8

poniedziałek, 14 kwietnia 2025 12:18

Implementacja testów dla klasy efs.task.unittests.FitCalculator w klasie testowej efs.task.unittests.FitCalculatorTest.

- testy dla metody FitCalculator.isBMICorrect:
  - analogicznie do istniejącego testu shouldReturnTrue_whenDietRecommended utwórz test sprawdzający przypadek, dla którego isBMICorrect zwraca false (np. wzrost 1.72, waga 69.5).
  - utwórz test dla przypadku: wzrost użytkownika równy 0.0, waga dowolna -> oczekiwanie rezultat: metoda rzuca wyjątkiem IllegalArgumentException.
  - Korzystając z adnotacji **@ParameterizedTest** oraz **@ValueSource** utwórz test dla przypadku: wybrany wzrost, waga jako parametr- minimum 3 różne wartości -> oczekiwany rezultat dla wszystkich wag metoda zwraca true.
  - Korzystając z adnotacji **@ParameterizedTest** oraz **@CsvSource** utwórz test dla przypadków: wzrost i waga jako para parametrów, minimum 3 różne wartości -> oczekiwany rezultat dla wszystkich par wartości metoda zwraca false.
  - Korzystając z adnotacji **@ParameterizedTest** oraz **@CsvFileSource** utwórz test dla przypadków: wzrost i waga jako para parametrów pobierane z pliku resources.data.csv-> oczekiwany rezultat dla wszystkich par wartości metoda zwraca false.
  - Dla poprawienia czytelności test result korzystając z argumentu name przyjmowanego przez**@ParameterizedTest** nadaj nazwy parametrom przyjmowanym przez testy.

Po kolei

```
@Test
void shouldReturnTrue_whenDietRecommended() {
    //given
    double weight = 89.2;
    double height = 1.72;

    //when
    boolean recommended = FitCalculator.isBMICorrect(weight, height);

    //then
    assertTrue(recommended);
}
```

Zwraca true

```
@Test
void shouldReturException_when0() {
    double weight = 69.5;
    double height = 0;

    assertThrows(IllegalArgumentException.class, () -> FitCalculator.isBMICorrect(weight, height));
}
```

Wyrzuca exxception

```
@ParameterizedTest(name = "Should return true for weight = {0} and height = 1.80")
@ValueSource(doubles = {82.0, 86.5, 95.0})
    // minimum 3 wartości
void shouldReturnTrue_forVariousWeights(double weight) {
    // given
    double height = 1.80;

    // when
    boolean result = FitCalculator.isBMICorrect(weight, height);

    // then
    assertTrue(result);
}
```

```
@ParameterizedTest(name = "Should return true for weight = {0} and height = 1.80")
@ValueSource(doubles = {82.0, 86.5, 95.0})
```
Zamiast {0} wpisuje wartosci po kolei które podalismy w valuesoirce, i dla tyc wartosci odpala nam testy

```
/*    Korzystając z adnotacji @ParameterizedTest oraz @CsvSource utwórz test dla przypadków: wzrost i waga jako para parametrów,
    minimum 3 różne wartości -> oczekiwany rezultat dla wszystkich par wartości metoda zwraca false.*/

    @ParameterizedTest(name = "Should return false for weight = {0}, height = {1}")
    @CsvSource({
            "70, 1.72",
            "40.0, 1.50",
            "45.0, 1.80"
    })
    void shouldReturnFalse_forManyCases(double weight, double height) {
        // given – wartości przekazane w CsvSource

        // when
        boolean result = FitCalculator.isBMICorrect(weight, height);

        // then
        assertFalse(result);
    }
```

