EXAMEN TAP

curso 2017/18 convocatoria extraordinaria (29/6/18)
FCT/ZTF – UPV/EHU

Los ejercicios se resolverán dentro de un mismo proyecto, cada uno en un paquete denominado: edu.upv.fct.examentap17_18extra.dni<su DNI>.ej<N>, donde N será el número del ejercicio.

1- ANÁLISIS DE UNA CLASE

Queremos tener a nuestra disposición en un programa en Java los datos bibliográficos que se encuentran en un fichero con el formato del ejemplo de la imagen.

```
102 t Flash disk opportunity for server applications
102 y 2008
102 p portal.acm.org
102 a B. Fitzgerald "Indiana University, indiana.edu, USA"
103 t Rotavirus G and P genotypes in rural Ghana
103 y 2001
103 p Am Soc Microbiol
103 r 4 26 40 5 8 3 6 96 31 201 231 29
102 a J. Gray "University of California Irvine, uci.edu, USA"
103 a J. Green "Harvard University, harvard.edu, USA"
103 a GE. Armah "University of Nebraska Lincoln, unl.edu, USA"
102 r 16 5 4 34 462 471 208 350 446 9 6 2 365 21 517 10 53 254 27
```

La primera columna es un número de registro bibliográfico; la segunda es una letra que indica qué es el resto de cada línea (t=título, y=año, p=publicación, r=referencias, a=autores con su filiacion, l=link). Los autores pueden ser más de uno en cada referencia, mientras que el resto de campos sólo pueden tener un valor.

```
public class Register {
  private final static Map<Integer,Register> LIBRARY = new TreeMap<>();

  List<String> authors;
  String title, year, publication, link, references;

  private Register(){ authors = new ArrayList<>(); }

  public static void addLine(String line) {
    String[] fields = line.split(" ",3);
    Integer id = null;

    if (fields.length!=3) throw new IllegalArgumentException("Wrong line format: "+line);
    if (!"typral".contains(fields[1])) throw new IllegalArgumentException("Wrong code: "+line);
    try { id = Integer.parseInt(fields[0]);
    } catch(NumberFormatException ex) {
      throw new IllegalArgumentException("Reference must be an integer: "+line);
    }

    Register reg=LIBRARY.get(id);
    if (reg==null) LIBRARY.put(id,reg=new Register());
    switch (fields[1]){
      case "t":reg.title=fields[2];break;
      case "y":reg.year=fields[2];break;
      case "p":reg.publication=fields[2];break;
      case "r":reg.references=fields[2];break;
      case "a":reg.authors.add(fields[2]);break;
      case "l":reg.link=fields[2];break;
    }
  }

  public static Map<Integer,Register> getMap(){ return LIBRARY; }

  public static void main(String[] args) throws IOException  {
    JFileChooser dialog = new JFileChooser(".");
    if (dialog.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {
      try (
        BufferedReader br =new BufferedReader(new FileReader(dialog.getSelectedFile()));
      ) {String linea; while ((linea = br.readLine()) != null) Register.addLine(linea);}
      System.out.println("File processed. " + Register.getMap().size() + " records loaded.");
    }
  }
}
```

Con este fin se ha escrito la clase "Register", que permite construir cada una de las referencias (agrupando todas las líneas con un mismo número en un mismo objeto de clase Register) y almacenarlas en una colección.

Explique qué hace este código con el mayor detalle posible (línea por línea cuando lo crea preciso) de modo que quede claro QUÉ hace y CÓMO lo hace, además de POR QUÉ se hace de ese modo. No comente lo obvio (p.ej. "asigna dato a reg.titulo").

