INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	FUNDAMENTOS PROGRAMACION 2				
TÍTULO DE LA PRÁCTICA:	*DEFINICIÓN DE CLASES*				
NÚMERO DE PRÁCTICA:	*10*	**AÑO LECTIVO:**	*2024-A*	**NRO. SEMESTRE:**	*II*
FECHA DE PRESENTACIÓN		**HORA DE PRESENTACIÓN**			
INTEGRANTE (s): PACHECO ESQUINARILA MILENE				**NOTA:**	
DOCENTE(s): *ING. RONALD MANCINI TICONA*					

SOLUCIÓN Y RESULTADOS

Este laboratorio requiere que usted escriba un programa utilizando clases definidas por el programador. No deberá utilizar sintaxis o constructores que no han sido cubiertos durante las clases teóricas. Será penalizado por esta falta. A menos que una plantilla sea dada, deberá utilizar cada programa desde cero de manera que obtenga suficiente práctica en la escritura de programas en Java.

Un consejo: Programe incrementalmente. No trate de terminar todas las partes del programa y luego compilarlo. Escriba sus programas en partes y compílelo de forma frecuente. Trate de mantener un programa compilable aun cuando esté trabajando en él. Presentar un programa compilable que funcione parcialmente es mejor que presentar un programa no-compilable. EN SERIO, programe incrementalmente.

Soldier.java

```java
public class Soldier {
    private String name;
    private int attackLevel;
    private int defenseLevel;
    private int healthLevel;
    private int currentHealth;
    private int speed = 0;
    private String attitude = "defense";
    private boolean isAlive = true;
    private int row;
    private int column;
    private int army;

    public static final int MAX_SOLDIERS_PER_ARMY = 10;
    public static final int MAX_HEALTH = 5;
```

```java
    public static final int MIN_HEALTH = 1;

    public Soldier(Soldier original) {
        this.name = original.name;
        this.attackLevel = original.attackLevel;
        this.defenseLevel = original.defenseLevel;
        this.healthLevel = original.healthLevel;
        this.currentHealth = original.currentHealth;
        this.speed = original.speed;
        this.attitude = original.attitude;
        this.isAlive = original.isAlive;
        this.army = original.army;
    }

    public Soldier(String name, int health, int row, int col) {
        this.name = name;
        this.healthLevel = health;
        this.currentHealth = health;
        this.row = row;
        this.column = col;
    }

    public Soldier(String name, int health, int row, int col, int army) {
        this.name = name;
        this.healthLevel = health;
        this.currentHealth = health;
        this.row = row;
        this.column = col;
        this.army = army;
    }

    public Soldier(String name, int attack, int defense, int health, int row, int col,
int army) {
        this.name = name;
        this.attackLevel = attack;
        this.defenseLevel = defense;
        this.healthLevel = health;
        this.currentHealth = health;
        this.row = row;
        this.column = col;
        this.army = army;
    }

    public String getName() {
        return name;
    }

    public int getHealthLevel() {
        return healthLevel;
    }
```

```java
public void setHealthLevel(int health) {
    this.healthLevel = health;
}

public int getRow() {
    return row;
}

public void setRow(int row) {
    this.row = row;
}

public int getColumn() {
    return column;
}

public void setColumn(int col) {
    this.column = col;
}

public String getAcronym() {
    return name.substring(0, 1) + healthLevel;
}

public int getArmy() {
    return army;
}

public void attack() {
    attitude = "attack";
    moveForward();
}

public void defend() {
    attitude = "defense";
    if (speed > 0)
        speed = 0;
    else
        speed--;
}

public void moveForward() {
    speed++;
}

public void moveBackward() {
    speed--;
}
```

```
    public void beAttacked() {
        currentHealth--;
        if (currentHealth == 0)
            die();
    }

    public void flee() {
        attitude = "flee";
        speed += 2;
    }

    public void die() {
        isAlive = false;
    }

    public boolean isAlive() {
        return isAlive;
    }

    public int getCurrentHealth() {
        return currentHealth;
    }

    public void setCurrentHealth(int newHealth) {
        currentHealth = newHealth;
    }

    public void setAttackLevel(int attackLevel) {
        this.attackLevel = attackLevel;
    }

    public void setDefenseLevel(int defenseLevel) {
        this.defenseLevel = defenseLevel;
    }

    public String getPosition() {
        return Integer.toString(row + 1) + (char) (column + 65);
    }

    public String getSoldierInfo() {
        return "Name: " + name + ", Current Health: " + currentHealth + ", Position: " +
getPosition();
    }

    public void increaseHealth(int amount) {
        currentHealth += amount;
    }

    public boolean equals(Soldier other) {
        return this.name.equals(other.name) && this.currentHealth == other.currentHealth
```

```
                && this.attackLevel == other.attackLevel && this.defenseLevel ==
other.defenseLevel
                && this.isAlive == other.isAlive;
    }

    public Soldier addAttributes(Soldier otherSoldier) {
        this.healthLevel += otherSoldier.healthLevel;
        this.attackLevel += otherSoldier.attackLevel;
        this.defenseLevel += otherSoldier.defenseLevel;
        this.speed += otherSoldier.speed;
        return this;
    }

    public int calculateTotalAttributes() {
        return this.attackLevel + this.defenseLevel + this.healthLevel + this.speed;
    }

    public String toString() {
        String status = isAlive ? "Alive" : "Dead";

        return "- Name: " + name + ", Attack Level: " + attackLevel + ", Defense Level:
" + defenseLevel
                + ", Health Level: " + healthLevel + ", Current Health: " +
currentHealth + ", Speed: " + speed
                + ", Attitude: " + attitude + ", Status: " + status + ", Position: " +
getPosition();
    }
}
```

