for (int i = 0; i < 8; i++) {
			for (int j = 0; j < 8; j++) {
				tablero[i][j] = '.';
			}
		}
		int caballo_fila = 0, caballo_columna = 1;
		int peon_fila = 6, peon_columna = 1;
		tablero[caballo_fila][caballo_columna] = 'C';
		tablero[peon_fila][peon_columna] = 'P';
		boolean salir = false;
		while (!salir) {
			System.out.println("    0 1 2 3 4 5 6 7");
			System.out.println("------------------");
```

```
                        for (int i = 0; i < 8; i++) {
                                System.out.print(i + " | ");
                                for (int j = 0; j < 8; j++) {
                                        System.out.print(tablero[i][j] + " ");
                                }
                                System.out.println();
                        }
                        System.out.println("------------------");
                        System.out.print("¿Qué pieza quieres mover?
(caballo/peon/salir): ");
                        String opcion = sc.next();
                        if (opcion.equals("salir")) {
                                salir = true;

                        } else if (opcion.equals("caballo")) {
                                int[][] movimientos = { { -2, -1 }, { -2, 1 }, { -1, -2 }, { -1, 2 },
{ 1, -2 }, { 1, 2 }, { 2, -1 },
                                                { 2, 1 } };
                                int[][] validos = new int[8][2];
                                int contador = 0;
                                for (int i = 0; i < movimientos.length; i++) {
                                        int nueva_fila = caballo_fila + movimientos[i][0];
                                        int nueva_columna = caballo_columna +
movimientos[i][1];
                                        if (nueva_fila >= 0 && nueva_fila < 8 &&
nueva_columna >= 0 && nueva_columna < 8
                                                        &&
tablero[nueva_fila][nueva_columna] == '.') {
                                                validos[contador][0] = nueva_fila;
                                                validos[contador][1] = nueva_columna;
                                                contador++;
                                        }
                                }
                                if (contador == 0) {
                                        System.out.println("No hay movimientos válidos.");
                                        continue;
                                }
                                System.out.println("Movimientos válidos del caballo:");
                                for (int i = 0; i < contador; i++) {
                                        System.out.println("[" + i + "] (" + validos[i][0] + ","
+ validos[i][1] + ")");
                                }
                                System.out.print("Elige el índice del movimiento: ");
                                int eleccion = sc.nextInt();
```

```
                        tablero[caballo_fila][caballo_columna] = '.';
                        caballo_fila = validos[eleccion][0];
                        caballo_columna = validos[eleccion][1];
                        tablero[caballo_fila][caballo_columna] = 'C';
                    } else if (opcion.equals("peon")) {
                        int nuevaFila = peon_fila - 1;
                        if (nuevaFila >= 0 && tablero[nuevaFila][peon_columna]
== '.') {
                            int eleccion = 0;
                            do {
                                System.out.println("Movimientos válidos del
peón:");
                                System.out.println("[0] (" + nuevaFila + "," +
peon_columna + ")");
                                System.out.print("Elige el índice del
movimiento: ");
                                eleccion = sc.nextInt();
                            } while (eleccion != 0);
                            tablero[peon_fila][peon_columna] = '.';
                            peon_fila = nuevaFila;
                            tablero[peon_fila][peon_columna] = 'P';
                        } else {
                            System.out.println("El peón no puede moverse.");
                        }
                    }
                    else {
                        System.out.println("Opción no válida.");
                    }
                }
                System.out.println("Fin del programa.");
                sc.close();
        }
}
```

// Dificultades: No era tan difícil como pensaba durante el examen. En el examen solo hice la matriz del tablero y el diseño. La parte del caballo me ha costado un poco hacerla porque los movimientos no eran tan simples pero una vez que te sale lo haces. La comprobación de si el movimiento es válido y si no quitarlo lo mismo. En mi opinión. La verdad es que en el examen me daba tiempo a hacer algo más pero entre que no me funcionan a veces las neuronas pues se quedo asi.