Acceda a ambos ficheros en modo texto pinchando en las imágenes para hacer corta-pega en el proyecto que desarrolla como solución al examen. De este modo puede poner las explicaciones que se le piden como comentarios (ojo: se piden unos comentarios algo más extensos de lo que puede ser habitual encontrar en un fichero de código)

2- ESCRITURA DE ALGORITMOS - TEOREMA DE FERMAT

El conocido como Último Teorema de Fermat es uno de los teoremas más famosos en la historia de la matemática. Su enunciado es el siguiente:

$$\forall\, x,y,z,n \in \mathbb{N},\ x^n + y^n = z^n \ \Rightarrow\ n \leqslant 2$$

Si ***n*** es un número entero mayor que 2, entonces no existen números enteros positivos ***x***, ***y, z***, tales que se cumpla la igualdad $x^n + y^n = z^n$

Escriba dos rutinas, fermat2(int max) y fermat3(int max) que muestren por pantalla aquellas combinaciones de números enteros **x, y, z** en el rango **[1,max]**que cumplan las relaciones $x^2 + y^2 = z^2$ y $x^3 + y^3 = z^3$ respectivamente. Dado que se trata de una búsqueda exhaustiva, trate de minimizar los cálculos que realizan las rutinas. **Genere además una rutina principal** que compruebe el funcionamiento de ambas rutinas, la primera de ellas en el rango [1,20] y la segunda en el rango [1,2048]. La rutina pudiera mostrar lo siguiente por pantalla:

nota: los caracteres correspondientes a los números 2 y 3 como superindices son los codificados como \u00B2 y \u00B3

nota: Si encuentra un ejemplo que contradiga el Último Teorema de Fermat… revise el código!

```
run:
-----  x² + y² = z²   x,y,z ∈ [1,20]  -----
3² + 4² = 5²
5² + 12² = 13²
6² + 8² = 10²
8² + 15² = 17²
9² + 12² = 15²
12² + 16² = 20²
-----  x³ + y³ = z³   x,y,z ∈ [1,2048]  -----
BUILD SUCCESSFUL (total time: 0 seconds)
```

3- ESTRUCTURA DE UNA SOLUCIÓN – GESTIÓN DE UNA PEQUEÑA EMPRESA

Para desarrollar una aplicación que gestione una pequeña empresa, se hace un planteamiento centrado en la facturacion como sigue:

- Todas **facturas**, tanto de compras como de ventas se encontrarán en una tabla de datos en la que se consignarán con: el número de factura, la fecha, el NIF (número de identificación fiscal) del comprado/vendedor , el ID del vendedor o cero si se trata de una venta o una compra respectivamente, y si la operación está pagada o no.

```
//factura;      fecha;       nif;         ID; pagado
{00012018; 12/03/2018; 12345678A; 88888888Z;     si}
{032417;     7/7/2017;  A4356001;          0;     si}
...
```

```
//  denominacion; cantidad; precio;   factura
{ tuerca  W7Univ;     1500;   0.23; 00012018}
{barra acero 8mm;      100;   5.99;   032417}
...
```

- El desglose de todos los **elementos comprados o vendidos** se almacenarán en otra tabla donde cada elemento constará de la denominación del elemento comprado/vendido, el número de unidades, el precio por unidad sin IVA y la factura a la que corresponde.

- Los **productos/materiales** con que trabaja la empresa se encontrarán en otra tabla con la información: denominación, cantidad en stock, unidad en que se valora el precio, el porcentaje de IVA aplicable.

```
//  denominacion; cantidad; unidad; iva
{ tuerca  W7Univ;     9500; unidad;  21}
{barra acero 8mm;      500;  metro;  21}
...
```

- además de estas tablas se dispondrá de **otras** para caracterizar a **suministradores, clientes y trabajadores** que quedan a su criterio en cuanto a su composición (habrá otras, como cuentas bancarias, etc... que no tendremos en cuenta).

Toda esta información estará guardada en una base de datos pero se cargará sobre **las clases Java asociadas que debe usted escribir** (sólo el planteamiento de los campos, constructor, toString(), posibles interfaces, etc.; no la relación con la base de datos, que será escrita adecuadamente dentro de su clase por otra persona de la empresa).

Posteriormente será preciso escribir clases de control que permitan realizar diferentes procesos como obtener la información de una factura determinada, obtener las cantidades pendientes de cobro y de pago, las cantidades facturadas por cada vendedor en un periodo de tiempo dado, etc.... **Escriba al menos una clase de control que contenga al menos un método** (puede ser uno sencillo, como obtener por pantalla la lista items incluidos en una factura determinada)