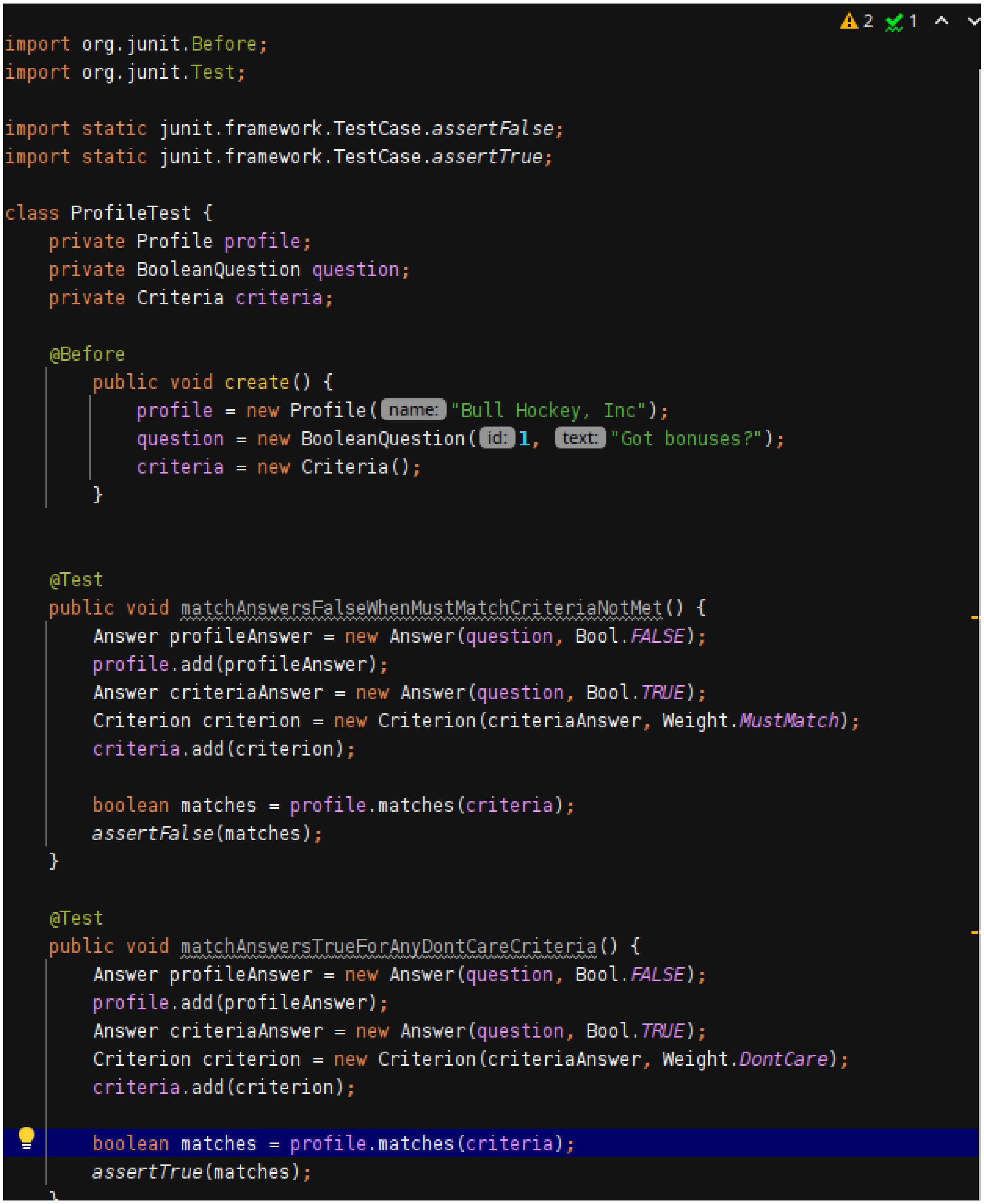
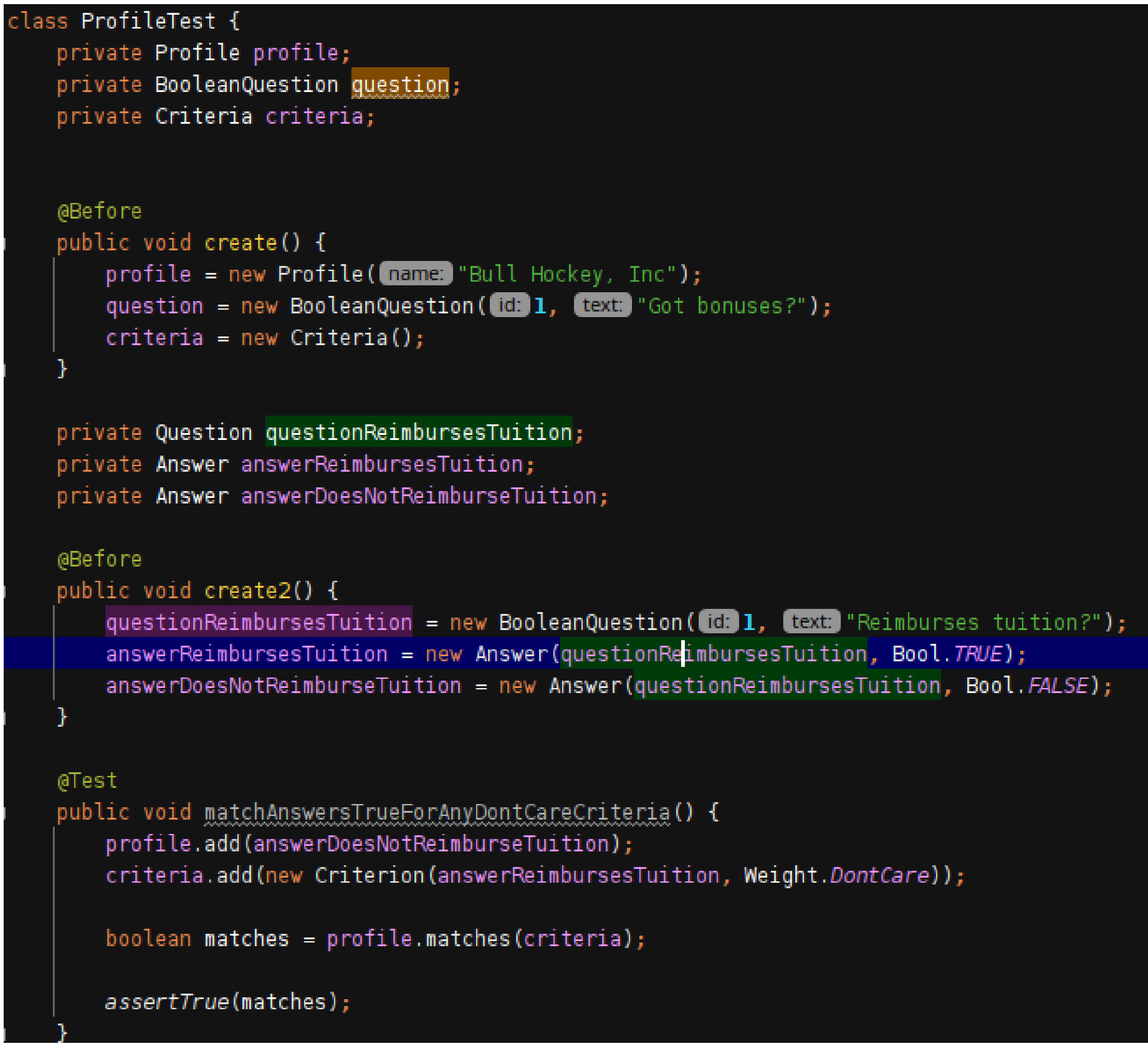
1. 

```java
import org.junit.Before;
import org.junit.Test;

import static junit.framework.TestCase.assertFalse;
import static junit.framework.TestCase.assertTrue;

class ProfileTest {
    private Profile profile;
    private BooleanQuestion question;
    private Criteria criteria;

    @Before
        public void create() {
            profile = new Profile( name: "Bull Hockey, Inc");
            question = new BooleanQuestion( id: 1,  text: "Got bonuses?");
            criteria = new Criteria();
        }


    @Test
    public void matchAnswersFalseWhenMustMatchCriteriaNotMet() {
        Answer profileAnswer = new Answer(question, Bool.FALSE);
        profile.add(profileAnswer);
        Answer criteriaAnswer = new Answer(question, Bool.TRUE);
        Criterion criterion = new Criterion(criteriaAnswer, Weight.MustMatch);
        criteria.add(criterion);

        boolean matches = profile.matches(criteria);
        assertFalse(matches);
    }

    @Test
    public void matchAnswersTrueForAnyDontCareCriteria() {
        Answer profileAnswer = new Answer(question, Bool.FALSE);
        profile.add(profileAnswer);
        Answer criteriaAnswer = new Answer(question, Bool.TRUE);
        Criterion criterion = new Criterion(criteriaAnswer, Weight.DontCare);
        criteria.add(criterion);

        boolean matches = profile.matches(criteria);
        assertTrue(matches);
```

