# Ejercicio E

## Parte 1

Esta parte no cumple las siguientes características de clean code:

- En primer lugar, la clase Result no tiene un nombre significativo, ya que debería tener un nombre que diga la responsabilidad de la misma como Search
- En la clase SerchInfo, el atributo baseURL no es un nombre significativo y esta aclarado con un comentario así que lo mejor fue renombrarlo a wikiToSearch

```
this.baseURL = baseURL; //in which wiki we should search
```

- Luego, en el método srch de la clase SearchLogic además de que tiene demasiados comentarios también ocupa más de 10 líneas de código lo cual es demasiado extenso y el nombre tampoco es demasiado significativo. Las variables no tienen nombres significativos y no te da ninguna pista de para que sirve.

```
// GET Request to search in Wiki. Format: <base-wiki-url>/w/api.php?<actions>&srsearch=<search information>
// Example:
// https://en.wikipedia.org/w/api.php?action=query&format=json&list=search&srsearch=Baldurs%20Gate%203%20articletopic%3A%22video-games%22"
// Result example: { "result" : "[pageid1, pageid2, pageid3,...]" }
public String[] srch(SearchInfo searchInfo) {

    //This is used at the end to return the result
    String[] resPosta = new String[0];

    //String baseUrl = "https://en.wikipedia.org"; //Moving this to searchInfo, because reasons
    String rest = "action=query&format=json&list=search";

    String s ="https://"+ searchInfo.getBaseURL() + "/w/api.php?"+ rest +"&srsearch=" + searchInfo.getTextToSeach();

    if(searchInfo.getTopic() != null && !searchInfo.getTopic().equals(""))
        s+=" articletopic%3A\"" + searchInfo.getTopic() + "\"";

    // replaces spaces with encoding %20
    s = s.replace( target: " ",  replacement: "%20");
    // replaces " with encoding %22
    s = s.replace( target: "\"",  replacement: "%22");
```

```
    try {
        json = serviceProvider.resolveCall(new URI(s));
    } catch (URISyntaxException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    Result addUserResult = null;

    if (json != null) {
        Gson gson = new Gson();
        addUserResult = gson.fromJson(json, Result.class);
        resPosta = addUserResult.getResult();
    }

    return resPosta;
}
```

- ♥ A su vez, hice refactoring en varias partes del código:
  - El primero de ellos es replaceSpaces() el cual se encarga de reemplazar los espacios con “%20”
  - El segundo de ellos es replaceQuotationMarks() el cual reemplaza las comillas en “22%”
  - El tercero es jsonFormat() el cual se encarga de convertirlo en el formato de Json

## Parte 2

Esta parte no cumple las siguientes características de clean code:

- ♥ En la clase Club y Player todos los atributos no son significativos y tienen comentarios en base a que se refiere. La mejor forma de resolverlo es dar un nombre a los atributos directamente a que se refiere

```
private int id; // unique identifier
private String name; // team name
private String home; // stadium name
private int year; // founding year
private Image colours; // image depicting a shirt or the shield of the club
private List<Player> roster; // A list with the players of the first division team
private int wins; //The amount of matches that the team has won in the current season
private int loses; //The amount of matches that the team has lost in the current season
private int goalsf; //The amount of goals that the team made in the current season
private int goalsa; //The amount of goals that the team received in the current season
```

```
private String name; // song name
private int born; // birth year
```

- ♥ Algunos métodos tienen más de 2 parámetros lo cual no hace al método muy clean, ya que hay que evitarlos por lo que empeora la legibilidad. Para solucionar este problema lo mejor fue dividirlo en 3 métodos distintos con un parámetro cada uno y renombrarlos

```
public void addMatchInfo(boolean win, int gf, int ga){
    if(win) wins++;
    else loses++;
    goalsa += ga;
    goalsf += gf;
}
```

- ♥ Como la clase Club cumplía más de una responsabilidad, ya que tenía atributos que además de estar relacionadas al club también se relacionaban al team lo mejor fue extraer dicha clase creando una nueva clase llamada Team y acomodar los atributos en base a la responsabilidad de las clases
- ♥ Luego, en la clase Manager el atributo no era significativo y además en el método a parte de tampoco tener un nombre significativo tampoco lo tiene sus atributos locales y tiene demasiados comentarios

```
public Club get(int id) {

    Club a =  repo.getFilmLocalSource().getClub(id);

    // If it's not locally saved or too old, fetch it from remote and save it locally
    if(a == null || System.currentTimeMillis() - a.getTimeStamp() > ClubRepository.T * 24 * 60 * 1000) {
        //Replace these lines with calls to parte1 module, when they decide to finish it
        a = repo.getFilmRemoteSource().get(id);
        repo.getFilmLocalSource().storeClub(a);
    }

    return a;
}
```