What is Record in java? Tell me your opinion on whether I should use Lombok ,Records and simple DTO with setter/setter. How you are doing it in your application.

This is an opinionated question designed to test if you understand modern Java vs. legacy patterns. Here is a strong, balanced answer.

1. What is a Record?

"A **Record** (introduced in Java 14, standard in Java 16) is a special class designed to be a transparent carrier for immutable data.
It automatically generates:

- A canonical constructor.
- Getters (named `name()` instead of `getName()`).
- `equals()`, `hashCode()`, and `toString()`.
- All fields are `private final` by default.

It is concise: `public record UserDTO(String name, int age) {}` replaces 50 lines of POJO code."

2. My Opinion: Record vs. Lombok vs. POJO

"My preference depends on the use case:"

- **For DTOs (Data Transfer Objects): I prefer Records.**
  - **Why:** They are immutable by design, which is perfect for passing data between layers (Controller → Service). They are standard Java (no external dependency like Lombok) and play well with modern frameworks (Spring Boot 3 supports them natively).
- **For JPA Entities: I prefer Lombok (`@Data` or `@Getter/@Setter`).**
  - **Why:** JPA Entities **cannot** be Records because JPA requires:
    1. No-args constructor (Records don't have one by default).
    2. Mutability (Setters) for Hibernate proxies to update fields.
    3. Class inheritance (Records cannot extend other classes).
  - So, for Entities, Lombok is still the king to reduce boilerplate.
- **For "Simple POJO with Setters": I avoid this.**

- **Why:** Writing getters/setters manually is noise. It clutters the codebase and makes code reviews harder. I only do this if the project strictly forbids external libraries like Lombok.

3. How I do it in my application

"In my current Spring Boot 3 application, we follow this standard:

1. **API Layer (Request/Response):** We use **Records**.
   - `public record CreateUserRequest(String email, String password) {}`
2. **Persistence Layer (Entities):** We use **Lombok**.
   - `@Entity @Getter @Setter @NoArgsConstructor`
3. **Mapping:** We use **MapStruct** to convert the Record (DTO) to the Lombok-annotated Entity."

Summary Table

Type	Use Case	Recommendation
Record	DTOs, Event Messages, Config Props	**YES (Java 17+)**
Lombok	JPA Entities, Mutable Domain Objects	**YES**
Manual POJO	Legacy code, No-library policy	**Avoid**

⁂

1. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/429036f4-3bc3-4a7a-8c6f-90246e73b9b9/image.jpg
2. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/24f18c0a-b56e-4862-a619-e2959036a5c2/image.jpg
3. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/97e9ed86-ed11-4035-a18b-1eb05c1bab4e/image.jpg
4. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/87f62423-96f2-4071-9802-8f6699e0ecd8/image.jpg