The `@CsvSource` annotation in JUnit 5 is used to provide multiple sets of arguments to a `@ParameterizedTest` using CSV (Comma-Separated Values) format.

✅ Key points:

- Each row represents a set of arguments.
- Strings with commas **must be quoted** using double quotes.
- Data types are automatically converted based on the method parameters.

📘 Basic Example

```
@ParameterizedTest
@CsvSource({
    "apple, 5",
    "banana, 7",
    "cherry, 12"
})
void testFruits(String fruit, int quantity) {
    assertNotNull(fruit);
    assertTrue(quantity > 0);
}
```

🔎 Output / Test Iterations:

fruit	quantity
apple	5
banana	7
cherry	12

📘 Example with String containing a comma

```
@ParameterizedTest
@CsvSource({
    "\"red, apple\", 10",
    "\"green, banana\", 15",
    "\"big, cherry\", 20"
})
void testFruitsWithComma(String fruit, int quantity) {
    assertNotNull(fruit);
    assertTrue(quantity > 0);
}
```

🔎 **Output / Test Iterations:**

fruit	quantity
red, apple	10
green, banana	15
big, cherry	20

> Note: Quotes are **required** when a string value includes a comma so that it's not misinterpreted as separate columns.

📘 Example with more than two parameters

```
@ParameterizedTest
@CsvSource({
    "apple, 5, true",
    "\"mango, sweet\", 10, false",
    "grape, 20, true"
})
void testMultipleValues(String fruit, int quantity, boolean isFresh) {
    assertNotNull(fruit);
    assertTrue(quantity > 0);
}
```

🔎 **Output / Test Iterations:**

fruit	quantity	isFresh
apple	5	true
mango, sweet	10	false
grape	20	true

Would you like a visual table or diagram to illustrate how quotes affect parsing?