Универсальный Логгер

Разработать систему, которая позволяет использовать несколько **несовместимых** сторонних библиотек логирования через **единый, стандартизированный интерфейс**.

Контекст

Ваше основное приложение разработано для использования логгеров с очень специфическим методом: log(message: str, level: str). Однако, в вашем проекте уже есть две старые, сторонние утилиты для вывода информации, которые имеют совершенно разные интерфейсы.

Название утилиты	**Исходный метод**	**Ожидаемый формат данных**
OldFileLogger (Адаптируемый класс 1)	log_to_file(message: str, level: str)	Принимает сообщение и уровень **раздельно** как строки.
ConsolePrinter (Адаптируемый класс 2)	print_message(data: dict)	Принимает **один словарь** (dict) с ключами 'level' и 'message'.

Задачи

Выполните следующие шаги для реализации паттерна Адаптер:

1. Создание Несовместимых Классов (Adaptees)

1. Определите класс OldFileLogger с единственным методом log_to_file(message, level), который просто выводит сообщение, имитирующее запись в файл.
2. Определите класс ConsolePrinter с единственным методом print_message(data), который форматирует и выводит словарь data в консоль.

2. Определение Целевого Интерфейса (Target)

1. Создайте абстрактный класс под названием **LoggerInterface**.

2. Внутри него определите абстрактный метод **log(message: str, level: str)**. Это наш целевой интерфейс, который будет использовать основное приложение.

3. Разработка Классов-Адаптеров (Adapters)

1. Создайте класс **FileLoggerAdapter**, который:
    - Реализует **LoggerInterface**.
    - Принимает и сохраняет экземпляр OldFileLogger в конструкторе.
    - Реализует метод log(), который вызывает внутренний несовместимый метод log_to_file() объекта OldFileLogger.
2. Создайте класс **ConsoleAdapter**, который:
    - Реализует **LoggerInterface**.
    - Принимает и сохраняет экземпляр ConsolePrinter в конструкторе.
    - Реализует метод log(), который **преобразует** входные аргументы (message, level) в требуемый словарь {'level': ..., 'message': ...} и затем вызывает внутренний несовместимый метод print_message().

4. Тестирование Приложения (Client)

1. Создайте функцию run_application_code(logger: LoggerInterface), которая принимает любой объект, соответствующий LoggerInterface, и вызывает его метод log() несколько раз с разными сообщениями и уровнями (INFO, WARNING, ERROR).
2. Создайте экземпляры несовместимых логгеров, оберните их в соответствующие адаптеры и передайте адаптеры в функцию run_application_code().