Exámen métodos de programación

José Mauricio Muñoz Diéguez

Un pequeño resumen del proyecto...

Casino ATS el cual contiene varios juegos:

- Blackjack
- Connecta 4
- Adivina Adivinador
- Tragamonedas

Organización del código

Todo en sus respectivas clases separadas por carpetas

Games: Contiene las clases de los 4 juegos.
Util: Clases de utilidad para realizar diferentes acciones (limpiar consola, imprimir con colores, etc.)

Root: Contiene la clase Main, User y Game.

Menú (`Casino.java`)

```java
public static void main(String[] args) {

    ConsoleUtil.clearConsole();

    System.out.println(Color.BLUE + "Ingresa tu nombre de usuario:" + Color.RESET);
    String username = scanner.nextLine();

    ConsoleUtil.clearConsole();

    System.out.println("----------------------------------------");
    System.out.println(Color.RED + "Bienvenido, " + username + " al Casino ATS!" + Color.RESET);
    System.out.println("----------------------------------------");

    float balance;

    do {
      System.out.println(Color.BLUE + "Ingresa tu saldo inicial." + Color.RESET
          + " (Debe ser mayor a $50.00 y menor a $1000.00):");
      balance = scanner.nextFloat();
    } while (balance < 50.0 || balance > 1000.0);

    User user = new User(username, balance);

    handleMenu(user);

}
```

```
...

choice = scanner.nextint();

consoleutil.clearconsole();

switch (choice) {
  case 1:
    handlegame(user);
    break;
  case 2:
    system.out.println("\ntu saldo es: " + color.green + user.getbalance() + color.reset);
    consoleutil.pressanykeytocontinue();
    consoleutil.clearconsole();
    break;
  case 3:
    user.playhistory.printPlayHistory(10);
    ConsoleUtil.pressAnyKeyToContinue();
    ConsoleUtil.clearConsole();
    break;
  case 4:
    System.out.println("\nGracias por jugar!");
    scanner.close();
    System.exit(0);
    break;
  default:
    System.out.println(Color.RED + "La opción que elegiste es inválida" + Color.RESET);
    break;
}
```

Menú de juegos

Ejemplo: **Tragamonedas**

```
switch (choice) {
  case 1:
    Slot slot = new Slot(user, bet);

    Game.handleInstructions(slot.NAME,
        "El programa mostrará 3 palabras aleatorias. Si 3...");

    while (slot.status != Game.GameStatus.COMPLETED) {
      slot.play();
    }
    ConsoleUtil.clearConsole();
    break;
}
```

Otros juegos que son un poco diferente...

Ejemplo: **Blackjack**

```
while (blackjack.status != Game.GameStatus.COMPLETED) {
  blackjack.play();
  // Ask if the user wants to play another round
  System.out.println("\nDeseas jugar otra ro...");
  String ans = scanner.next();
  if (ans.equals("n")) {
    blackjack.finishGame();
    break;
  }
}
```

Un vistazo a las clases de utilidad

Color.java

Clase estática que permite impimir en consola a color
(No funciona en Windows)

```
public static final String GREEN = "\u001B[32m";
```

ConsoleUtil.java

Clase estática que ayuda a limpiar la consola y a pedir al usuario que presione `enter` para continuar

```java
// Limpia la consola
public static void clearConsole() {
  System.out.print("\033[H\033[2J");
  System.out.flush();
}

// Le indica al usuario que espere para dar enter
public static void pressAnyKeyToContinue() {
  System.out.println("\nPresiona [enter] para continuar...");
  try {
    System.in.read();
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

PlayHistory.java

Una clase que guarda todas las partidas que he hecho, si gané o perdí, y cuánto diero se me sumó o restó

```
List<PlayInfo> playHistory;
```

Donde `PlayInfo` es una clase que contiene...

