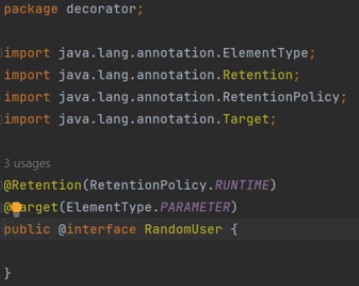
**Автоматическое заполнение тестов сотнями разных параметров.**

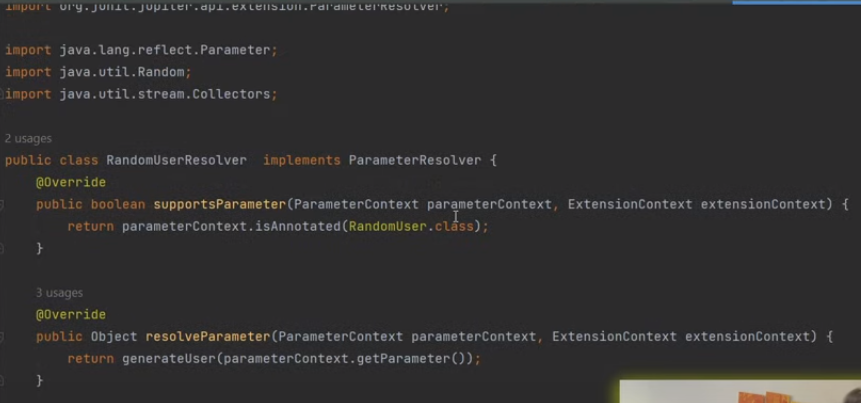
**I**. Создаем собственную аннотацию:



**II**. Реализуем класс, в котором:

- вызываем нашу аннотацию,

- прописываем действия, выполняемые при вызове нашей аннотации.





, где:

- **ParametrResolver** - определяет API для расширений, которые хотят динамически разрешать аргументы параметров во время выполнения. Если конструктор тестового класса или метод @Test, @BeforeEach, @AfterEach, @BeforeAll или @AfterAll объявляет параметр, аргумент для этого параметра должен быть разрешен во время выполнения с помощью ParameterResolver. Реализации должны предоставлять конструктор без аргументов.

- **Parametr** – класс с информацией о параметрах метода. Параметр предоставляет информацию о параметрах метода, включая его имя и модификаторы. Он также предоставляет альтернативные способы получения атрибутов параметра.

1. Имплементируем интерфейс **ParametrResolver.**

2. Переопределяем два метода интерфейса **ParametrResolver**, где:

- в методе ***supportsParameter*** мы привязываем к нашему классу созданную нами ранее аннотацию.

- в методе ***resolveParameter*** мы возвращаем 3-й метод с прописанной в нем логикой поведения

3. Заполняем логикой работы 3-й метод:

- проверяем введенный параметр на соответствие необходимому нам классу (тут “UserModel”)

- если проверка пройдена, то заполняем с помощью класса “Random” входящие параметры, каждый раз новыми данными.

**III**. В тесте, в параметрах метода, вначале прописав нашу аннотацию “@RandomUser”, мы запускаем механизм автоматического заполнения полей принимаемого на вход объекта класса “UserModel”:

