Examen escrito de Programación 1

Escuela de Ingeniería y Arquitectura
Departamento de Informática e Ingeniería de Sistemas

3 de septiembre de 2014

- Disponer sobre la mesa en lugar visible un **documento de identificación** provisto de fotografía.
- Escribir **nombre y dos apellidos** en cada una de las hojas de papel que haya sobre la mesa.
- Comenzar a resolver cada uno de los problemas del examen **en una hoja diferente** para facilitar su corrección por profesores diferentes.
- El tiempo total previsto para realizar el examen es de **tres horas**.
- No está permitido utilizar dispositivos electrónicos de ningún tipo, ni consultar libros ni apuntes, excepto los dos documentos facilitados por los profesores de la asignatura: *Breve resumen del lenguaje Java* y *Resumen de las únicas clases Java predefinidas que pueden ser utilizadas en esta asignatura.*
- En todos los métodos a diseñar en este examen se valorará de forma destacada la adecuada especificación de los mismos.

Problema 1.º (1 punto)

En matemáticas recreativas, un *número repituno* es un número formado exclusivamente con el dígito 1. Así, 1, 11, 111 y 1111 son ejemplos de números repitunos.

Se define el *orden* de un número repituno como el número de cifras que lo componen. Así, el orden de 11 es 2 y el orden de 111111 es 6.

Se debe escribir el código del método `ordenRepituno` cuya especificación se muestra a continuación:

```java
/**
 * Pre:  n > 0
 * Post: Si «n» es un número repituno, ha devuelto el orden del mismo.
 *       En caso contrario, ha devuelto -1.
 */
public static int ordenRepituno(int n)
```

Problema 2.º (2 puntos)

En tenis, un set es una parte o manga independiente de un partido. Uno de los dos jugadores que lo disputan lo gana cuando ha conseguido anotarse al menos seis *juegos* y, además, ese número de juegos supera en dos al número de juegos de su rival.

Por ejemplo, un set con tanteos como 6–0, 6–3 o 6–4, habrían sido ganados por el primer jugador por haber alcanzado 6 juegos con una diferencia de dos con respecto a los de su rival. Un set con tanteos como 5–7, 6–8 o 7–9 habría sido ganado por el segundo jugador, por haber conseguido al menos 6 juegos, con una diferencia de dos con respecto a los de su rival. Sets cuyos tanteos fueran 3–5, 6–5 o 7–8 serían sets que todavía estarían disputándose, por no haber llegado ninguno de los dos jugadores al mínimo de seis juegos o por no tener uno de los jugadores una ventaja de dos juegos sobre su oponente.

Se debe escribir el código de una clase Java de nombre `Set`, cuyos objetos representan sets de un partido de tenis. La única información que van a gestionar es el nombre de los dos jugadores que lo disputan y el número de juegos ganados por cada uno de ellos.

El diagrama UML de la clase que se debe diseñar se muestra a continuación, seguido de explicaciones sobre sus atributos y métodos. En las especificaciones que siguen, cuando se utiliza un dato entero para distinguir a los jugadores, uno de ellos estará identificado con el entero 0 y el otro, con el entero 1.

examen.septiembre.Set
− nombres: String[] − juegos: **int**[]
+ Set(nombre0, nombre1: String) + Set(nombre0, nombre1: String; juegos0, juegos1: **int**) + juegos(jugador: **int**): **int** + anotarJuego(jugador: **int**): **void** + haGanado(jugador: **int**): **boolean** + boolean terminado(): **boolean** + ganador(): String

El atributo `nombres` de cada objeto de la clase `Set` almacena los nombres de cada jugador. En la componente indexada por 0, se almacenan el nombre del jugador identificado con el entero 0 y en la componente indexada por 1, el nombre del jugador identificado con el entero 1.

El atributo `juegos` de cada objeto de la clase `Set` almacena el número de juegos ganados por cada jugador en ese set. En la componente indexada por 0, se almacenan los juegos ganados por el jugador identificado con el entero 0 y en la componente indexada por 1, se almacenan los juegos ganados por el jugador identificado con el entero 1.