```
String game; // El nombre del juego
float bet; // La apuesta ganada/perdida
Game.WinLoseStatus status; // Si ganó, perdió o quedó empate
```

`Stopwatch.java`

Clase de utilidad que permite tener un cronometro y tiene funciones como `elapsedTime()` o `reset()`

Utiliza por dentro `System.currentMillis()`

Dejando a un lado las clases de utilidad...

Clase `Game.java`

Una clase con funciones estáticas que contienen lógica que comparten todos los juegos

- Status
- Lógica cuando se gana
- Lógica cuando se pierde
- Etc.

Cuando el usuario gana algún juego...

```
public static void handleWin(User user, float bet, float ratio, String gameName) {
  float oldBalance = user.getBalance();

  float balanceToAdd = bet * ratio - bet;

  user.addBalance(balanceToAdd);

  System.out
      .println(
          Color.GREEN + "\n\nFelicidades, has ganado " + balanceToAdd + "$");
  System.out.println("Saldo Anterior: " +  oldBalance + "$");
  System.out.println("Saldo Actual: "+ user.getBalance() + "$\n");
  user.playHistory.addPlay(gameName, bet, WinLoseStatus.WIN);
  ConsoleUtil.pressAnyKeyToContinue();
}
```

Función que llama `handleLose()` para burlarse del jugador...

```
public static void makeLoserSentence() {
  int randomIndex = new Random().nextInt(loserStrings.length);
  System.out.println(loserStrings[randomIndex]);
}
```

Listado de todas las funciones:

- `handleBet()` : Se encarga de pedirle al usuario la apuesta a realizar y verifica si es válida
- `handleWin()` : Muestra en consola que el usuario ganó cierta cantidad de dinero. Lo modifica de la clase `User`
- `handleLose()` : Lo mismo que `handleWin()` solo que resta dinero y manda a llamar a `makeLoserSentence()`
- `handleDraw()` : Lo mismo que las dos anteriores solo que no te quita dinero
- `makeLoserSentence()` : Se burla del usuario utilizando frases pre-escritas
- `handleInstructions()` : Muestra en consolas las instrucciones de un juego determinado

Clase `User.java`

Contiene información básica del usuario como el nombre, salario y su historial de jugadas.

```
public class User {

  String name;
  float balance;
  PlayHistory playHistory;
}
```

Cuenta también con funciones para accesar y modificar cada una de ellas.

Ahora si lo divertido...

Las clases de los juegos

Antes de iniciar...

Todas las clases de Juegos comparten estos atributos:

```
public final String NAME = "Adivina Adivinador";

public Game.GameStatus status = Game.GameStatus.NOT_STARTED;

private float bet;
private User user;
```

Clase `Guess.java`

(Adivina Adivinador)

Constructor y atributos

```
private int numberToGuess;
private int tries = 0;

public Guess(User user, float bet) {
  this.bet = bet;
  this.user = user;
  numberToGuess = new Random().nextInt(MAX_NUMBER) + 1;
}
```

Función `play()`

```
tries++;
if (number == numberToGuess) {
  // Print tries
  System.out.println("\nAdivinaste en " + tries + " intentos");
  finishGame();
} else if (number > numberToGuess) {
  System.out.println("El numero es más pequeño");
} else {
  System.out.println("El numero es más grande");
}
```

Función `finishGame()`

```
public void finishGame() {
  status = Game.GameStatus.COMPLETED;
  if (tries == 1) {
    Game.handleWin(user, bet, 3, NAME);
  } else if (tries == 2) {
    Game.handleWin(user, bet, 2.5f, NAME);
  } else if (tries == 3) {
    Game.handleWin(user, bet, 2, NAME);
  } else if (tries >= 4 && tries <= 6) {
    Game.handleWin(user, bet, 1.5f, NAME);
  } else {
    Game.handleLose(user, bet, NAME);
  }
}
```

Clase `Blackjack.java`

Constructor y atributos