Teraz odpalamy testy dla odpowidnic wartosci i podajemy je zamiast 0 i 1 przy wypisaniu dla podniesiania czytelnsoci

```
/*   Korzystając z adnotacji @ParameterizedTest oraz @CsvFileSource utwórz test dla przypadków: wzrost i waga jako para parametrów pobierane z
    pliku resources.data.csv-> oczekiwany rezultat dla wszystkich par wartości metoda zwraca false.
}*/


    @ParameterizedTest(name = "From CSV: weight = {0}, height = {1} => should return false")
    @CsvFileSource(resources = "/data.csv", numLinesToSkip = 1)
    void shouldReturnFalse_forAllValuesFromCSV(double weight, double height) {
        // given – dane z CSV

        // when
        boolean result = FitCalculator.isBMICorrect(weight, height);

        // then
        assertFalse(result);
    }
```
Teraz bierzemy wartosci z pliku zamiast je recznie wpisywac
Skipujemy 1 linie bo tam mamy podpisane kolumny

```
/*    utwórz test dla przypadku: dla listy efs.task.unittests.TestConstants.TEST_USERS_LIST ->
    oczekiwania: użytkownik z najgorszym wynikiem BMI waga: 97.3, wzrost 1.79;*/

    @Test
    void shouldReturnWorstBMI() {
        //given
        List<User> usersList = TestConstants.TEST_USERS_LIST;


        //when
        User worst = FitCalculator.findUserWithTheWorstBMI(usersList);
        //then
        assertEquals(97.3, worst.getWeight(), 0.01);
        assertEquals(1.79, worst.getHeight(), 0.01);
    }
```
Mamy liste już stworzona w testconstanss i powornujemy czy teo sameo znalezlimy

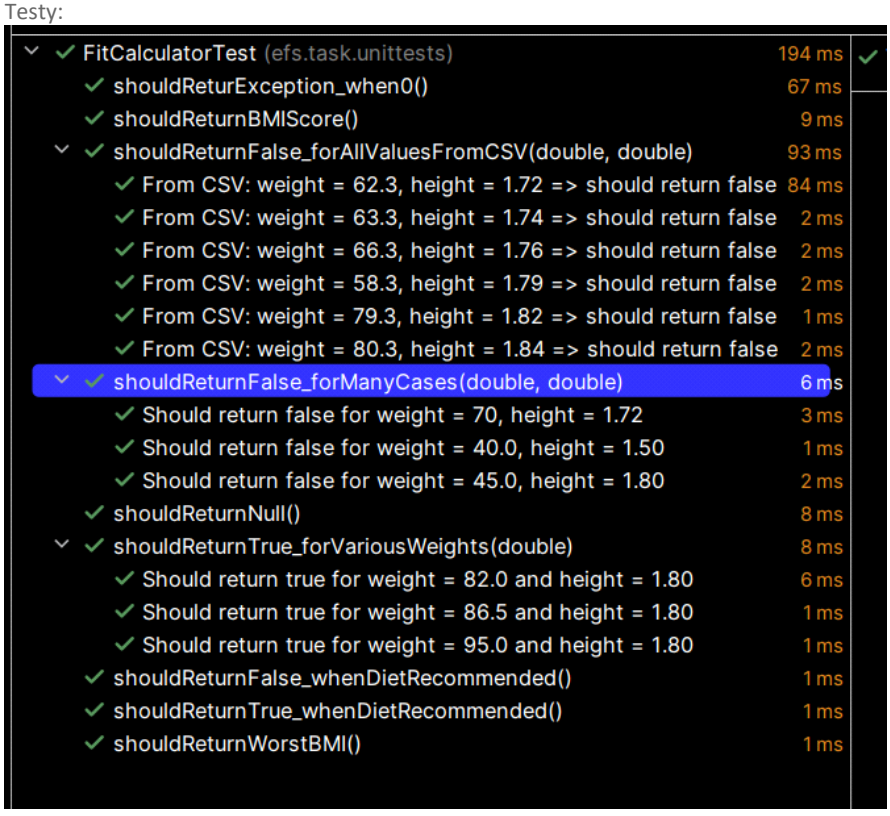
```
/*
    utwórz test dla przypadku: pusta lista użytkowników -> oczekiwania: metoda zwraca null;
```

```
*/
@Test
void shouldReturnNull() {
    List<User> usersList = new ArrayList<>();
    User worst = FitCalculator.findUserWithTheWorstBMI(usersList);
    //then
    assertEquals(null, worst);
}
```

Robimy pusta liste i sprawdzmay co się dzieje

```
/* test dla metody FitCalculator.calculateBMIScore:

 utwórz test dla przypadku: dla listy efs.task.unittests.TestConstants.TEST_USERS_LIST
  -> oczekiwania: efs.task.unittests.TestConstants.TEST_USERS_BMI_SCORE;*/
@Test
void shouldReturnBMIScore() {
    //given
    List<User> usersList = TestConstants.TEST_USERS_LIST;
    double[] expectedScores = TestConstants.TEST_USERS_BMI_SCORE;


    //when
    double[] bmiScores = new double[usersList.size()];
    bmiScores=FitCalculator.calculateBMIScore(usersList);
    //then
    assertArrayEquals(expectedScores, bmiScores, 0.01); // tolerancja 0.01 dla double

}
```