VideoGame.java

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.InputMismatchException;
import java.util.Random;
import java.util.Scanner;

public class VideoGame {
    public static final int BOARD_SIZE = 10;
    public static final String ANSI_GREEN = "\u001B[32m";
    public static final String ANSI_CYAN = "\u001B[36m";
    public static final String ANSI_RESET = "\u001B[0m";

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        menu(sc);
        sc.close();
    }
```

```
    public static void menu(Scanner sc) {
        int option;

        while (true) {
            displayMenuOptions();
            option = getUserChoice(sc);

            switch (option) {
                case 1:
                    playQuickGame(sc);
                    break;
                case 2:
                    playCustomGame(sc);
                    break;
                case 3:
                    System.exit(3);
            }
        }
    }

    public static void displayMenuOptions() {
        System.out.println("\n1. Quick Game");
        System.out.println("2. Custom Game");
        System.out.println("3. Exit");
    }

    public static int getUserChoice(Scanner sc) {
        while (true) {
            try {
                System.out.print("Enter the corresponding number: ");
                return sc.nextInt();
            } catch (InputMismatchException e) {
                System.out.println("\nInvalid input. Please enter a number.\n");
                sc.nextLine();
            }
        }
    }

    public static void createArmy(Soldier[][] board, ArrayList<Soldier> army, int
armyNumber) {
        Random r = new Random();

        int numSoldiers = r.nextInt(Soldier.MAX_SOLDIERS_PER_ARMY) + 1;

        for (int i = 0; i < numSoldiers; i++) {
            String name = "Soldier" + i + "X" + armyNumber;
            int healthLevel = r.nextInt(Soldier.MAX_HEALTH - Soldier.MIN_HEALTH + 1) +
Soldier.MIN_HEALTH;
            int attackLevel = r.nextInt(Soldier.MAX_HEALTH - Soldier.MIN_HEALTH + 1) +
Soldier.MIN_HEALTH;
```