```
private int computerPoints = 0;
private int userPoints = 0;

public Blackjack(User user, float bet) {
  this.bet = bet;
  this.user = user;
}
```

Función `play()` y `throwDices()`

```java
public void play() {
  computerPoints += throwDices();
  userPoints += throwDices();
  System.out.println("Tu puntaje: " + userPoints);
}

private int throwDices() {
  Random random = new Random();
  int dice1 = random.nextInt(6) + 1;
  int dice2 = random.nextInt(6) + 1;
  return dice1 + dice2;
}
```

Función `finishGame()`

```
// Show computer points
System.out.println("Puntaje de la computadora: " + computerPoints);
if (userPoints > 21) {
  System.out.println("Te pasaste de 21" + computerPoints);
  Game.handleLose(user, bet, NAME);
} else if (computerPoints > 21) {
  Game.handleWin(user, bet, 2, NAME);
} else if (userPoints > computerPoints) {
  Game.handleWin(user, bet, 2, NAME);
} else if (userPoints < computerPoints) {
  Game.handleLose(user, bet, NAME);
} else {
  Game.handleDraw(user, NAME);
}
```

Clase `Slot.java`

(Tragamonedas)

Constructor y atributos

```
private int[] indexes = new int[3];

private final int DELAY_BETWEEN_WORDS_MS = 1000;
private final int DELAY_BETWEEN_SLOTS_MS = 100;

private final String[] words = new String[]
{
    "cerezas", "naranjas", "ciruelas",
    "campanas", "melones", "barras"
};

public Slot(User user, float bet) {
  this.bet = bet;
  this.user = user;
}
```

Quise hacer un poco más interesante...

```
Random random = new Random();

indexes[0] = random.nextInt(words.length);
indexes[1] = random.nextInt(words.length);
indexes[2] = random.nextInt(words.length);

int wordsShown = 0;

Stopwatch showWordStopwatch = new Stopwatch();
Stopwatch showRandomSloStopwatch = new Stopwatch();

showWordStopwatch.start();
showRandomSloStopwatch.start();
```

```
while (wordsShown <= 3) {

  if (showWordStopwatch.getElapsedTime() > DELAY_BETWEEN_WORDS_MS) {
    wordsShown++;
    showWordStopwatch.reset();
    showWordStopwatch.start();
  }

  if (showRandomSloStopwatch.getElapsedTime() > DELAY_BETWEEN_SLOTS_MS) {
    ConsoleUtil.clearConsole();
    System.out.print("| " + Color.BLUE);
    for (int i = 0; i < 3; i++) {
      int index = random.nextInt(words.length);
      // If we have already shown this word, don't show it randomly
      if (i < wordsShown) {
        index = indexes[i];
      }
      System.out.print(words[index] + Color.RESET + " | " + Color.BLUE);
    }
    showRandomSloStopwatch.reset();
    showRandomSloStopwatch.start();
  }
}
```

Función `finishGame()`

```
if (indexes[0] == indexes[1] && indexes[1] == indexes[2]) {
  Game.handleWin(user, bet, 3, NAME);
} else if (indexes[0] == indexes[1] || indexes[1] == indexes[2]) {
  Game.handleWin(user, bet, 2, NAME);
} else {
  Game.handleLose(user, bet, NAME);
}
```

Clase `Connect.java`

(Conecta 4)

Constructor y atributos

```
private enum Player {
  USER, COMPUTER, NONE
}

private final int ROWS = 7;
private final int COLS = 7;

private char userToken = '♥';
private char computerToken = '♠';

private char[][] board = new char[ROWS][COLS];

public Connect(User user, float bet) {
  this.bet = bet;
  this.user = user;
  initBoard();
}
```

Función `setToken()`

```
public void setToken(boolean heartToken) {
  if (heartToken) {
    userToken = '♥';
    computerToken = '♠';
  } else {
    userToken = '♠';
    computerToken = '♥';
  }
}
```

Función `init()`