La clase tiene dos constructores: uno con dos parámetros de tipo `String`, que inicializa el atributo `nombres` con los nombres de los dos jugadores especificados por los valores de los parámetros `nombre0` y `nombre1`, y que inicializa el atributo `juegos` de forma que represente un set cuando comienza a jugarse. El segundo constructor tiene, además de los dos parámetros de tipo `String` que sirven para inicializar el atributo `nombres`, dos parámetros de tipo entero, que deben ser no negativos, representan, respectivamente, los juegos ganados por los jugadores 0 y 1 y sirven para inicializar el atributo `juegos`.

El método `juegos` devuelve el número de juegos que lleva ganados el jugador identificado por el valor del parámetro `jugador`, que debe ser 0 o 1.

El método `anotarJuego` incrementa en una unidad el número de juegos ganados por el jugador identificado por el entero `jugador`, que debe ser 0 o 1.

El método `haGanado` devuelve el valor booleano **`true`** si y solo si, en función de los juegos ganados por cada jugador, el jugador identificado por el entero `jugador` ha ganado el set. El valor del parámetro `jugador` de nuevo tiene que ser 0 o 1.

El método `terminado` devuelve el valor booleano **`true`** si y solo si, en función de los juegos ganados por cada jugador, el set ha terminado.

El método `ganador` devuelve el nombre del jugador ganador del set, en el caso de que este haya terminado. En el caso de que el set no haya terminado, devuelve la cadena vacía.

Se pide escribir el contenido completo del fichero de código fuente Java «`Set.java`», ubicado en el paquete `examen.septiembre`, cuyos métodos públicos tengan un comportamiento acorde con la especificación proporcionada en este enunciado. El código debe estar adecuadamente documentado.

Problema 3.º (3 puntos)

Se pide diseñar el código de los métodos `buscar`, `incrementarSetsGanados` y `escribirNombreMaximoGanador`, cuya especificación se muestra a continuación:

```
/**
 * Pre:  «nombreJugador» no es nulo, «numJugadores» es mayor o igual que 0 y
 *       no alcanza el número de componentes de la tabla «nombreJugadores» y
 *       para cada índice i entre 0 y numJugadores-1, nombreJugadores[i] no
 *       es nulo.
 * Post: Si entre las primeras «numJugadores» componentes de la tabla
 *       «nombreJugadores» se encontraba un dato igual a «nombreJugador», ha
 *       devuelto el índice de dicha componente. En caso contrario, ha
 *       devuelto el valor de «numJugadores» (es decir, el índice de la
 *       primera componente «libre» de la tabla).
 */
private static int buscar(String nombreJugador,
        String[] nombreJugadores, int numJugadores)
```

```
/**
 * Pre:  «nombreJugador» no es nulo, las tablas «nombreJugadores» y
 *       «setsGanados» tienen el mismo número de componentes, «numJugadores»
 *       es mayor o igual que 0 y no alcanza ese número de componentes y para
 *       cada índice i entre 0 y numJugadores-1, setsGanados[i] representa el
 *       número de sets ganados por el jugador cuyo nombre es
 *       nombreJugadores[i].
 * Post: Si entre las primeras «numJugadores» componentes de la tabla
 *       «nombreJugadores» se encuentra un dato igual a «nombreJugador», ha
 *       incrementado en 1 el valor de la componente correspondiente de la
 *       tabla «setsGanados» y ha devuelto «numJugadores». En caso contrario,
 *       ha añadido el nombre «nombreJugador» a la tabla «nombreJugadores»
 *       (en su posición «numJugadores») ha incrementado en 1 igualmente el
 *       valor de la componente correspondiente de la tabla «setsGanados» y
 *       ha devuelto el valor numJugadores+1.
 */
private static int incrementarSetsGanados(String nombreJugador,
        String[] nombreJugadores, int[] setsGanados, int numJugadores)
```

```java
/**
 * Pre:  Las tablas «nombreJugadores» y «setsGanados» tienen el mismo número
 *       de componentes, «numJugadores» está entre 1 y ese número de
 *       componentes y para cada índice «i» entre 0 y numJugadores-1,
 *       setsGanados[i] representa el número de sets ganados por el jugador
 *       cuyo nombre es nombreJugadores[i].
 * Post: Ha escrito en la pantalla el nombre del jugador con mayor
 *       número de sets ganados y el número de sets que ha ganado, con un
 *       formato como el siguiente:
 *
 *       Tenista que más sets ha ganado: Rafael Nadal
 *       Número total de sets ganados: 21
 */
private static void escribirNombreMaximoGanador(
        String[] nombreJugadores, int[] setsGanados, int numJugadores)
```

