Acosta Porcayo Alan Omar, Gutiérrez Grimaldo Alejandro, Medina Villa Samuel

Práctica 5. Abstracción y Encapsulamiento

Resumen

Introducción

Objetivos

-

Metodología

Ejercicio realizado por el profesor

Ejemplo de uso de interfaces.

Código

```
interface Poligono {
  // firma de metodos
  void getArea(int a, int b);
}

class Rectangulo implements Poligono {
  // implementacion del metodo de la interfaz
  public void getArea(int a, int b) {
    System.out.println("El area del rectangulo es: " + (a * b));
  }
}

class Main {
  public static void main(String[] args) {
    Rectangulo r = new Rectangulo();
    r.getArea(4, 8);
  }
}
```

Recreación del juego “Gato”.

```
import java.util.*;

interface Board {
  // En interfaz por lo general los metodos son default
  String checkWinner();
  void printBoard();
}

public class TicTacToe implements Board {
  static String[] board;
  static String player; // Posible objeto que venga de la clase Player L11

  // Override
  public String checkWinner() {
    for(int i = 0; i <8; i ++) {
      String line = null;
      switch(i) {
        case 0: line = board[0] + board[1] + board[2]; break;
        case 1: line = board[3] + board[4] + board[5]; break;
        case 2: line = board[6] + board[7] + board[8]; break;
        case 3: line = board[0] + board[3] + board[6]; break;
        case 4: line = board[1] + board[4] + board[7]; break;
        case 5: line = board[2] + board[5] + board[8]; break;
        case 6: line = board[0] + board[4] + board[8]; break;
        case 7: line = board[2] + board[4] + board[6]; break;
      }

      if(line.equals("XXX")) // Si gana el jugador X
        return "X";
      else if(line.equals("OOO")) // Si gana el jugador O
        return "O";
    }

    for(int a = 0; a < 9; a ++) {
      if(Arrays.asList(board).contains(String.valueOf(a + 1)))
          break;
      else if(a==8)
          return "DRAW";
    }

    System.out.println("\nEs el turneo de " + player + ", ingrese una casilla: ");

    return null;
  }

  //imprimir
  /*
  |-------|-------|-------|
  |   1   |   2   |   3   |
  |-------|-------|-------|
  |   4   |   5   |   6   |
  |-------|-------|-------|
  |   7   |   8   |   9   |
  */

  public void printBoard() {
    System.out.println("\n|---|---|---|");
    System.out.println("| " + board[0] + " | " + board[1] + " | " + board[2] + " |");
    System.out.println("|---|---|---|");
    System.out.println("| " + board[3] + " | " + board[4] + " | " + board[5] + " |");
    System.out.println("|---|---|---|");
    System.out.println("| " + board[6] + " | " + board[7] + " | " + board[8] + " |");
    System.out.println("|---|---|---|");
  }

  public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    board = new String[9];
    player = "X";
    String winner = null;

    TicTacToe t = new TicTacToe();

    // Llenar la matriz board
    for(int i=0;i<9;i++)
      board[i] = String.valueOf(i+1);

    System.out.println("Bienvenido Tic Tac Toe 3x3");

    t.printBoard();

    System.out.println("Es el turneo de " + player + ", ingrese una casilla: ");
    while(winner == null) {
      int numSlot;

      numSlot = in.nextInt();
      if(!(numSlot>0 && numSlot<=9)) {
        System.out.println("Opcion no valida");
        continue;
      }

      if(board[numSlot-1].equals(String.valueOf(numSlot))) {
        board[numSlot-1] = player; // Player vale "X"
        if(player.equals("X"))
          player = "O";
        else
          player = "X";
        t.printBoard();
        winner = t.checkWinner();
      } else
        System.out.println("El slot ya esta ocupado");
    }
```

Acosta Porcayo Alan Omar Ing. en Computación 320206102
Gutiérrez Grimaldo Alejandro Ing. en Computación 320282098
Medina Villa Samuel Ing. en Computación 320249538

```
    if(winner.equals("DRAW"))
      System.out.println("Nadie gana. Gracias por jugar");
    else
      System.out.println("Ganaste " + winner + " eres un PRO");

    in.close();
  }
}
```

Resultados

Problema 1

Modifique el programa de *Tic Tac Toe* haciendo:

a. El Juego de 5×5 y el que gane lo haga con 4 símbolos unidos.

b. Cree una clase abstracta que sea el jugador (X-O) y herede la clase principal.

Implementar la herencia hibrida, usar interfaces y clases abstractas.

Explicación

Código

Ejecución

Problema 2

Realice un verificador de contraseña usando encapsulamiento de tipo *private* y *protected*. El programa recibe una cadena candidata para ser contraseña y se valida que:

a. Tenga entre 8 y 16 de caracteres

b. Que tenga al menos un carácter especial.

c. Que tenga al menos un número.

d. Que tenga al menos una letra minúscula.

e. Que tenga al menos una letra mayúscula.

Investigue como se hace para ocultar los caracteres escritos en la consola.

