perplexity

シングルトンパターンでEmployeeInfoLoggerクラスを作りたい。

メインクラスからlogOutputメソッドを呼び出したい。
ロガーはアプリ内で1回だけ作りたい。

シングルトンパターンを使用して`EmployeeInfoLogger`クラスを作成し、メインクラスから`logOutput`メソッドを呼び出す方法を以下に示します。

```
public class EmployeeInfoLogger {
    private static EmployeeInfoLogger instance;
    private static final Logger LOGGER = Logger.getLogger("EmployeeInfoLogger");

    private EmployeeInfoLogger() {
        // ロガーの初期化処理
        try {
            FileHandler handler = new FileHandler("employee_info.log", true);
            LOGGER.addHandler(handler);
            SimpleFormatter formatter = new SimpleFormatter();
            handler.setFormatter(formatter);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static EmployeeInfoLogger getInstance() {
        if (instance == null) {
            instance = new EmployeeInfoLogger();
        }
        return instance;
    }

    public void logOutput(String message) {
        LOGGER.info(message);
    }
}
```

メインクラスでの使用例：

```
public class Main {
    public static void main(String[] args) {
        EmployeeInfoLogger logger = EmployeeInfoLogger.getInstance();
        logger.logOutput("アプリケーションが起動しました");
```

```
    }
}
```

この実装では:

1. `EmployeeInfoLogger`クラスは`private`コンストラクタを持ち、外部からのインスタンス化を防ぎます[1] [2]。
2. `getInstance()`メソッドを通じて唯一のインスタンスにアクセスします[1] [2]。
3. ロガーの初期化は`EmployeeInfoLogger`のコンストラクタ内で一度だけ行われます[3]。
4. メインクラスや他のクラスから`getInstance()`を使ってロガーにアクセスし、`logOutput()`メソッドを呼び出すことができます[2]。

この方法により、アプリケーション全体で一つのロガーインスタンスを共有し、一貫したログ管理が可能になります[3]。

⁂

1. https://academy.cyzennt.co.jp/blog/java-シングルトンパターンの使い方徹底解説-実装/
2. https://qiita.com/Dragoninfo/items/0bd05e0beedadb5a6c4a
3. https://style.potepan.com/articles/32867.html