Record in Java : -

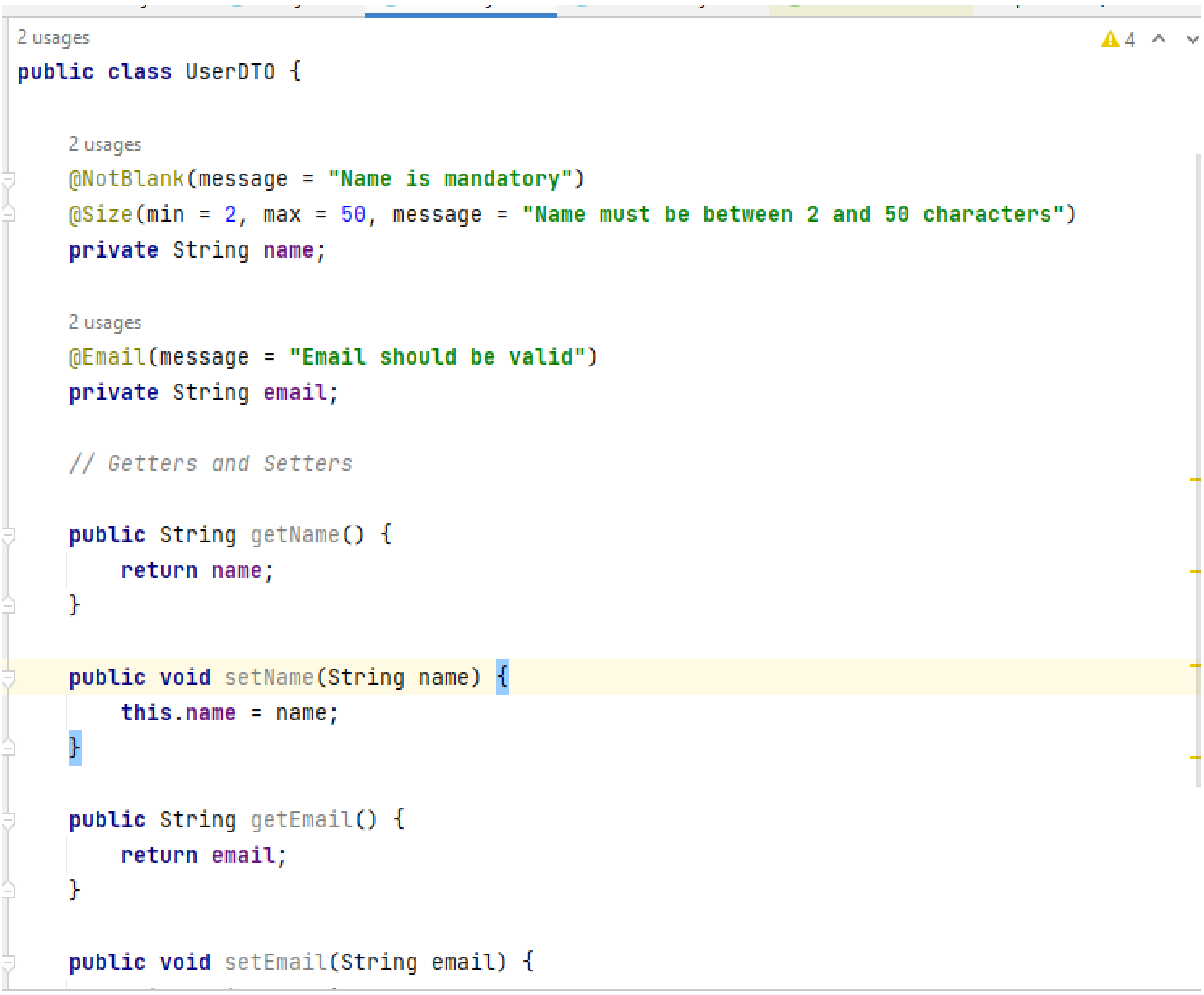
What is record?

A **record** in Java is a special type of class that makes it super easy to create objects that only hold data. It automatically generates all the common code you need, like:

- A constructor to initialize the data
- Methods to get the values (getters)
- A nice toString() method
- equals() and hashCode() methods

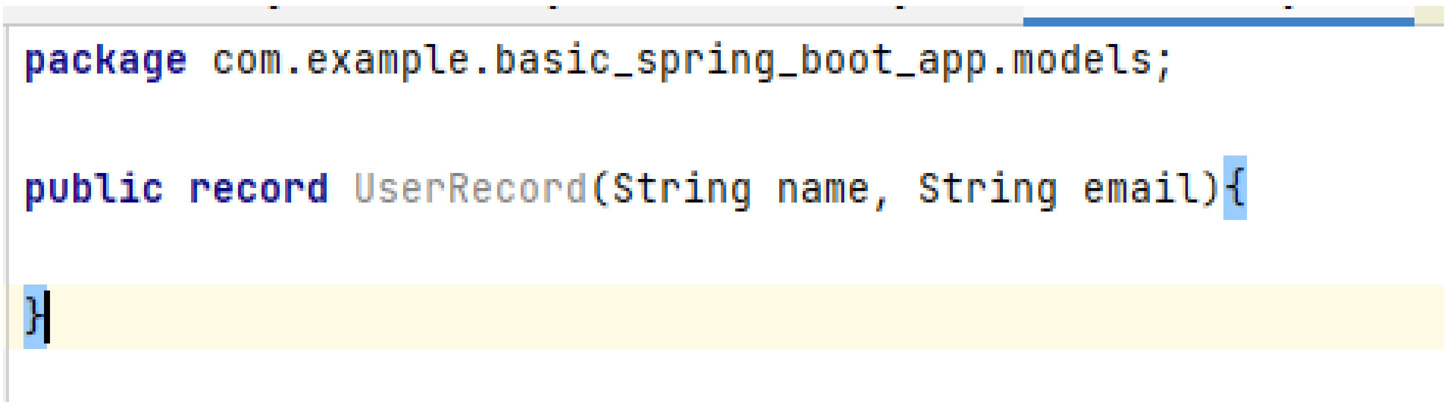
How to create Record class in Java ?