Al final regrese si la contraseña es débil, mediana o fuerte.

Implementar encapsulamiento privado y protegido.

Explicación

Código

Ejecución

Problema 3

Programa un ahorcado utilizando la composición de objetos:

a. La palabra adivinanza tiene que ser seleccionada de manera aleatoria de un arreglo de mínimo 30 palabras.

b. El ahorcado deberá mostrarte en consola con una figura simple por ejemplo

Palabra oculta: “MATEMATICAS”
Adivina: _ _ _ _ _ _ _ _ _ _ _

c. Si el usuario gana mostrar un mensaje de ganador y de la misma manera si se pierde.

d. Considere las clases:

i. Palabra (palabras ocultas)

ii. Ahorcado (figura)

iii. *Main* (se programa la lógica del juego)

Implementar composición de objetos.

Explicación

Para programar el juego de ahorcado se utilizaron las 3 clases sugeridas. La clase *Palabra* se encarga de elegir aleatoriamente una palabra entre las 30 posibles para crear dos variables, una con la palabra que se tiene que adivinar y otra con guiones bajos (_) que el usuario ira llenando con sus intentos. Esta clase tiene dos métodos *getters* para saber la palabra y la palabra escondida y otro método para comprobar si la letra que ingresó el usuario esta en la palabra original para agregar la letra a la palabra escondida.

La clase *Ahorcado* se encarga de dibujar la figura simple del ahorcado y también lleva un control de las vidas del jugador por medio de un método *get* y uno para restar una vida en caso de equivocarse.

En la clase *Main* se realiza la composición de objetos de las clases *Palabra* y *Ahorcado*. Dentro de un ciclo *while* se muestra la palabra escondida con el progreso que lleva el jugador además de la figura del ahorcado. Se pide una letra y se revisa si está en la palabra original, y en caso de que no esté se resta una vida. El ciclo *while* se interrumpe cuando se acabaron las vidas o si se adivinó la palabra. Por último se imprime un mensaje de victoria o derrota dependiendo el caso.

Clase *Palabra*

```
import java.util.Random;

public class Palabra {
  private StringBuilder word = new StringBuilder();
  private StringBuilder hiddenWord = new StringBuilder();
  private Random rand = new Random();

  public Palabra() {
    int r = rand.nextInt(30);
    this.word = new StringBuilder(chooseWord(r).toUpperCase());

    for(int i = 0; i < word.length(); i++)
      this.hiddenWord.append('_');
  }

  private String chooseWord(int r) {
    switch(r) {
      case 1: return "personas";
      case 2: return "computadora";
      case 3: return "ciudad";
      case 4: return "sistema";
      case 5: return "historia";
      case 6: return "internet";
      case 7: return "perro";
      case 8: return "gato";
      case 9: return "casa";
      case 10: return "dia";
      case 11: return "noche";
      case 12: return "importante";
      case 13: return "pelicula";
      case 14: return "musica";
      case 15: return "libro";
      case 16: return "desarrollo";
      case 17: return "tiempo";
      case 18: return "programacion";
      case 19: return "cuerpo";
      case 20: return "problema";
      case 21: return "solucion";
      case 22: return "informacion";
      case 23: return "movimiento";
      case 24: return "lenguaje";
      case 25: return "actividad";
      case 26: return "luz";
      case 27: return "fuerza";
      case 28: return "nuevo";
      case 29: return "espacio";
      default: return "naturaleza";
    }
  }

  public StringBuilder getWord() {
    return this.word;
  }

  public StringBuilder getHiddenWord() {
    return this.hiddenWord;
  }

  public boolean guessLetter(char l) {
    boolean inWord = false;

    for(int i = 0; i < word.length(); i++) {
      if(word.charAt(i) == l) {
        this.hiddenWord.setCharAt(i, l);
        inWord = true;
      }
    }

    return inWord;
  }
}
```

Clase *Ahorcado*

```
public class Ahorcado {
  private int lives = 7;

  public void paintAhorcado() {
    for(int i = 0; i < lives; i++) {
      switch(i) {
        case 0: System.out.print(" O"); break;
        case 1: System.out.print("\n/"); break;
        case 2: System.out.print(" | "); break;
        case 3: System.out.print((char)92); break;
        case 4: System.out.print("\n |"); break;
        case 5: System.out.print("\n/ "); break;
        case 6: System.out.print((char)92); break;
      }
    }
  }

  public int getLives() {
    return this.lives;
  }

  public void loseLife() {
    this.lives--;
  }
}
```

Clase *Main*

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    Ahorcado game = new Ahorcado();
    Palabra p = new Palabra();
    StringBuilder w = new StringBuilder(p.getWord());

    while(game.getLives() > 0 && !p.getHiddenWord().toString().equals(w.toString())) {
      System.out.println("Adivina: " + p.getHiddenWord());
      game.paintAhorcado();

      System.out.print("\nLetra: ");
      char l = sc.next().charAt(0);
      l = Character.toUpperCase(l);
      if(!p.guessLetter(l))
        game.loseLife();
      System.out.println();
    }

    if(game.getLives() == 0)
      System.out.println("Perdiste :( La palabra era: " + w + "");
    else
      System.out.println("Adivinaste la palabra " + w + "!");

    sc.close();
  }
}
```

Ejecución

Problema 4

El matemático S. Ulam en 1964 planteo una secuencia de números entera que sigue una serie de reglas planteada en https://edabit.com/challenge/RkicZ4kkcSx8K3d4ex. Programe la secuencia de Ulam, pero que reciba desde consola los primeros dos elementos de la secuencia y se calcule sobre esos dos elementos.

