Certamen 2, Programación II

Prof. Rodrigo Olivares
Ayud. Juan Carlos Tapia

Noviembre 3, 2015

Instrucciones:

- El puntaje máximo del certamen es 100%, siendo el 60% el mínimo requerido para aprobar.
- Responda cada pregunta en la hoja indicada, agregando su nombre. Si no responde alguna pregunta, debe entregar la hoja con su nombre e indicar que **no responde**.
- El certamen es **individual**. Cualquier intento de copia, será sancionado con nota **1,0**.

1. *30pts.* De las siguentes afirmaciones, encierre en un círculo la o las alternativas correctas.

i. Un constructor:
(a) No necesariamente debe ir en una clase.
(b) Al ser private sólo instancia objetos dentro de la clase.
(c) No puede ser sobrecargado.
(d) Si no se declara, se crea uno en tiempo de compilación
(e) Si no se declara, se crea uno en tiempo de ejecución

ii. La lectura de datos de la entrada estándar:
(a) Puede realizarse con la clase FileReader.
(b) Puede realizarse con la clase Scanner.
(c) Sólo puede ser de tipo String.
(d) Permite leer los argumentos del método *main*.
(e) Ninguna de las anteriores

iii. En relación a los arreglos:
(a) No almacenan los elementos de tipo char.
(b) Pueden almacenar tipos de datos primitivos.
(c) Pueden almacenar objetos.
(d) Se instancian sólo con el operador new.
(e) Su dimensión puede ser determinada en ejecución.

iv. Una clase abstracta:
(a) Puede instanciar objetos.
(b) Poseé sólo métodos abstractos.
(c) Poseé al menos un método abstracto.
(d) No permite extender una interfaz.
(e) Ninguna de las anteriores.

v. La interfaz List.
(a) Puede ser implementada por la clase ArrayVector.
(b) Puede instanciar objetos.
(c) Poseé sólo métodos abstractos.
(d) Puede ser extendida en una clase hija.
(e) No poseé atributos.

vi. Con respecto al paso de parámetros:
(a) Puede ser por valor.
(b) Puede ser por omisión.
(c) Puede ser por referencia.
(d) Puede ser por convensión.
(e) Puede ser por default.

vii. El relación a la auto-referencia:
(a) Cumple la misma función que el operador *super*.
(b) Permite referenciar el construtor de la clase.
(c) Permite referenciar los atributos de la clase.
(d) Permite referenciar los método de la clase.
(e) Ninguna de las anteriores.

viii. Sobre la herencia:
(a) Se realiza con la palabra reservada extends.
(b) Se realiza con la palabra reservada include.
(c) Todas las clases heredan de la clase Object.
(d) La clase padre hereda el comportamiento de la clase hija.
(e) La clase hija hereda el comportamiento de la clase padre.

xi. Respecto a la herencia múltiple:
(a) No existe en Java.
(b) Existe en Java.
(c) Se puede emular con clases abstractas.
(d) Se puede emular con interfaces.
(e) Se puede emular con clases estáticas.

x. En relación a la manipulación de archivos.
(a) Se lee un archivo con la instancia FileWriter.
(b) Se lee un archivo con la instancia FileReader.
(c) No es factible agregar contenido a un archivo existente.
(d) StringTokenizer se usa para archivos con delimitador.
(e) Ninguna de las anteriores.

2. *30pts.* Desarrolle un programa en Java que permita llenar un array con 10 nombres de personas, ingresados desde la entrada estándar. A partir de ese arreglo, construya un nuevo arreglo que almacene el largo de cada nombre. Por último, muestre el nombre y su largo, desde los arreglos ya creados.

```
import java.util.Scanner;

public class EjercicioNombre {

    private String[] nombres;
    private int[] largoNombnres;

    private static final int MAXIMO = 10;

    public EjercicioNombre() {
        nombres = new String[MAXIMO];
        largoNombnres = new int[MAXIMO];
    }

    public void llenar() {
        Scanner sc = new Scanner(System.in);

        for (int i = 0; i < MAXIMO ;i++) {
            System.out.print("Ingrese un nombre: ");
            nombres[i] = sc.nextLine();
        }
        System.out.println();
    }

    public void largos() {
        for (int i = 0; i < MAXIMO; i++) {
            largoNombnres[i] = nombres[i].length();
        }
    }

    public void mostrar() {
        for (int i = 0; i < MAXIMO; i++) {
            System.out.println("El nombre " + nombres[i] + " tiene un largo de " + largoNombnres[i] + " caracteres");
        }
    }

    public static void main(String[] args) {
        EjercicioNombre en = new EjercicioNombre();
        en.llenar();
        en.largos();
        en.mostrar();
    }
}
```