POJO



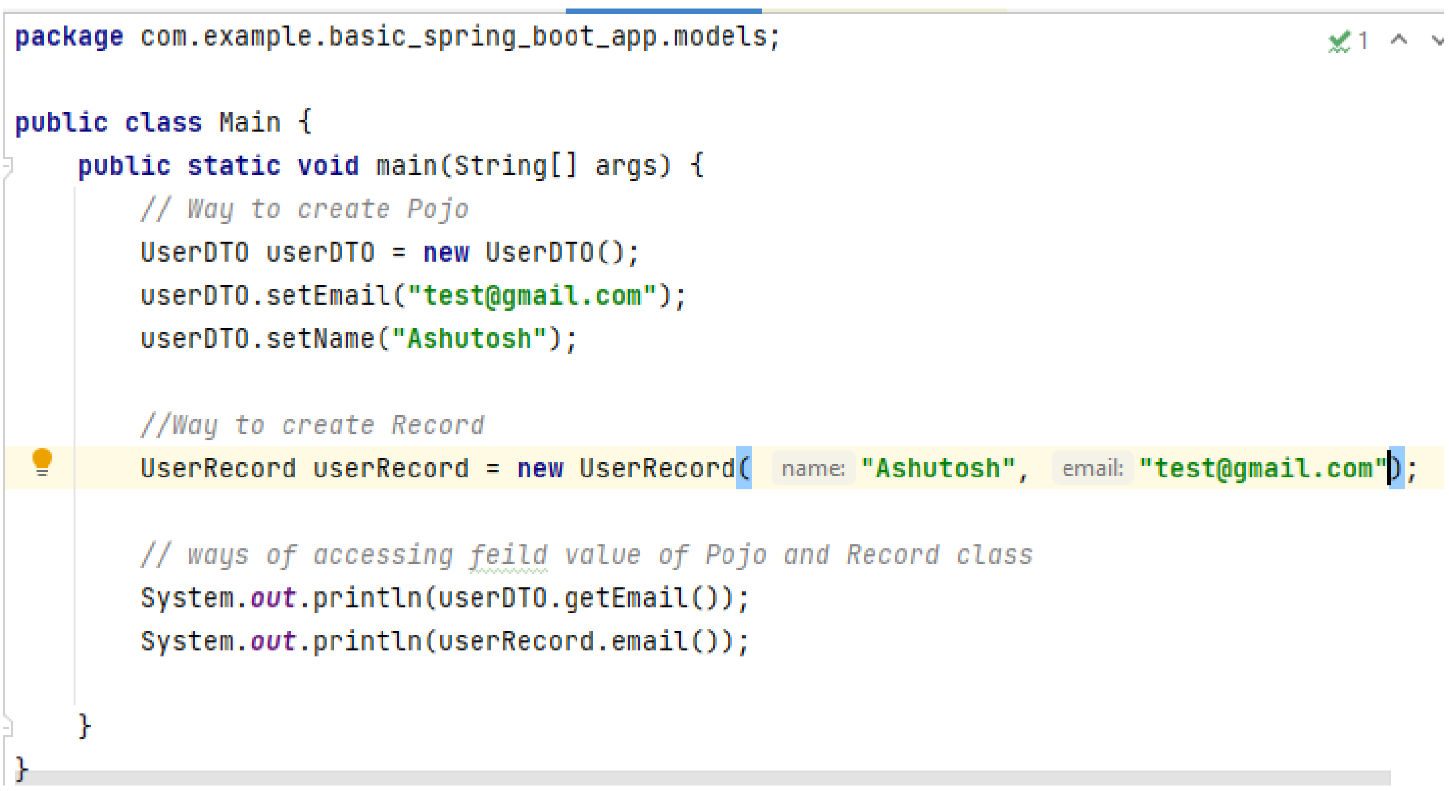
```
2 usages
public class UserDTO {

    2 usages
    @NotBlank(message = "Name is mandatory")
    @Size(min = 2, max = 50, message = "Name must be between 2 and 50 characters")
    private String name;

    2 usages
    @Email(message = "Email should be valid")
    private String email;

    // Getters and Setters

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
```

record



```
package com.example.basic_spring_boot_app.models;

public record UserRecord(String name, String email){

}
```

How to Use record instead of POJO



```java
package com.example.basic_spring_boot_app.models;

public class Main {
    public static void main(String[] args) {
        // Way to create Pojo
        UserDTO userDTO = new UserDTO();
        userDTO.setEmail("test@gmail.com");
        userDTO.setName("Ashutosh");

        //Way to create Record
        UserRecord userRecord = new UserRecord( name: "Ashutosh",  email: "test@gmail.com");

        // ways of accessing feild value of Pojo and Record class
        System.out.println(userDTO.getEmail());
        System.out.println(userRecord.email());

    }
}
```

Benefits:

1. **Less Code**: Records remove the need to write boilerplate code.
2. **Immutable by Default**: The data inside a record can't be changed after it's created.
3. **Automatic Methods**: You get methods like toString(), equals(), and hashCode() without doing anything extra.