Problema 4.º (4 puntos)

Disponemos de varios ficheros de texto que almacenan los resultados de todos los partidos de tenis correspondientes a un determinado torneo. La estructura de dichos ficheros de texto es la siguiente: la información de cada partido se reparte en tres líneas. En las dos primeras, aparecen los nombres de los jugadores del partido: en la primera línea el del jugador 0 y en la segunda línea el del jugador 1. En la tercera línea, aparecen los tanteos de cada set jugado en el partido, consistiendo este en un par de números enteros que representan el número de juegos ganados por el jugador 0 y por el jugador 1, en ese orden. El número de sets de los que consta un partido de tenis es variable y está comprendido entre dos y cinco.

Dicha estructura, en notación BNF, es la siguiente:

```
<fichero_torneo_tenis> ::= <partido> { <partido> }
<partido> ::= <nombre_jugador_0> fin_de_línea
              <nombre_jugador_1> fin_de_línea
              { <set> } fin_de_línea
<set> ::= <juegos_jugador_0> <juegos_jugador_1>
<nombre_jugador_0> ::= literal_String
<nombre_jugador_1> ::= literal_String
<juegos_jugador_0> ::= literal_int
<juegos_jugador_1> ::= literal_int
```

A modo de ejemplo, se muestra a continuación el contenido de un fichero de texto denominado «`RolandGarros2014.txt`», que sigue dicho formato:

```
Agnieszka Radwańska
Zhang Shuai
6 3   6 0
Roger Federer
Lukáš Lacko
6 2   6 4   6 2
Serena Williams
Alizé Lim
6 2   6 1

...

Novak Đoković
Ernests Gulbis
6 3   6 3   3 6   6 3
Rafael Nadal
Andy Murray
6 3   6 2   6 1
Marija Šarapova
Simona Halep
6 4   5 7   6 4
Rafael Nadal
Novak Đoković
3 6   7 5   6 2   6 4
```

Se pide escribir un programa Java que solicite al operador el nombre de un fichero de texto que respete el formato establecido previamente y escriba en la pantalla el nombre del tenista que, según los datos contenidos en el fichero suministrado, más sets haya ganado en el torneo, junto con el número de estos sets.

A modo de ejemplo, se presenta a continuación una posible ejecución del programa:

```
Nombre de un fichero con resultados de un torneo: RolandGarros2014.txt
Tenista que más sets ha ganado: Rafael Nadal
Número total de sets ganados: 21
```

Al diseño de este método se le debe aplicar la metodología de diseño descendente utilizada en el curso. Se puede, y se recomienda, utilizar objetos de la clase `Set` y los métodos diseñados en el problema anterior.

Solución al problema 1.º

```
package examen.septiembre;

public class Repituno {

    /**
     * Pre:  n > 0
     * Post: Si «n» es un número repituno, ha devuelto el orden del mismo.
     *       En caso contrario, ha devuelto -1.
     */
    public static int ordenRepituno(int n) {
        int numCifras = 0;
        boolean esRepituno = true;
        while (n != 0 && esRepituno) {
            int ultimoDigito = n % 10;
            n = n / 10;
            numCifras++;
            esRepituno = (ultimoDigito == 1);
        }
        // n == 0 || !esRepituno

        if (esRepituno) {
            return numCifras;
        }
        else {
            return -1;
        }
    }
}
```