```
            int defenseLevel = r.nextInt(Soldier.MAX_HEALTH - Soldier.MIN_HEALTH + 1) +
Soldier.MIN_HEALTH;
            int row = r.nextInt(BOARD_SIZE);
            int column = r.nextInt(BOARD_SIZE);

            while (board[row][column] != null) {
                row = r.nextInt(BOARD_SIZE);
                column = r.nextInt(BOARD_SIZE);
            }

            army.add(new Soldier(name, attackLevel, defenseLevel, healthLevel, row,
column, armyNumber));
            board[row][column] = army.get(i);
        }
    }

    public static void playQuickGame(Scanner sc) {
        Soldier[][] board = new Soldier[BOARD_SIZE][BOARD_SIZE];
        ArrayList<Soldier> army1 = new ArrayList<Soldier>();
        ArrayList<Soldier> army2 = new ArrayList<Soldier>();

        createArmy(board, army1, 1);
        createArmy(board, army2, 2);

        System.out.println("\nYou have selected Quick Game");

        displayArmiesAndBoard(army1, army2, board);

        System.out.println("\nEnter 'q' to exit the game");
        playGame(army1, army2, board, sc);
    }

    public static void playCustomGame(Scanner sc) {
        Soldier[][] board = new Soldier[BOARD_SIZE][BOARD_SIZE];
        ArrayList<Soldier> army1 = new ArrayList<>();
        ArrayList<Soldier> army2 = new ArrayList<>();

        createArmy(board, army1, 1);
        createArmy(board, army2, 2);

        System.out.println("\nYou have selected Custom Game");

        displayArmiesAndBoard(army1, army2, board);

        int selectedArmyNumber = selectArmy(sc);
        ArrayList<Soldier> selectedArmy = (selectedArmyNumber == 1) ? army1 : army2;
        String color = (selectedArmyNumber == 1) ? ANSI_GREEN : ANSI_CYAN;

        while (true) {
            System.out.println();
```

```
            printGameBoard(board);
            System.out.println("\n" + color + "Selected Army: " + selectedArmyNumber +
ANSI_RESET);
            displaySubMenuOptions();
            int option = getUserChoice(sc);

            switch (option) {
                case 0:
                    selectedArmy = (selectedArmyNumber == 1) ? army2 : army1;
                    selectedArmyNumber = (selectedArmyNumber == 1) ? 2 : 1;
                    color = (selectedArmyNumber == 1) ? ANSI_GREEN : ANSI_CYAN;
                    break;
                case 1:
                    createSoldier(board, selectedArmy, sc);
                    break;
                case 2:
                    removeSoldier(board, selectedArmy, sc);
                    break;
                case 3:
                    cloneSoldier(board, selectedArmy, sc);
                    break;
                case 4:
                    modifySoldier(board, selectedArmy, sc);
                    break;
                case 5:
                    compareSoldiers(selectedArmy, sc);
                    break;
                case 6:
                    swapSoldiers(selectedArmy, sc);
                    break;
                case 7:
                    viewSoldierDetails(selectedArmy, sc);
                    break;
                case 8:
                    viewArmy(selectedArmy);
                    break;
                case 9:
                    sumSoldierLevels(selectedArmy);
                    break;
                case 10:
                    playGame(army1, army2, board, sc);
                    break;
                case 11:
                    menu(sc);
                    break;
            }
        }
    }

    public static void displaySubMenuOptions() {
```

```
        System.out.println("\n0. Switch to the other army");
        System.out.println("1. Create a new Soldier");
        System.out.println("2. Remove a Soldier");
        System.out.println("3. Clone a Soldier");
        System.out.println("4. Modify a Soldier");
        System.out.println("5. Compare Soldiers");
        System.out.println("6. Swap Soldiers");
        System.out.println("7. View Soldier Details");
        System.out.println("8. View Army");
        System.out.println("9. Sum Soldier Levels");
        System.out.println("10. Play");
        System.out.println("11. Back to Main Menu");
    }

    public static int selectArmy(Scanner sc) {
        int selectedArmyNumber;
        do {
            System.out.print("\nEnter the army number you want to manage (1 or 2): ");
            selectedArmyNumber = getUserChoice(sc);
        } while (selectedArmyNumber != 1 && selectedArmyNumber != 2);
        System.out.println("\nYou have selected to manage Army " + selectedArmyNumber);
        return selectedArmyNumber;
    }

    public static void printGameBoard(Soldier[][] board) {
        String cellSeparator = "      |";
        String[] columnLetters = { "A", "B", "C", "D", "E", "F", "G", "H", "I", "J" };

