

FacebookUser - Encapsulation Smell

[Link to chat](#)

Chat 1: Before being told that it was an Encapsulation Smell:

There are a few design smells in this code:

Exposing Fields: Direct access to class fields (`id`, `email`, `fullName`, `accessToken`) can lead to a lack of encapsulation and makes it difficult to enforce constraints or perform validation.

Inadequate Encapsulation: The class lacks appropriate encapsulation, as the fields are public. It's generally a good practice to encapsulate fields and provide accessors (getters) to access them.

Mutable State: The class is mutable, meaning its state can be changed after creation. This can lead to unexpected behavior, especially in concurrent environments.

Missing Validation: There's no validation of inputs in the constructor. This can lead to the creation of `FacebookUser` objects with invalid or inconsistent states.

Documentation: While the class has some comments, they are not particularly informative or helpful. Additionally, the comments are stating obvious things and are redundant.

Here's a refactored version addressing these issues:

```
package com.sismics.books.rest.model;

/**
 * Represents a Facebook user.
 *
 * Management: https://developers.facebook.com/apps/387037158089019/roles?
role=test%20users
 *
 * This class encapsulates user data retrieved from Facebook, providing
getters
 * to access the data.
 *
 * Note: This class assumes that the provided data is valid and does not
perform
 * any validation.
 *
 * @author jtremaux
 */
public class FacebookUser {
    private final String id;
    private final String email;
    private final String fullName;
    private final String accessToken;

    /**
```

```

    * Constructs a Facebook user with the specified data.
    *
    * @param id          The user's Facebook ID.
    * @param email       The user's email address.
    * @param fullName    The user's full name.
    * @param accessToken The access token associated with the user.
    */
    public FacebookUser(String id, String email, String fullName, String
accessToken) {
        this.id = id;
        this.email = email;
        this.fullName = fullName;
        this.accessToken = accessToken;
    }

    /**
     * Retrieves the Facebook user's ID.
     *
     * @return The Facebook user's ID.
     */
    public String getId() {
        return id;
    }

    /**
     * Retrieves the Facebook user's email address.
     *
     * @return The Facebook user's email address.
     */
    public String getEmail() {
        return email;
    }

    /**
     * Retrieves the Facebook user's full name.
     *
     * @return The Facebook user's full name.
     */
    public String getFullName() {
        return fullName;
    }

    /**
     * Retrieves the access token associated with the Facebook user.
     *
     * @return The access token associated with the Facebook user.
     */
    public String getAccessToken() {
        return accessToken;
    }
}

```

In this refactored version:

The fields are made private, ensuring encapsulation. Getter methods are provided to access the fields. The constructor initializes the fields but does not expose them directly. Documentation is improved to provide better clarity on class usage and assumptions.

Chat 2: After being told that it was an Encapsulation smell:

To address the encapsulation smell in the initial code, we'll make the fields private and provide public getter methods to access them. This ensures that the fields are accessed through methods rather than being directly exposed. Here's the modified code:

```
package com.sismics.books.rest.model;

/**
 * Facebook test user.
 *
 * Management : https://developers.facebook.com/apps/387037158089019/roles?
 * role=test%20users
 *
 * @author jtremeaux
 */
public class FacebookUser {
    private String id;
    private String email;
    private String fullName;
    private String accessToken;

    public FacebookUser(String id, String email, String fullName, String
accessToken) {
        this.id = id;
        this.email = email;
        this.accessToken = accessToken;
        this.fullName = fullName;
    }

    public String getId() {
        return id;
    }

    public String getEmail() {
        return email;
    }

    public String getFullName() {
        return fullName;
    }

    public String getAccessToken() {
        return accessToken;
    }
}
```

In this refactoring:

The fields `id`, `email`, `fullName`, and `accessToken` are changed to private access level. Getter methods are provided to allow access to these private fields from outside the class.

Here, we see that the LLM was able to figure out that the smell is actually an encapsulation smell and thus was able to refactor it also by making all the public variables final and also just providing the getters for these variables.

The results of the AI model in either case is exactly what is needed to fix the smell, which showcases that it was able to figure out the smell on its own, and then fix the same in the very first attempt itself.