3. *40pts.* El BancoPais le ha solicitado implementar un sistema de seguridad de transferencias bancarías, utilizando un "digipass". Este dispositivo trabaja con una combinación aleatoria de 3 pares de números (entre el 10 y el 99, inclusive) de un total de 50 posibles valores, no necesariamente distintos. Estos números deben estar almacenados en una fuente de datos permanente, único por cliente. Diseñe un programa en Java que gestione una transferencia bancaria, incluyendo el monto a transferir e ingresando un código de "digipass" y lo compare con cualquier combinación de los valores almacenados. Debe informar al usuario (por la salida estándar) si la transferencia se realizó correcta o incorrectamente, informando para este caso, los errores comentidos.

```
import java.io.File;
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import java.util.ArrayList;

public class FuenteDatos {

    public static List<String> leer(String nombreArchivo) {
        File archivo;
        FileReader fileReader = null;
        List<String> lineas = null;
        String linea;

        try {
            archivo = new File(nombreArchivo);
            fileReader = new FileReader(archivo);
            BufferedReader br = new BufferedReader(fileReader);
            lineas = new ArrayList<String>();

            while ((linea = br.readLine()) != null) {
                lineas.add(linea);
            }
        } catch (IOException e) {
            System.out.println(e);
        } finally {
            try {
                if (fileReader != null) {
                    fileReader.close();
                }
            } catch (IOException e) {
                System.out.println(e);
            }
        }
        return lineas;
    }

    public static void escribir(String nombreArchivo, List<String> lineas) {
        FileWriter archivo;
        PrintWriter printWriter = null;
        try {
            if (!existe(nombreArchivo)) {
                archivo = new FileWriter(nombreArchivo);
                printWriter = new PrintWriter(archivo);
                for (String linea : lineas) {
                    printWriter.println(linea);
                }
            }
        } catch (IOException e) {
            System.out.println(e);
        } finally {
            printWriter.close();
        }
    }

    public static boolean existe(String nombreArchivo) {
        return new File(nombreArchivo).exists();
    }
}

*****************************************************************

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class Digipass {
    public static final String NOMBRE_ARCHIVO = "C:\\digipass.txt";
    private static final int MAX_NUMEROS = 50;
```

```
    public void crear() {
        if (!FuenteDatos.existe(NOMBRE_ARCHIVO)) {
            Random rand = new Random();
            List<String> aleatoriosDigipass = new ArrayList<String>();
            for (int i = 0; i < MAX_NUMEROS; i++) {
                aleatoriosDigipass.add(String.valueOf(rand.nextInt(90) + 10));
            }
            FuenteDatos.escribir(NOMBRE_ARCHIVO, aleatoriosDigipass);
        }
    }
}

**************************************************************

import java.util.List;
import java.util.Scanner;

public class Transferencia {
    private final String[] digipass;
    private final int num = 3;

    public Transferencia() {
        digipass = new String[num];
    }

    private void leerDigipass() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Ingrese los valores del digipass");
        for (int i = 0; i < num; i++) {
            digipass[i] = sc.nextLine();
        }
    }

    private boolean validarDigipass() {
        boolean status = true;
        int i = 0;
        if (digipass.length > 0) {
            List<String> digipassStore = FuenteDatos.leer(Digipass.NOMBRE_ARCHIVO);

            while (i < num && status) {
                status = digipassStore.contains(digipass[i]);
                i++;
            }
        }
        return status;
    }

    private void transferir() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Digipass correcto. Ingrese el valor a transferir");
        String valor = sc.nextLine();
        System.out.println("Se ha trasnferido $" + valor);
    }

    private void noTransferir() {
        System.out.println("Digipass incorrecto. Imposible transferir");
    }

    public void realizarTransferencia() {
        leerDigipass();
        if (validarDigipass()) {
            transferir();
        } else {
            noTransferir();
        }
    }
}

**************************************************************

public class BancoPais {
    public static void main(String[] args) {
        Digipass dp = new Digipass();
        dp.crear();

        Transferencia tans = new Transferencia();
        tans.realizarTransferencia();
    }
}
```