        System.out.println("
_____________________________________________________________________________");

        printEmptyRow();
        System.out.print("\n|      |");

        for (String letter : columnLetters)
            System.out.printf("  %s.  |", letter);

        printRowSeparator();

        for (int i = 0; i < board.length; i++) {
            System.out.println();
            printEmptyRow();

            System.out.printf("\n|  %02d  |", (i + 1));

            for (int j = 0; j < board[i].length; j++)
                if (board[i][j] != null) {
                    String color = (board[i][j].getArmy() == 1) ? ANSI_GREEN :
ANSI_CYAN;
                    String format = String.format("%02d",
```

```
board[i][j].getCurrentHealth());
                    System.out.print(color + "  " + format + ANSI_RESET + "  |");
                } else
                    System.out.print(cellSeparator);

            printRowSeparator();
        }

        System.out.println();
    }

    public static void printEmptyRow() {
        String cellSeparator = "      |";
        System.out.print("|");

        for (int j = 0; j < 11; j++) {
            System.out.print(cellSeparator);
        }
    }

    public static void printRowSeparator() {
        String rowSeparator = "|";
        System.out.print("\n|");

        for (int k = 0; k < 11; k++) {
            System.out.print(rowSeparator);
        }
    }

    public static void displaySoldiers(ArrayList<Soldier> army) {
        for (Soldier soldier : army) {
            if (soldier.isAlive()) {
                System.out.println(soldier);
            }
        }
    }

    public static void displayArmiesAndBoard(ArrayList<Soldier> army1,
ArrayList<Soldier> army2, Soldier[][] board) {
        System.out.println("\nGame Board\n");
        printGameBoard(board);

        System.out.println(ANSI_GREEN + "\nArmy Number 1\n" + ANSI_RESET);
        displaySoldiers(army1);

        System.out.println(ANSI_CYAN + "\nArmy Number 2\n" + ANSI_RESET);
        displaySoldiers(army2);
    }

    public static void playGame(ArrayList<Soldier> army1, ArrayList<Soldier> army2,
```

```
Soldier[][] board, Scanner sc) {
        int currentTurn = 1;
        ArrayList<Soldier> winner = null;

        while (winner == null) {
            currentTurn = (currentTurn == 2) ? 1 : 2;

            ArrayList<Soldier> currentArmy = (currentTurn == 2) ? army1 : army2;
            ArrayList<Soldier> opponentArmy = (currentTurn == 2) ? army2 : army1;

            playTurn(currentArmy, opponentArmy, board, sc);

            winner = determineWinner(army1, army2);
        }

        printGameBoard(board);
        int winningArmy = winner.get(0).getArmy();
        String color = (winningArmy == 1) ? ANSI_GREEN : ANSI_CYAN;
        System.out.println(color + "\nArmy " + winningArmy + " has won the war!" +
ANSI_RESET);
    }

    public static void playTurn(ArrayList<Soldier> army, ArrayList<Soldier> oppArmy,
Soldier[][] board, Scanner sc) {
        System.out.println();
        printGameBoard(board);
        Soldier soldier = selectSoldier(army, sc);
        System.out.println("\n" + soldier + "\n");
        ArrayList<String> validMoves = getValidMoves(board, soldier.getRow(),
soldier.getColumn());
        String coordinate = selectValidMove(validMoves, sc);
        moveSoldier(board, army, oppArmy, soldier, coordinate);
    }

    public static Soldier selectSoldier(ArrayList<Soldier> army, Scanner sc) {
        int armyNumber = army.get(0).getArmy();
        String color = (armyNumber == 1) ? ANSI_GREEN : ANSI_CYAN;

        System.out.println(color + "\nArmy " + armyNumber + "." + ANSI_RESET);
        System.out.println("\nSelect a soldier\n");

        int position;
        do {
            for (int i = 0; i < army.size(); i++)
                System.out.println((i + 1) + ". " + army.get(i).getSoldierInfo());

