MimeType - Missing Encapsulation

Chat 1: Before being told that it is a Missing Encapsulation Smell

Link to chat

Yes, there are a couple of design smells in this code:

1. **Duplication**: The MIME types are declared as string literals in multiple places, which can lead to duplication and potential errors if they need to be updated.
2. **Hardcoding**: The MIME types are hardcoded directly into the `MimeType` class, which makes it inflexible and harder to maintain if new MIME types need to be added or existing ones modified.

To refactor this code, you can use an enumeration to define the MIME types, which helps organize them logically and avoids duplication. Here's how you can refactor the code:

```
package com.sismics.books.core.util.mime;

/**
 * A collection of MIME types.
 */
public enum MimeType {
    IMAGE_X_ICON("image/x-icon"),
    IMAGE_PNG("image/png"),
    IMAGE_JPEG("image/jpeg"),
    IMAGE_GIF("image/gif"),
    APPLICATION_ZIP("application/zip"),
    APPLICATION_PDF("application/pdf");

    private final String value;

    MimeType(String value) {
        this.value = value;
    }

    public String getValue() {
        return value;
    }
}
```

With this refactoring, each MIME type is defined only once, in a more structured and maintainable way. You can access the MIME types using the enum constants, and if you need to retrieve the string value, you can use the `getValue()` method.

Chat 2: After being told that it is a Missing Encapsulation Smell

Link to Chat

Certainly! The Missing Encapsulation smell refers to the absence of proper encapsulation of data within classes. In this case, the `MimeType` class exposes its fields directly, which can lead to potential issues if other classes modify them directly. To fix this smell, we can encapsulate these fields by making them private and providing public getter methods. Here's the modified `MimeType` class:

```
package com.sismics.books.core.util.mime;

/**
 * A collection of MIME types.
 *
 * @author jtremeaux
 */
public class MimeType {

    private static final String IMAGE_X_ICON = "image/x-icon";

    private static final String IMAGE_PNG = "image/png";

    private static final String IMAGE_JPEG = "image/jpeg";

    private static final String IMAGE_GIF = "image/gif";

    private static final String APPLICATION_ZIP = "application/zip";

    private static final String APPLICATION_PDF = "application/pdf";

    public static String getImageXIcon() {
        return IMAGE_X_ICON;
    }

    public static String getImagePng() {
        return IMAGE_PNG;
    }

    public static String getImageJpeg() {
        return IMAGE_JPEG;
    }

    public static String getImageGif() {
        return IMAGE_GIF;
    }

    public static String getApplicationZip() {
        return APPLICATION_ZIP;
    }

    public static String getApplicationPdf() {
        return APPLICATION_PDF;
    }
}
```

By encapsulating the MIME types as private static constants and providing public getter methods, we ensure that other classes can access these values but cannot modify them directly, thus improving the encapsulation of the `MimeType` class.

> here, we see a refactoring done by the LLM that closely resembles what we did manually thus, the performance of the LLM was drastically different and improved when it was informed of which specific smell to fix

> The AI generated code looks almost correct and what should be done in this case. In the manual refactoring, we changed the variable types to `protected` so that they could be used in only sub-classes directly, which was an observation after reading the code manually. The AI model didn't have the context os usage of these variables, and thus did correct in that sense.