```
Adivina: LEN__AJE
  0
/ | \
  |
Letra: m

Adivina: LEN__AJE
  0
/ | \
Letra: g

Adivina: LENG_AJE
  0
/ | \
Letra: u

Adinivaste la palabra LENGUAJE!
```

Por ejemplo:

(u, v)	sequence
(1, 2)	1, 2, 3, 4, 6, 8, 11, 13, 16, 18, ...
(1, 3)	1, 3, 4, 5, 6, 8, 10, 12, 17, 21, ...
(1, 4)	1, 4, 5, 6, 7, 8, 10, 16, 18, 19, ...
(1, 5)	1, 5, 6, 7, 8, 9, 10, 12, 20, 22, ...
(2, 3)	2, 3, 5, 7, 8, 9, 13, 14, 18, 19, ...
(2, 4)	2, 4, 6, 8, 12, 16, 22, 26, 32, 36, ...
(2, 5)	2, 5, 7, 9, 11, 12, 13, 15, 19, 23, ...

Para manejar la secuencia haga uso de ArrayList exclusivamente.

Explicación

El programa funciona a partir de los dos primeros números de la secuencia ingresados en la consola, los cuales se envían al método *ulam* que se encarga de calcular los siguientes 8 números.

En el método se utiliza la colección *ArrayList* para crear dos variables, la secuencia (*sequence*) y la suma de los elementos de la secuencia (*sums*). Primero se agregan los dos números ingresados por el usuario, luego, dentro de un ciclo *for*, se vacia la variable *sums* para volver a llenarla con los sumas de los elementos de la secuencia. Después, utilizando el método *sort* y la clase *Comparator* se ordena la lista de sumas para eliminar los elementos de *sums* que ya están en la secuencia.

Por último, dentro de un ciclo *while* se busca a la suma menor y que no este repetida a través de dos condicionales. La primera solo se aplica cuando la lista de sumas tiene un elemento, pues no tiene más con que comparar. La segunda comprueba si un elemento de *sums* está repetido, y si lo está procede a eliminar todas las repeticiones de este número. Si no se cumplieron las dos condicionales quiere decir que el elemento en 0 de *sums* es la suma mínima y no repetida y se agrega a la secuencia. Al terminar con los 8 elementos de la secuencia, se retorna y se imprime en el método *main*.

Código

```
import java.util.Scanner;
import java.util.ArrayList;
import java.util.Comparator;

public class Problema4 {
  static ArrayList<Integer> ulam(int u, int v) {
    ArrayList<Integer> sequence = new ArrayList<Integer>();
    ArrayList<Integer> sums = new ArrayList<Integer>();

    sequence.add(u);
    sequence.add(v);

    for(int i = 0; i < 8; i ++) {
      sums.clear();

      for(int j = 0; j < sequence.size(); j ++) {
        for(int k = j + 1; k < sequence.size(); k ++)
          sums.add(sequence.get(j) + sequence.get(k));
      }

      sums.sort(Comparator.naturalOrder());

      for(int j = 0; j < sums.size(); j ++) {
        if(sequence.contains(sums.get(j))) {
          sums.remove(j);
          j --;
        }
      }

      while(true) {
        if(sums.size() == 1) {
          sequence.add(sums.get(0));
          break;
        } else if(sums.get(0) == sums.get(1)) {
          int temp = sums.get(0);
          while(sums.get(0) == temp)
            sums.remove(0);
        } else {
          sequence.add(sums.get(0));
          break;
        }
      }
    }

    return sequence;
  }

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

    System.out.print("Numero 1: ");
    int u = sc.nextInt();
    System.out.print("Numero 2: ");
    int v = sc.nextInt();


    System.out.println("\nPrimeros 10 numeros de la secuencia: ");
    System.out.println(ulam(u, v));

    sc.close();
  }
}
```

Ejecución

```
Numero 1: 2
Numero 2: 4

Primeros 10 numeros de la secuencia:
[2, 4, 6, 8, 12, 16, 22, 26, 32, 36]
```

Conclusiones

Referencias

Solano, J. (2017, 20 enero). *Manual de prácticas de Programación Orientada a Objetos*. Laboratorio de Computación Salas A y B. http://lcp02.fi-b.unam.mx/

Matt. *Ulam Sequence*. Edabit. https://edabit.com/challenge/RkicZ4kkcSx8K3d4e

Java ArrayList sort(). (n.d.). https://www.programiz.com/java-programming/library/arraylist/sortx