            System.out.print("\nEnter the number of the soldier: ");
            position = checkForQuitInput(sc) - 1;

            if (position < 0 || position >= army.size()) {
```

```
                System.out.println("\nInvalid number. Please enter a number within the
valid range.\n");
            }

        } while (position < 0 || position >= army.size());

        Soldier soldier = army.get(position);

        return soldier;
    }

    public static ArrayList<String> getValidMoves(Soldier[][] board, int row, int
column) {
        ArrayList<String> validMoves = new ArrayList<>();

        for (int i = -1; i <= 1; i++) {
            for (int j = -1; j <= 1; j++) {
                int newRow = row + i;
                int newColumn = column + j;

                if (isValidMove(board, newRow, newColumn, board[row][column].getArmy()))
{
                    validMoves.add(getCoordinate(newRow, newColumn));
                }
            }
        }

        return validMoves;
    }

    public static String selectValidMove(ArrayList<String> validMoves, Scanner sc) {
        int position;

        do {
            for (int i = 0; i < validMoves.size(); i++)
                System.out.println((i + 1) + ". " + validMoves.get(i));

            System.out.print("\nEnter the number of the position: ");
            position = checkForQuitInput(sc) - 1;

            if (position < 0 || position >= validMoves.size()) {
                System.out.println("\nInvalid number. Please enter a number within the
valid range.\n");
            }

        } while (position < 0 || position >= validMoves.size());

        return validMoves.get(position);
    }
```

```
    public static boolean isValidMove(Soldier[][] board, int destRow, int destColumn,
int armyNumber) {
        if (destRow < 0 || destRow >= board.length || destColumn < 0 || destColumn >=
board[0].length) {
            return false;
        }

        Soldier destinationSoldier = board[destRow][destColumn];
        return destinationSoldier == null || destinationSoldier.getArmy() != armyNumber;
    }

    public static int checkForQuitInput(Scanner sc) {
        while (true) {
            String input = sc.next();

            if (input.equalsIgnoreCase("q")) {
                handleGameCancellation(sc);
            }

            try {
                return Integer.parseInt(input);
            } catch (InputMismatchException | NumberFormatException e) {
                System.out.print("\nInvalid input. Please enter a number: ");
                sc.nextLine();
            }
        }
    }

    public static String getCoordinate(int row, int column) {
        char columnLetter = (char) (column + 65);
        int rowNumber = row + 1;

        return Integer.toString(rowNumber) + columnLetter;
    }

    public static void moveSoldier(Soldier[][] board, ArrayList<Soldier> currentArmy,
ArrayList<Soldier> opponentArmy,
            Soldier soldier, String coordinate) {
        int currentRow = soldier.getRow();
        int currentColumn = soldier.getColumn();

        HashMap<String, Integer> rowColumn = convertPosition(coordinate);

        int destRow = rowColumn.get("row");
        int destColumn = rowColumn.get("column");

        Soldier winner = soldier;
        if (isEnemyOccupied(board, destRow, destColumn, soldier.getArmy())) {
            Soldier enemySoldier = board[destRow][destColumn];
            winner = resolveBattle(currentArmy, opponentArmy, soldier, enemySoldier);
```

```
        }

        board[currentRow][currentColumn] = null;
        board[destRow][destColumn] = winner;
        winner.setRow(destRow);
        winner.setColumn(destColumn);
    }

    public static HashMap<String, Integer> convertPosition(String position) {
        HashMap<String, Integer> rowColumn = new HashMap<>();

        char column = Character.toUpperCase(position.charAt(position.length() - 1));
        int row = Integer.parseInt(position.substring(0, position.length() - 1)) - 1;
        int col = (int) column - 65;

        rowColumn.put("row", row);
        rowColumn.put("column", col);

        return rowColumn;
    }

    public static boolean isEnemyOccupied(Soldier[][] board, int destRow, int
destColumn, int currentArmy) {
        return board[destRow][destColumn] != null &&
board[destRow][destColumn].getArmy() != currentArmy;
    }

    public static Soldier resolveBattle(ArrayList<Soldier> army, ArrayList<Soldier>
oppArmy, Soldier soldier,
            Soldier oppSoldier) {
        double totalHealth = soldier.getCurrentHealth() + oppSoldier.getCurrentHealth();
        double probabilitySoldier = (soldier.getCurrentHealth() / totalHealth) * 100;
        double probabilityOpp = 100 - probabilitySoldier;