```
public void init() {
  int computerColumn = new Random().nextInt(COLS);
  placePiece(computerColumn, Player.COMPUTER);
  drawBoard();
}
```

Función `play()`

```
Random random = new Random();

column -= 1;

if (column > COLS || column < 0) {
  System.out.println("Columna invalida");
  return;
}
```

```
placePiece(column, Player.USER);

if (status == Game.GameStatus.COMPLETED)
  return;

int computerColumn = random.nextInt(COLS);
while (board[0][computerColumn] != ' ') {
  computerColumn = random.nextInt(COLS);
}

placePiece(computerColumn, Player.COMPUTER);

drawBoard();
```

Función `placePiece()`

```
for (int i = ROWS - 1; i >= 0; i--) {
  if (board[i][column] == ' ') {
    board[i][column] = player == Player.USER ? 'X' : 'O';
    break;
  }
}
Player winner = checkWinner();
if (winner != Player.NONE) {
  finishGame(winner);
}
```

Función `checkWinner()`

```
int[] upperPiecesCount = new int[COLS];
int[] leftDiagonalPiecesCount = new int[COLS];
int[] rightDiagonalPiecesCount = new int[COLS];

for (int i = 0; i < COLS; i++) {
  upperPiecesCount[i] = 0;
  leftDiagonalPiecesCount[i] = 0;
  rightDiagonalPiecesCount[i] = 0;
}

int consecutivePieces = 1;
```

```
if (board[i][j] == ' ') {
  consecutivePieces = 0;
  upperPiecesCount[j] = 0;
  if (j > 0)
    leftDiagonalPiecesCount[j - 1] = 0;
  if (j < COLS - 1)
    rightDiagonalPiecesCount[j + 1] = 0;
  continue;
}

if (i == 0) {
  upperPiecesCount[j] = 1;
  if (j > 0)
    leftDiagonalPiecesCount[j - 1] = 1;
  if (j < COLS - 1)
    rightDiagonalPiecesCount[j + 1] = 1;
}

if (j == 0) {
  consecutivePieces = 1;
}
```

```
if (j > 0 && board[i][j - 1] == board[i][j]) {
  consecutivePieces++;
} else {
  consecutivePieces = 1;
}

// Check vertical
if (i > 0 && board[i - 1][j] == board[i][j]) {
  upperPiecesCount[j]++;
} else {
  upperPiecesCount[j] = 1;
}

// Check left diagonal
if (i > 0 && j > 0) {
  if (board[i - 1][j - 1] == board[i][j]) {
    leftDiagonalPiecesCount[j - 1]++;
  } else {
    leftDiagonalPiecesCount[j - 1] = 1;
  }
}
```

```
// Check right diagonal
if (i > 0 && j < COLS - 1) {
  if (board[i - 1][j + 1] == board[i][j]) {
    rightDiagonalPiecesCount[j + 1]++;
  } else {
    rightDiagonalPiecesCount[j + 1] = 1; // Aqui
  }
}

// Any 4 are consecutives
if (consecutivePieces == 4 || upperPiecesCount[j] == 4 ||
 (j > 0 && leftDiagonalPiecesCount[j - 1] == 4) ||
 (j < COLS - 1 && rightDiagonalPiecesCount[j + 1] == 4)) {
  return board[i][j] == 'X' ? Player.USER : Player.COMPUTER;
}
```

Función `drawBoard()`

```
ConsoleUtil.clearConsole();
for (int i = 0; i < ROWS; i++) {
  System.out.print("| ");
  for (int j = 0; j < COLS; j++) {
    if (board[i][j] == 'X') {
      System.out.print(Color.GREEN + userToken + Color.RESET);
    } else if (board[i][j] == 'O') {
      System.out.print(Color.RED + computerToken + Color.RESET);
    } else {
      System.out.print(board[i][j]);
    }
    System.out.print(" | ");
  }
  System.out.print('\n');
}
System.out.print('\n');
```

Hora de un demo:)