1. JSON (<u>1 punto</u>)

Convierte el siguiente JSON en clases Java, no hace falta rellenar con datos, solo las clases con los campos sin métodos.


```
{
  "productoId": 12345,
  "nombre": "Smartphone XYZ Pro",
  "descripcion": "Último modelo de la marca XYZ con cámara de alta resolución y
batería de larga duración.",
  "precio": 799.99,
  "disponible": true,
  "especificaciones": {
    "peso": 190.5,
    "dimensiones": {
      "alto": 150.2,
      "ancho": 70.4,
      "profundidad": 8.1
    },
    "bateria": null,
    "coloresDisponibles": ["Negro", "Blanco", "Azul"]
  },
  "comentarios": [
    {
      "usuario": "AnaG",
      "comentario": "Excelente teléfono, la cámara es increíble.",
      "fecha": "2023-10-05T14:48:00.000Z",
      "calificacion": 5
    },
    {
      "usuario": "JuanP",
      "comentario": "Buen rendimiento pero la batería podría mejorar.",
      "fecha": "2023-10-06T09:20:00.000Z",
      "calificacion": 4
    }
  ],
  "fechaLanzamiento": "2023-09-01T00:00:00.000Z",
  "promociones": [
    {
      "tipo": "descuento",
      "valor": 10.0
    },
    {
      "tipo": "regalo",
      "valor": null
    }
  ]
}
```


2. Mostrar árbol de un directorio (2 puntos)

Crea un método llamado **printDirectoryTree**. El método recibirá una ruta. La ruta debe ser un directorio y mostrará la estructura de directorios y subdirectorios en un formato de árbol indentado en la consola. Haz las validaciones oportunas. Si alguna validación falla informa al usuario. Utiliza clases dentro del paquete java.nio.

```
ada
    level1
        level2
            level3
                test.bmp
    Tienda_Peliculas_jar
        README.md
        Tienda_Peliculas.jar
    vacia
```

Ejemplo:

- printDirectoryTree(tmp/ada) → imprime el árbol de la derecha. Las carpetas level1, Tienda_Peliculas_jar y vacia están en el mismo nivel.

Ayuda: puedes usar el método getNameCount() en Path. Este método devuelve un entero con el número de elementos que tiene el path. Esto me puede servir para indentar.

Ejemplo:

src\es\severo\ud1\examen\Test.java → devuelve que tiene un NameCount de 6, ya que hay 6 elementos en el path.

ada\level1 → devuelve 2

ada\level1\level2 → devuelve 3

3. RandomAccessFile - Encuentra el mensaje oculto (1 puntos)
Dado un fichero llamado file.txt que contiene en cada línea: caracteres + un número (2 dígitos) + un número (2 dígitos). Se pide averiguar la palabra secreta. Para ello utiliza los dos números, que indican la posición inicial inclusiva y final exclusiva de donde se encuentra el texto que contendrá letra/s del mensaje oculto final en la línea. Al obtener estos caracteres de cada línea llegarás al mensaje oculto final.
Muestra por pantalla el mensaje oculto. Solo se permite el uso de la clase RandomAccessFile. Para escribir las líneas se han escrito con los métodos writeBytes, así que **utiliza readLine para leer la línea**.
Ayuda: Lee línea a línea, una vez tengas la línea coge los dos números que están al final representados con dos dígitos. El último número estará a dos posiciones del final, y el penúltimo a 4. Estos números te servirán ya que contienen la posición de la letra/s ocultas de esa línea. Con los números ve a esa posición en la línea y extráela del string. Y así irás formando el mensaje para cada línea.
Ejemplo de una línea:
>hola soy patri0104 → para esta línea la cadena oculta del mensaje final es ***ola***, ya que estoy cogiendo los caracteres del [0 al 4).
Se hará esto para todas las líneas hasta obtener el mensaje oculto final.

Este es el fichero, cópialo y pégalo en tu IntelliJ:

```
tomate0001
hola patri estoy en clase de acceso a datos programacion2728
log servidor caido0405
log0102
el perro maxi0506
tengo un gato1213
```

4. Lectura y manipulación de ficheros (2 puntos)

Escribe el siguiente método: ***findTheMostRepeatedWord(Path path)***. Se le pasa un path, debe comprobar que exista y sea un fichero. Si lo es, deberás leerlo para encontrar dentro del fichero de texto, la palabra que se repite más veces. El método imprime la palabra y acaba.
Incluye manejo de excepciones.