Mamy już liste oraz expectedscores liczymy bmi scores i porownujemy z tolerancja 0.01
Testy:



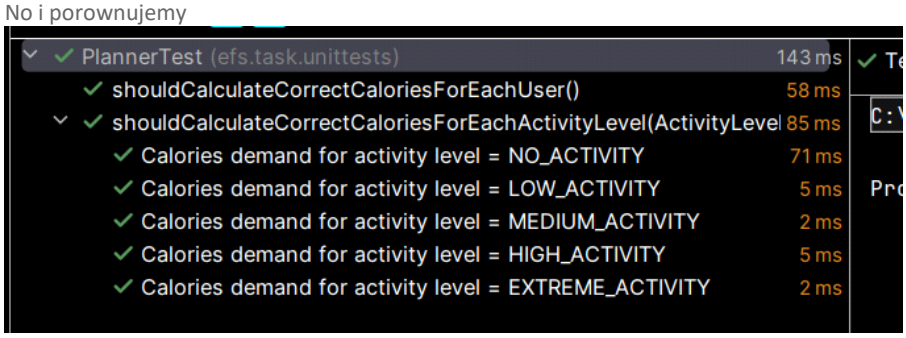
Stwórz klasę testową dla efs.task.unittests.Planner. Klasa testowa powinna zawierać pole typu Planner inicjalizowane jako nowa instancja obiektu planer przed każdym testem jednostkowym.

- testy dla metody Planner.calculateDailyCaloriesDemand:

- utwórz test parametryzowany sprawdzający poprawność wyliczenia dziennego zapotrzebowania kalorii dla wszystkich wartości typu wyliczeniowego efs.task.unittests.ActivityLevel, obliczenie dla użytkownika efs.task.unittests.TestConstants.TEST_USER -> oczekiwania: prawidłowe wartości zapotrzebowania kalorii dla użytkownika efs.task.unittests.TestConstants.TEST_USER znajdziesz w mapie efs.task.unittests.TestConstants.CALORIES_ON_ACTIVITY_LEVEL;

```java
public class PlannerTest {

    private Planner planner;  3 usages

    @BeforeEach
    void setUp() {
        planner = new Planner();
    }

    @ParameterizedTest(name = "Calories demand for activity level = {0}")
    @EnumSource(ActivityLevel.class)
    void shouldCalculateCorrectCaloriesForEachActivityLevel(ActivityLevel activityLevel) {
        // given
        User user = TestConstants.TEST_USER;
        int expectedCalories = TestConstants.CALORIES_ON_ACTIVITY_LEVEL.get(activityLevel);

        // when
        int actualCalories = planner.calculateDailyCaloriesDemand(user, activityLevel);

        // then
        assertEquals(expectedCalories, actualCalories);
    }
```

Najpierw przed kazdym testem tworrzymy sobie nowa isntancje planera
Znowu dajemy sobie parameterized dla wszsytkic activitylevel, pobieramy je z activitylevel
No i porownujemy




```
/*   testy dla metody Planner.calculateDailyIntake:

    utwórz test sprawdzający poprawność wyliczenia dziennego zapotrzebowania na składniki odżywcze (dailyIntake)
    dla użytkownika efs.task.unittests.TestConstants.TEST_USER -> oczekiwania: prawidłowe wartości zapotrzebowania
    na składniki odżywcze dla efs.task.unittests.TestConstants.TEST_USER takie jak w
        efs.task.unittests.TestConstants.TEST_USER_DAILY_INTAKE;*/
@Test
void shouldCalculateCorrectCaloriesForEachUser() {
    User user = TestConstants.TEST_USER;

    DailyIntake dailyIntake = planner.calculateDailyIntake(user);

    DailyIntake expected = TestConstants.TEST_USER_DAILY_INTAKE;

    assertEquals(expected.getProtein(), dailyIntake.getProtein(), 0.01);
    assertEquals(expected.getFat(), dailyIntake.getFat(), 0.01);
    assertEquals(expected.getCarbohydrate(), dailyIntake.getCarbohydrate(), 0.01);
    }
```

Mamy soobie jakieos usera i sprawdzamy czy dobrze nam policzylo z 0.01 odcylem