Unit Testing a Class

Consider the following class:

```
public class Point {
    private double x;
    private double y;
    public Point(double x, double y) {
        super();
        this.x = x;
        this.y = y;
    }
    public double getX() {
        return x;
    }
    public void setX(double x) {
        this.x = x;
    }
    public double getY() {
        return y;
    }
    public void setY(double y) {
        this.y = y;
    }
    public Point translater(double dx, double dy) {
        return new Point(this.getX() + dx, this.getY() + dy);
    }
}
```

1. Add the test method `testTranslater0_0()`.

```
    @Test
    public final void testTranslater0_0() {
        Point a = new Point(1, 2);
        Point expected = new Point(1, 2);
        Point obtained = a.translater(0, 0);
        assertEquals(expected, obtained);
    }
```

2. Explain why this test does not pass.
3. Add the `equals()` method to the `Point` class for testing the equality of two points.
4. Write a test method for this new `equals()` method.
5. Add the test method `testTranslater1_3()`.

```
    @Test
    public final void testTranslater1_3() {
        Point a = new Point(1, 2);
        Point expected = new Point(2, 5);
        Point obtained = a.translater(1, 3);
        assertEquals(expected, obtained);
    }
```

6. Explain why the test does not pass and modify the code accordingly.
7. Verify that all tests pass.
8. Both methods `testTranslater1_3()` and `testTranslater0_0()` have common initialization. Write an external method in the test class that encapsulates this initialization. Ensure that the initialization is performed before each test method is executed.
9. Similarly, if after each execution of test methods, you want to execute common code (e.g., deallocation), how should you proceed?
10. Answer question 8 using the `@BeforeAll` annotation.