2. If the JUnit chooses to run matchAnswersTrueForAnyDontCareCriteria() first, what is the sequence of events?
    a. Well before it even runs it creates a new profile, boolean question, and criteria

b. Then, it adds that profile answer to profile and does that for criteria to see if it is true and then matches the criteria answer to the weight
c. If done correctly, the criterion is added to the criterion
d. After everything is finished, it then checks if the boolean matches the profile and makes it true.
e. Returns true

3. In order to minimize the impact any one test has on another (avoiding static fields in test cases as well), create a more condensed but more readable arrange portion of each test by inlining some local variables.

a. 

```
class ProfileTest {
    private Profile profile;
    private BooleanQuestion question;
    private Criteria criteria;


    @Before
    public void create() {
        profile = new Profile( name: "Bull Hockey, Inc");
        question = new BooleanQuestion( id: 1,  text: "Got bonuses?");
        criteria = new Criteria();
    }

    private Question questionReimbursesTuition;
    private Answer answerReimbursesTuition;
    private Answer answerDoesNotReimburseTuition;

    @Before
    public void create2() {
        questionReimbursesTuition = new BooleanQuestion( id: 1,  text: "Reimburses tuition?");
        answerReimbursesTuition = new Answer(questionReimbursesTuition, Bool.TRUE);
        answerDoesNotReimburseTuition = new Answer(questionReimbursesTuition, Bool.FALSE);
    }

    @Test
    public void matchAnswersTrueForAnyDontCareCriteria() {
        profile.add(answerDoesNotReimburseTuition);
        criteria.add(new Criterion(answerReimbursesTuition, Weight.DontCare));

        boolean matches = profile.matches(criteria);

        assertTrue(matches);
    }
```