        System.out.println("\nBattle at position " + soldier.getPosition() + "!\n");

        System.out.println(soldier.getName() + " has a probability of winning: " +
probabilitySoldier + "%.");
        System.out.println(oppSoldier.getName() + " has a probability of winning: " +
probabilityOpp + "%.");

        Random random = new Random();
        double randomNumber = random.nextDouble() * 100;
        Soldier winner;

        if (randomNumber <= probabilitySoldier) {
            winner = soldier;
            removeSoldierFromArmy(army, oppArmy, oppSoldier);
        } else {
            winner = oppSoldier;
```

```
            removeSoldierFromArmy(army, oppArmy, soldier);
        }

        System.out.println(winner.getArmy() == 1 ? ANSI_GREEN : ANSI_CYAN + "\n");
        System.out.println("With a probability of " + randomNumber + "%, " +
winner.getName() + " has won the battle!");
        System.out.println(ANSI_RESET);
        winner.increaseHealth(1);

        return winner;
    }

    public static void removeSoldierFromArmy(ArrayList<Soldier> army, ArrayList<Soldier>
oppArmy, Soldier soldier) {
        soldier.die();

        if (army.contains(soldier))
            army.remove(soldier);
        else
            oppArmy.remove(soldier);
    }

    public static ArrayList<Soldier> determineWinner(ArrayList<Soldier> army1,
ArrayList<Soldier> army2) {
        if (army1.size() != 0 && army2.size() != 0) {
            return null;
        }

        return (army1.size() == 0) ? army2 : army1;
    }

    public static void handleGameCancellation(Scanner sc) {
        while (true) {
            System.out.println("\nDo you want to cancel the current game?");
            System.out.println("1. Start a new game");
            System.out.println("2. Return to the main menu");

            int choice = getUserChoice(sc);

            switch (choice) {
                case 1:
                    playQuickGame(sc);
                    break;
                case 2:
                    menu(sc);
                    break;
            }
        }
    }
```

```
    public static void createSoldier(Soldier[][] board, ArrayList<Soldier> army, Scanner
sc) {
        if (army.size() >= Soldier.MAX_SOLDIERS_PER_ARMY) {
            System.out.println("\nThe army is full, no more soldiers can be created!");
            return;
        }

        System.out.print("\nEnter the soldier's name: ");
        String name = sc.next();
        System.out.print("Enter attack level: ");
        int attackLevel = sc.nextInt();
        System.out.print("Enter defense level: ");
        int defenseLevel = sc.nextInt();
        System.out.print("Enter health level: ");
        int healthLevel = sc.nextInt();
        System.out.print("Enter the position (e.g. 1A): ");
        String position = sc.next();

        HashMap<String, Integer> rowColumn = convertPosition(position);
        int row = rowColumn.get("row");
        int column = rowColumn.get("column");

        while (!isValidPosition(board, army, position)) {
            System.out.print("\nEnter the position (e.g. 1A): ");
            position = sc.next();
            rowColumn = convertPosition(position);
            row = rowColumn.get("row");
            column = rowColumn.get("column");
        }

        int armyNumber = army.get(0).getArmy();

        Soldier soldier = new Soldier(name, attackLevel, defenseLevel, healthLevel, row,
column, armyNumber);
        army.add(soldier);
        board[row][column] = soldier;

        System.out.println("\nSoldier created successfully!");
    }

    public static boolean isValidPosition(Soldier[][] board, ArrayList<Soldier> army,
String position) {
        HashMap<String, Integer> rowColumn = convertPosition(position);
        int row = rowColumn.get("row");
        int column = rowColumn.get("column");

        if (!isValidMove(board, row, column, army.get(0).getArmy())) {
            System.out.println("\nInvalid position. Please enter a valid position.");
            return false;
        }
```

