The `@BeforeEach` annotation is part of the JUnit 5 lifecycle, and it ensures that a fresh instance of the object is created before each test is executed.

Here's a step-by-step guide in IntelliJ, including the code and how to set up everything:

Step-by-Step Guide for Adding `@BeforeEach` in IntelliJ

1. Set Up JUnit 5 in IntelliJ

Before creating the test class, make sure you have JUnit 5 set up in your project. If you're using Maven or Gradle, add the dependencies as follows:

For Maven (`pom.xml`):

```
<dependencies>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.8.1</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.8.1</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

For Gradle (`build.gradle`):

```
dependencies {
    testImplementation 'org.junit.jupiter:junit-jupiter-api:5.8.1'
    testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.8.1'
}
```

2. Create the `Calculator` Class

Create a simple `Calculator` class with an `add` method:

```
public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
}
```

3. Create the `CalculatorTest` Class with `@BeforeEach`

1. **Step 1**: Right-click on the `src/test/java` directory in IntelliJ.
2. **Step 2**: Select **New > Java Class**.
3. **Step 3**: Name the class `CalculatorTest`.

In the `CalculatorTest` class, you will import `@BeforeEach` from `org.junit.jupiter.api.BeforeEach` and create a setup method to initialize the `Calculator` object before each test.

Here is the code:

```
import org.junit.jupiter.api.BeforeEach;  // Importing @BeforeEach
annotation
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class CalculatorTest {

    // Declare the calculator object that will be shared across all tests
    private Calculator calculator;

    // Step 1: Create the @BeforeEach method
    @BeforeEach
    public void setup() {
        // This method is called before each test
```

```
        calculator = new Calculator(); // Initialize the Calculator object
    }

    // Step 2: Write the test method for addition
    @Test
    public void testAddition() {
        int result = calculator.add(2, 3);  // Perform addition using the
calculator
        assertEquals(5, result, "The addition result should be 5");  //
Assert the result
    }
}
```

4. What the Code Does

1. **`@BeforeEach setup()`**:
   - This method is annotated with `@BeforeEach`, meaning it will run **before each test**. In this case, it initializes the `Calculator` object so that each test method gets a fresh instance of `Calculator`.
2. **Test Method `testAddition()`**:
   - The `testAddition()` method tests the `add()` method of the `Calculator` class. We call `calculator.add(2, 3)` and then assert that the result is `5`.

5. Run the Tests in IntelliJ

1. **Step 1**: To run the tests, simply click the **Run** icon (green triangle) next to the test method `testAddition()` or next to the `CalculatorTest` class name.
   - You can also right-click the test class or method and select **Run 'CalculatorTest'** or **Run 'testAddition()'**.
2. **Step 2**: IntelliJ will compile the project and run the JUnit tests. You should see the results in the **Run** window at the bottom.
   - If everything is correct, the test will pass, and you should see an output like:

```
PASSED: testAddition
```

6. Expected Output

If the `add` method of `Calculator` works correctly, the output in IntelliJ will show:

```
Test passed: testAddition()
```

How `@BeforeEach` Works

The `@BeforeEach` annotation allows you to run setup code before each test method. Here's what happens:

1. **Before each test method**: The `setup()` method is called, initializing the `Calculator` instance.
2. **Each test method**: The test methods, such as `testAddition()`, will be run with the `calculator` object already set up.
3. **Fresh state for each test**: Since `setup()` runs before every test, it ensures that the `calculator` is always in a fresh state for each test method, avoiding test interference.

Algorithm for Testing with `@BeforeEach`

1. **Initialize the Object**: Use `@BeforeEach` to initialize the object before each test.
2. **Write Test Methods**: Write test methods to verify functionality (e.g., `testAddition()`).
3. **Run Tests**: Execute tests and verify that the results match expectations.
4. **Isolate Test Logic**: Since `@BeforeEach` ensures a fresh state, each test will not affect others.

Summary

- **`@BeforeEach`** is used to set up necessary objects or conditions before each test.
- It ensures that tests don't affect each other by creating a fresh state for each test.
- You can run the tests easily in IntelliJ by right-clicking the test class or method and choosing **Run**.