Práctica Individual 1 – Ejercicios iterativos, recursivos lineales y notación funcional

A resolver en clases de prácticas por el profesor (<u>NO hay que incluirlos en la entrega</u>):

1.
```
public static String ejemplo1(Integer a, Integer b,
                    Function<Integer,Integer> f, String sp, String pf, String sf){
    return Stream.iterate(a, x->x<=b, x -> f.apply(x))
                    .map(x->x*x)
                    .map(x->x.toString())
                    .collect(Collectors.joining(sp,pf,sf));
}
```

2. Un punto es un tipo con las siguientes propiedades:

- X, Double, básica, individual
- Y, Double, básica, individual
- Cuadrante, Cuadrante, derivada, individual. Enumerado {PRIMER_CUADRANTE, SEGUNDO_CUADRANTE, TERCER_CUADRANTE, CUARTO_CUADRANTE}.

```
public static Map<Punto2D.Cuadrante,Double> ejemplo2(List<Punto2D> l){
    return l.stream()
            .collect(Collectors.groupingBy(Punto2D::getCuadrante,
            Collectors.<Punto2D,Double>reducing(0.,x->x.x(),(x,y)->x+y)));
}
```

3. Dadas 2 cadenas de caracteres A y B de la misma longitud, que cumplen que son iguales carácter a carácter hasta una determinada posición y distintas carácter a carácter a partir de dicha posición, determinar la primera posición en la que A y B son distintos. Por ejemplo A = “buenosdiaspepe” y B = “buenosdiasjuan”, devolvería la posición 10.

A resolver por los alumnos (<u>SÍ hay que incluirlos en la entrega</u>):

1.
```
public static boolean ejercicio1(List<String> ls, Predicate<String> pS,
                                 Predicate<Integer> pI, Function<String,Integer> f){
        return ls.stream()
                .filter(pS)
                .map(f)
                .anyMatch(pI);
}
```

2.
```
public static Map<Integer,List<String>> ejercicio2 (List<List<String>> listas) {
        return listas.stream()
                .flatMap(lista -> lista.stream())
                .collect(Collectors.groupingBy(String::length));
}
```

3.
```
public static String ejercicio3(Integer a, Integer limit) {
        return Stream
                .iterate(Par.of(0, a),
                t -> t.v1 < limit,
                t -> Par.of(t.v1+1, t.v1 % 3 == 1 ? t.v2 : t.v1+t.v2))
                .collect(Collectors.toList())
                .toString();


}
```

donde Par es una clase con 2 propiedades enteras v1 y v2, la cual debe implementar como un record.

4. Diseñe un algoritmo que dados dos números n y e (con n real positivo mayor que 1 y e real en el intervalo [0,1)), devuelva un número real que se corresponda con la raíz cúbica de n con un error menor que e.

Tenga en cuenta que:

- Para cada ejercicio debe leer los datos de entrada de un fichero, y mostrar la salida por pantalla. Dicha lectura debe ser independiente del algoritmo concreto que resuelva el ejercicio.
- La solución tiene que ser acorde al material de la asignatura proporcionado.

SE PIDE resolver de forma eficiente:

- Ejercicios 1, 2 y 3: Analice el código que se muestra y proporcione una solución iterativa y otra recursiva final equivalentes.
- Ejercicio 4: Proporcione una solución iterativa usando while, una recursiva final y una en notación funcional.

DEBE REALIZAR SU ENTREGA EN 2 PARTES:

1. Proyecto en eclipse con las soluciones en Java.
2. Memoria de la práctica en un único archivo PDF, que debe contener:
    - Código realizado
    - Volcado de pantalla con los resultados obtenidos para las pruebas realizadas, incluyendo al menos los resultados obtenidos para los tests proporcionados.