Solución al problema 2.º

```
package examen.septiembre;

/**
 * Los objetos de esta clase representan sets de un partido de tenis. La
 * única información que gestionan es el nombre de los jugadores y el número
 * de juegos ganados en el set por cada jugador. En los métodos que siguen,
 * cuando los jugadores se identifican a través de datos enteros, uno de los
 * jugadores estará identificado con el entero «0» y el otro, con el entero
 * «1».
 */
public class Set {

    /**
     * Número mínimo de juegos que tiene que anotarse un jugador para poder
     * ganar el set.
     */
    private static final int MIN_NUM_JUEGOS = 6;

    /**
     * Diferencia mínima de juegos que tiene que haber entre los dos
     * jugadores para que uno de ellos pueda ganar el set.
     */
    private static final int MIN_DIF_JUEGOS = 2;

    /**
     * Nombres de los jugadores. En la componente indexada por 0, se
     * almacena el nombre del jugador identificado con el entero «0» y en la
     * componente indexada por 1, se almacena el nombre del jugador
     * identificado con el entero «1».
     */
    private String[] nombres;

    /**
     * Número de juegos ganados por los jugadores. En la componente indexada
     * por 0, se almacena el número de juegos ganados por el jugador
     * identificado con el entero «0» y en la componente indexada por 1, el
     * número de juegos ganados por el jugador identificado con el entero
     * «1».
     */
    private int[] juegos;
```

```java
/**
 * Pre:  ---
 * Post: Ha inicializado los atributos de este objeto con los nombres de
 *       los dos jugadores y de forma que represente un set cuando
 *       comienza a jugarse.
 */
public Set(String nombre0, String nombre1) {
    this.nombres = new String[] { nombre0, nombre1 };
    this.juegos = new int[] { 0, 0 };
}

/**
 * Pre:  juegos0 >= 0 y juegos1 >= 0.
 * Post: Ha inicializado los atributos de este objeto para que
 *       represente un set en el que el jugador «0» se llama «nombre0» y
 *       tiene «juegos0» juegos ganados y el jugador «1» se llama
 *       «nombre1» y tiene «juebos1» juegos ganados.
 */
public Set(String nombre0, String nombre1, int juegos0, int juegos1) {
    this.nombres = new String[] { nombre0, nombre1 };
    this.juegos = new int[] { juegos0, juegos1 };
}

/**
 * Pre:  jugador == 0 || jugador == 1
 * Post: Ha devuelto el número de juegos que lleva ganados el jugador
 *       identificado por el entero «jugador».
 */
public int juegos(int jugador) {
    return this.juegos[jugador];
}

/**
 * Pre:  jugador == 0 || jugador == 1
 * Post: Ha incrementado en una unidad el número de juegos ganados por
 *       el jugador identificado por el entero «jugador».
 */
public void anotarJuego(int jugador) {
    this.juegos[jugador]++;
}
```

```java
/**
 * Pre:  jugador == 0 || jugador == 1
 * Post: Ha devuelto true si y solo si, en función de los juegos ganados
 *       por cada jugador, el jugador identificado por el entero
 *       «jugador» ha ganado este set.
 */
public boolean haGanado(int jugador) {
    // Ha ganado si tiene 6 juegos o más y 2 juegos más que su rival
    return this.juegos[jugador] >= MIN_NUM_JUEGOS
            && this.juegos[jugador] - this.juegos[elOtro(jugador)]
                    >= MIN_DIF_JUEGOS;
}

/**
 * Pre:  jugador == 0 || jugador == 1
 * Post: Dado el jugador identificado por «jugador», ha devuelto el
 *       entero que identifica al otro jugador (1, si jugador==0 o 0, si
 *       jugador==1).
 */
private int elOtro(int jugador) {
    return 1 - jugador;
}

/**
 * Pre:  ---
 * Post: Ha devuelto true si y solo si, en función de los juegos ganados
 *       por cada jugador, este set ha terminado.
 */
public boolean terminado() {
    return haGanado(0) || haGanado(1);
}
```

```
    /**
     * Pre:  ---
     * Post: Si este set ha terminado, ha devuelto el nombre del jugador que
     *       lo ha ganado. En caso de que el set no haya terminado, ha
     *       devuelto la cadena vacía.
     */
    public String ganador() {
        if (haGanado(0)) {
            return this.nombres[0];
        }
        else if (haGanado(1)) {
            return this.nombres[1];
        }
        else {
            return "";
        }
    }
}
```

