**Records - Java 16 - THIS IS MOST AMAZING**

Before Java 14: The Classic Approach

In the days before Java 14, if you wanted to create a simple class to represent, say, a **Person**, you'd end up with something like this:

1. public class Person {
2. private final String name;
3. private final int age;
5. public Person(String name, int age) {
6. this.name = name;
7. this.age = age;
8. }
10. public String getName() {
11. return name;
12. }
14. public int getAge() {
15. return age;
16. }
18. @Override
19. public boolean equals(Object o) {
20. if (this == o) return true;
21. if (!(o instanceof Person)) return false;
22. Person person = (Person) o;
23. return age == person.age &&
24. Objects.equals(name, person.name);
25. }
27. @Override
28. public int hashCode() {
29. return Objects.hash(name, age);
30. }
32. @Override
33. public String toString() {
34. return "Person{" +
35. "name='" + name + '\'' +
36. ", age=" + age +
37. '}';
38. }
39. }

After Java 14: The Record Class Way

Now, let's see the magic of Java 14 with record classes. All that code up there can be simplified to just:

1. public record Person(String name, int age) {
2. }

And that's it! Java takes care of the rest, like generating getters, **equals**, **hashCode**, and **toString** methods. This new syntax is super sleek and saves you a ton of time and lines of code. It's like going from manually crafting an entire meal to just picking your favorite dish off a menu!