```
        if (board[row][column] != null) {
            System.out.println("\nPosition is already occupied. Please choose another
position.");
            return false;
        }

        return true;
    }


    public static void removeSoldier(Soldier[][] board, ArrayList<Soldier> army, Scanner
sc) {
        if (army.size() == 1) {
            System.out.println("\nUnable to delete the last soldier in the army. At
least one soldier must remain.");
            return;
        }

        Soldier removedSoldier = selectSoldier(army, sc);

        board[removedSoldier.getRow()][removedSoldier.getColumn()] = null;
        army.remove(removedSoldier);
        System.out.println("\nSoldier successfully eliminated.");
    }

    public static void cloneSoldier(Soldier[][] board, ArrayList<Soldier> army, Scanner
sc) {
        if (army.size() >= Soldier.MAX_SOLDIERS_PER_ARMY) {
            System.out.println("\nThe army is full, no more soldiers can be created!");
            return;
        }

        Soldier soldier = selectSoldier(army, sc);

        System.out.print("Enter the position (e.g. 1A): ");
        String position = sc.next();

        HashMap<String, Integer> rowColumn = convertPosition(position);
        int row = rowColumn.get("row");
        int column = rowColumn.get("column");

        while (!isValidPosition(board, army, position)) {
            System.out.print("\nEnter the position (e.g. 1A): ");
            position = sc.next();
            rowColumn = convertPosition(position);
            row = rowColumn.get("row");
            column = rowColumn.get("column");
        }
```

```
        Soldier clonedSoldier = new Soldier(soldier);
        clonedSoldier.setRow(row);
        clonedSoldier.setColumn(column);
        army.add(clonedSoldier);
        board[row][column] = soldier;
    }

    public static void modifySoldier(Soldier[][] board, ArrayList<Soldier> army, Scanner
sc) {
        Soldier soldierToModify = selectSoldier(army, sc);
        int option = -1;

        while (option < 1 || option > 3) {
            displayModifySoldierMenu();
            option = getUserChoice(sc);
        }

        switch (option) {
            case 1:
                System.out.print("\nEnter the new attack level: ");
                int attackLevel = sc.nextInt();
                soldierToModify.setAttackLevel(attackLevel);
                break;
            case 2:
                System.out.print("\nEnter the new defense level: ");
                int defenseLevel = sc.nextInt();
                soldierToModify.setDefenseLevel(defenseLevel);
                break;
            case 3:
                System.out.print("\nEnter the new health level: ");
                int healthLevel = sc.nextInt();
                soldierToModify.setCurrentHealth(healthLevel);
                break;
        }

        System.out.println("\nThe soldier has been successfully modified.");
    }

    public static void displayModifySoldierMenu() {
        System.out.println("\nSoldier Modification Submenu:");
        System.out.println("1. Modify attack level");
        System.out.println("2. Modify defense level");
        System.out.println("3. Modify current health");
    }

    public static void compareSoldiers(ArrayList<Soldier> army, Scanner sc) {
        System.out.println("\nEnter the position of the first soldier: ");
        Soldier firstSoldier = selectSoldier(army, sc);

        System.out.println("\nEnter the position of the second soldier: ");
```

```
        Soldier secondSoldier = selectSoldier(army, sc);

        if (firstSoldier.equals(secondSoldier))
            System.out.println("\nThe selected soldiers are identical.");
        else
            System.out.println("\nThe selected soldiers are different.");
    }

    public static void swapSoldiers(ArrayList<Soldier> army, Scanner sc) {
        System.out.println("\nEnter the position of the first soldier: ");
        Soldier firstSoldier = selectSoldier(army, sc);

        System.out.println("\nEnter the position of the second soldier: ");
        Soldier secondSoldier = selectSoldier(army, sc);

        if (!firstSoldier.equals(secondSoldier)) {
            int firstSoldierIndex = army.indexOf(firstSoldier);
            int secondSoldierIndex = army.indexOf(secondSoldier);

            army.set(firstSoldierIndex, secondSoldier);
            army.set(secondSoldierIndex, firstSoldier);