Solución a los problemas 3.º y 4.º

```
package examen.septiembre;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

/**
 * Esta clase dispone de un método «main» que, al ser invocado, solicita al
 * operador el nombre de un fichero que almacena los resultados de un torneo
 * de tenis con el formato establecido en el enunciado y, tras leerlo,
 * escribe en la pantalla el nombre del tenista que más sets ha ganado en
 * dicho torneo y el número de sets que ha ganado.
 */
public class Problema {

    /**
     * Número máximo de jugadores distintos que puede haber en el torneo
     */
    private static final int MAX_JUGADORES = 100;
```

```java
/**
 * Pre:  «nombreJugador» no es nulo, «numJugadores» es mayor o igual que
 *       0 y no alcanza el número de componentes de la tabla
 *       «nombreJugadores» y para cada índice i entre 0 y
 *       numJugadores-1, nombreJugadores[i] no es nulo.
 * Post: Si entre las primeras «numJugadores» componentes de la tabla
 *       «nombreJugadores» se encontraba un dato igual a
 *       «nombreJugador», ha devuelto el índice de dicha componente. En
 *       caso contrario, ha devuelto el valor de «numJugadores» (es
 *       decir, el índice de la primera componente «libre» de la tabla).
 */
private static int buscar(String nombreJugador,
        String[] nombreJugadores, int numJugadores) {
    // Esquema de búsqueda sin garantía de éxito
    int indice = 0;
    while (indice < numJugadores
            && !nombreJugadores[indice].equals(nombreJugador)) {
        indice++;
    }
    // indice == numJugadores
    //         || nombreJugadores[indice].equals(nombreJugador)
    return indice;
}
```

```java
/**
 * Pre:  «nombreJugador» no es nulo, las tablas «nombreJugadores» y
 *       «setsGanados» tienen el mismo número de componentes,
 *       «numJugadores» es mayor o igual que 0 y no alcanza ese número
 *       de componentes  y para cada índice «i» entre 0 y
 *       numJugadores-1, setsGanados[i] representa el número de sets
 *       ganados por el jugador cuyo nombre es nombreJugadores[i].
 * Post: Si entre las primeras «numJugadores» componentes de la tabla
 *       «nombreJugadores» se encuentra un dato igual a «nombreJugador»,
 *       ha incrementado en 1 el valor de la componente correspondiente
 *       de la tabla «setsGanados» y ha devuelto «numJugadores».
 *       En caso contrario, ha añadido el nombre «nombreJugador» a la
 *       tabla «nombreJugadores» (en su posición «numJugadores») ha
 *       incrementado en 1 igualmente el valor de la componente
 *       correspondiente de la tabla «setsGanados» y ha devuelto el
 *       valor numJugadores+1.
 */
private static int incrementarSetsGanados(String nombreJugador,
        String[] nombreJugadores, int[] setsGanados, int numJugadores) {
    int indice = buscar(nombreJugador, nombreJugadores, numJugadores);
    if (indice == numJugadores) {
        // Se trata de un nuevo jugador que no estaba en la tabla
        // «nombreJugadores»
        nombreJugadores[indice] = nombreJugador;
        numJugadores++;
    }
    setsGanados[indice]++;
    return numJugadores;
}
```

```java
/**
 * Pre:  Las tablas «nombreJugadores» y «setsGanados» tienen el mismo
 *       número de componentes, «numJugadores» está entre 1 y ese número
 *       de componentes y para cada índice «i» entre 0 y
 *       numJugadores-1, setsGanados[i] representa el número de sets
 *       ganados por el jugador cuyo nombre es nombreJugadores[i].
 * Post: Ha escrito en la pantalla el nombre del jugador con mayor
 *       número de sets ganados y el número de sets que ha ganado, con
 *       un formato como el siguiente:
 *
 *       Tenista que más sets ha ganado: Rafael Nadal
 *       Número total de sets ganados: 21
 */
private static void escribirNombreMaximoGanador(
        String[] nombreJugadores, int[] setsGanados, int numJugadores) {
    // Cálculo del índice donde está el máximo
    int indiceMaximo = 0;
    int i = 1;
    while (i < numJugadores) {
        if (setsGanados[i] > setsGanados[indiceMaximo]) {
            indiceMaximo = i;
        }
        i++;
    }
    // i == numJugadores

    // Escritura del tenista con más sets ganados y el número de sets
