Ejercicio 2.3 Publicaciones

Paso uno: Análisis

- Lo primero que salta a la vista es o largo que es el método ultimosPosts(). Es tan largo que se necesitan dos comentarios para aclarar sus respectivas partes del código → Extract Function, Replace Inline Code with Function
- También podemos ver como se utilizan varias estructuras de iteración/loops a lo largo del cuerpo del método → Replace Loop with Pipeline
- Lo tercero para mencionar es que algunos de los métodos tendrían mas sentido estando en otras clases, como sería el caso de las listas de posts con los usuarios, lo que de paso le otorgaría utilidad a las otras dos clases, que están anémicas, pero esto requeriría replantear el diseño del programa y no sé si es lo que se busca. Consultar.

Paso dos: Refactoring

1. **Code Smell:** Método Largo → El método tiene mas de 10 líneas. Se pueden extraer tres secciones del código que cumple una misma función.
    a. **Refactoring a aplicar:** Extract Function → Crear un método para cada extracto. Colocar los respectivos parámetros y valores de retorno.
2. **Code Smell:** Loops → Los tres métodos usan sentencias for y while para realizar sus operaciones.
    a. **Refactoring a aplicar:** Replace Loop with Pipeline → Reemplazar el cuerpo de cada loop por una operación *stream()* equivalente

Resultado final

```
public class PostApp {
    private List<Post> posts;
```

```
    public List<Post> ultimosPosts(Usuario user, int cantidad) {
        List<Post> postsOtrosUsuarios = obtenerPostsOtrosUsuarios(user);
        ordenarPostsPorFecha(postsOtrosUsuarios);
        List<Post> ultimosPosts = obtenerUltimosPosts(cantidad, postsOtrosUsuari

        return ultimosPosts;
    }

    private List<Post> obtenerUltimosPosts(int cantidad, List<Post> postsOtrosUs
        return postsOtrosUsuarios.stream().limit(3).toList();
    }

    private void ordenarPostsPorFecha(List<Post> postsOtrosUsuarios) {
        postsOtrosUsuarios.stream().sorted((p1, p2) -> p1.getFecha().compareTo(p2
    }

    private List<Post> obtenerPostsOtrosUsuarios(Usuario user) {
        return this.posts.stream().filter(p -> p.getUsuario() != user).toList();
    }
}
```