            System.out.println("\nSoldiers swapped successfully.");
        } else {
            System.out.println("\nInvalid soldiers selected. Please try again.");
        }
    }

    public static void viewSoldierDetails(ArrayList<Soldier> army, Scanner sc) {
        System.out.print("\nEnter the name of the soldier to view details: ");
        String soldierName = sc.next();

        Soldier foundSoldier = findSoldierByName(army, soldierName);

        if (foundSoldier != null) {
            System.out.println("\nSoldier details:\n");
            System.out.println(foundSoldier.toString());
        } else {
            System.out.println("\nSoldier with the name '" + soldierName + "' not
found.");
        }
    }

    public static Soldier findSoldierByName(ArrayList<Soldier> army, String name) {
        for (Soldier soldier : army)
            if (soldier.getName().equalsIgnoreCase(name))
                return soldier;

        return null;
    }
```

```
    public static void viewArmy(ArrayList<Soldier> army) {
        String color = army.get(0).getArmy() == 1 ? ANSI_GREEN : ANSI_CYAN;
        System.out.println(color + "\nArmy details:\n" + ANSI_RESET);
        displaySoldiers(army);
    }

    public static void sumSoldierLevels(ArrayList<Soldier> army) {
        Soldier totalAttributes = new Soldier("", 0, 0, 0, 0);

        for (Soldier soldier : army)
            totalAttributes.addAttributes(soldier);

        System.out.println("\nTotal attributes of the army: " +
totalAttributes.calculateTotalAttributes());
    }
}
```

Ejecución en Consola

```
PROBLEMAS    SALIDA    TERMINAL    ...                    Run: VideoGame

Microsoft Windows [Versión 10.0.22631.3593]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\UNIVERSIDAD\GIT\LAB-FP2\LAB10> cmd /C ""C:\Users\ASUS\AppData\Local\Programs
\Eclipse Adoptium\jdk-17.0.11.9-hotspot\bin\java.exe" -XX:+ShowCodeDetailsInExc
eptionMessages -cp C:\Users\ASUS\AppData\Roaming\Code\User\workspaceStorage\817
e03dc5cdb84bcc51ed9348ea76425\redhat.java\jdt_ws\LAB10_9abfcd1f\bin VideoGame "


1. Quick Game
2. Custom Game
3. Exit
Enter the corresponding number:
```

```
3. Exit
Enter the corresponding number: 1

You have selected Quick Game

Game Board

 ____________________________________________________________________________
|      |      |      |      |      |      |      |      |      |      |      |
|      |  A.  |  B.  |  C.  |  D.  |  E.  |  F.  |  G.  |  H.  |  I.  |  J.  |
||||||||||||
|      |      |      |      |      |      |      |      |      |      |      |
|  01  |      |      |      |      |      |      |      |      |      |      |
||||||||||||
|      |      |      |      |      |      |      |      |      |      |      |
|  02  |  03  |      |      |      |      |  03  |  04  |      |      |      |
||||||||||||
|      |      |      |      |      |      |      |      |      |      |      |
|  03  |      |      |      |      |      |      |      |  03  |      |      |
||||||||||||
|      |      |      |      |      |      |      |      |      |      |      |
|  04  |      |      |      |      |      |      |  01  |      |      |      |
||||||||||||
|      |      |      |      |      |      |      |      |      |      |      |
|  05  |      |      |      |      |      |      |      |      |      |      |
||||||||||||
|      |      |      |      |      |      |      |      |      |      |      |
|  06  |      |      |      |      |      |      |      |      |      |      |
||||||||||||
|      |      |      |      |      |      |      |      |      |      |      |
|  07  |      |      |      |      |      |      |      |      |      |      |
||||||||||||
|      |      |      |      |      |      |      |      |      |      |      |
|  08  |      |      |      |      |      |      |      |      |      |      |
||||||||||||
|      |      |      |      |      |      |      |      |      |      |      |
|  09  |      |      |      |      |  01  |      |      |      |      |      |
||||||||||||
|      |      |      |      |      |      |      |      |      |      |      |
|  10  |      |      |  03  |      |      |      |  04  |      |  05  |      |
||||||||||||
```