    // que ha ganado
    System.out.println("Tenista␣que␣más␣sets␣ha␣ganado:␣"
                       + nombreJugadores[indiceMaximo]);
    System.out.println("Número␣total␣de␣sets␣ganados:␣"
                       + setsGanados[indiceMaximo]);
}
```

```
/**
 * Pre:  ---
 * Post: Ha solicitado al operador el nombre de un fichero que almacena
 *       los resultados de un torneo de tenis con el formato establecido
 *       en el enunciado y, tras haberlo leído, ha escrito en la
 *       pantalla el nombre del tenista que más sets ha ganado en dicho
 *       torneo y el número de sets que ha ganado.
 */
public static void main(String[] args) {
    // Petición y lectura del nombre del fichero
    System.out.println("Escriba␣el␣nombre␣de␣un␣fichero␣"
            + "con␣resultados␣de␣un␣torneo:␣");
    @SuppressWarnings("resource")
    Scanner teclado = new Scanner(System.in);
    String nombreFichero = teclado.nextLine();

    try {
        // Creación de las estructuras de datos necesarias:
        String[] nombreJugadores = new String[MAX_JUGADORES];
        int[] setsGanados = new int[MAX_JUGADORES];
        int numJugadores = 0;

        // Lectura del fichero y cálculo del tenista con mayor número de
        // sets ganados
        Scanner fichero = new Scanner(new File(nombreFichero));
        while (fichero.hasNextLine()) {
            numJugadores = leerPartido(fichero, nombreJugadores,
                                       setsGanados, numJugadores);
        }
        fichero.close();

        // Escritura de los resultados
        escribirNombreMaximoGanador(nombreJugadores, setsGanados,
                                    numJugadores);
    }
    catch (FileNotFoundException ex) {
        System.out.println("Error:␣el␣fichero␣" + nombreFichero
                           + "␣no␣existe.");
    }
}
```

```
    /**
     * Pre:  «fichero» está abierto y en disposición de poder leerse de él
     *       un partido completo, de acuerdo con el formato establecido en
     *       el enunciado; las tablas «nombreJugadores» y «setsGanados»
     *       tienen el mismo número de componentes; «numJugadores» está
     *       entre 0 y ese número de componentes; y para cada índice i entre
     *       0 y numJugadores-1, setsGanados[i] representa el número de sets
     *       ganados por el jugador cuyo nombre es nombreJugadores[i].
     * Post: Ha leído un partido completo del «fichero», con la información
     *       de cada set del partido, ha actualizado las tablas
     *       «nombreJugadores» y «setsGanados», añadiendo el nombre de cada
     *       jugador que no estuviera en la tabla «nombreJugadores» a la
     *       misma, e incrementando la componente adecuada de «setsGanados».
     *       Ha devuelto el número de jugadores cuyos nombres y número de
     *       sets ganados están ahora almacenados en las tablas
     *       «nombreJugadores» y «setsGanados».
     */
    private static int leerPartido(Scanner fichero,
            String[] nombreJugadores, int[] setsGanados, int numJugadores) {
        // Primera línea del partido: nombre de un jugador
        String nombreJugador0 = fichero.nextLine();

        // Segunda línea del partido: nombre del otro jugador
        String nombreJugador1 = fichero.nextLine();

        // Tercera línea del partido: pares de enteros representando sets
        while (fichero.hasNextInt()) {
            // Para cada set, se leen los juegos de cada jugador
            int juegos0 = fichero.nextInt();
            int juegos1 = fichero.nextInt();

            // Se crea un objeto de la clase Set para obtener el nombre del
            // ganador y se incrementa el número de sets ganados por el
            // mismo, añadiéndolo a la tabla «nombreJugadores» si es preciso
            Set set = new Set(nombreJugador0, nombreJugador1,
                              juegos0, juegos1);
            numJugadores = incrementarSetsGanados(set.ganador(),
                               nombreJugadores, setsGanados, numJugadores);
        }
        fichero.nextLine(); // Se completa la lectura de la tercera línea
        return numJugadores;
    }
}
```