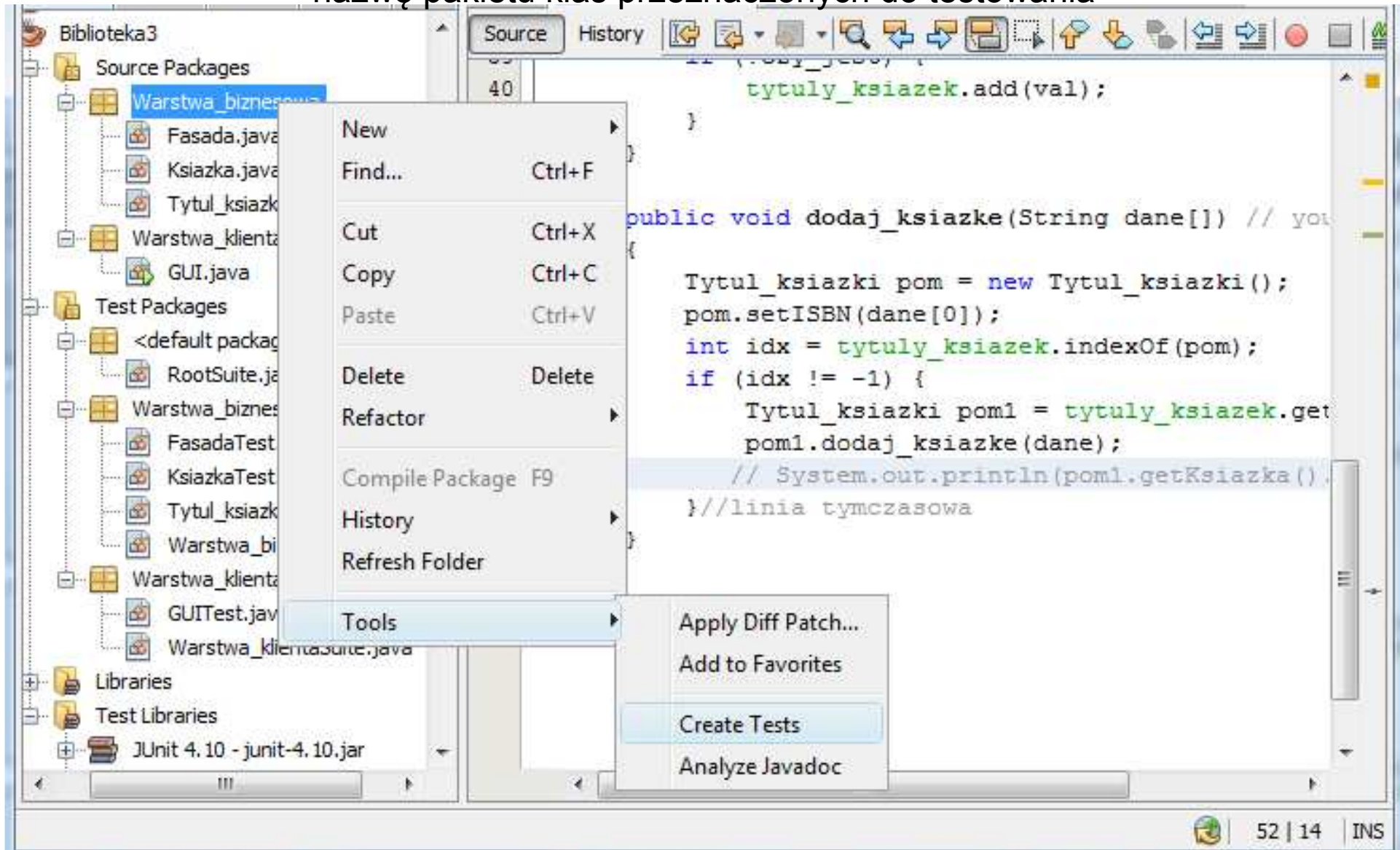


**Testy jednostkowe -
zastosowanie
oprogramowania JUNIT 4.0**
<http://www.junit.org/>

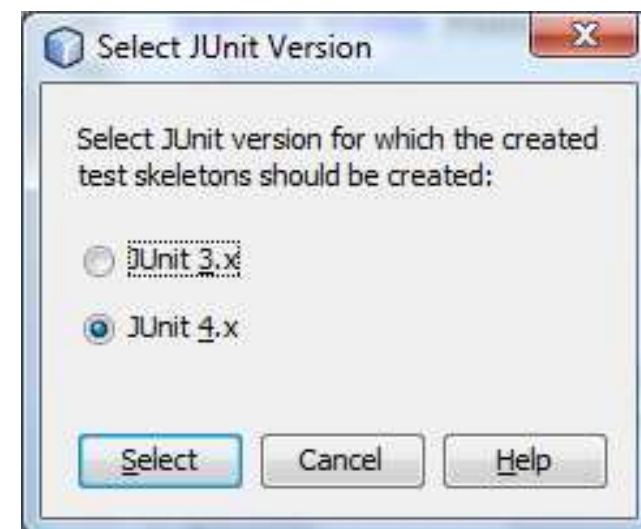
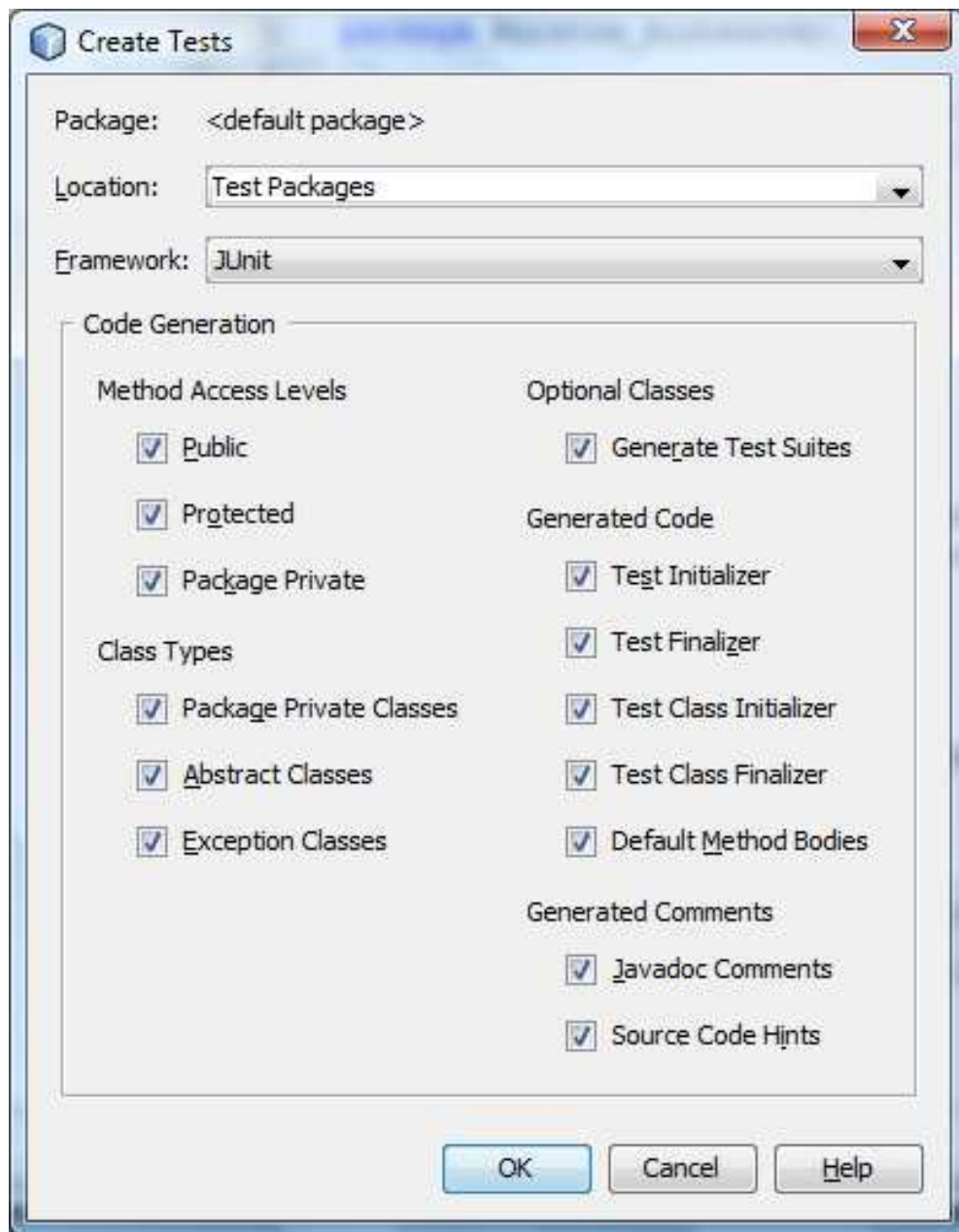
Zofia Kruczkiewicz

1. Aby utworzyć **test dla jednej klasy**, należy kliknąć prawym przyciskiem myszy w oknie *Projects* na wybraną klasę i wybrać *Tools > Create Tests (Ctrl-Shift-U)*, aby otworzyć okno dialogowe *Create Tests* .

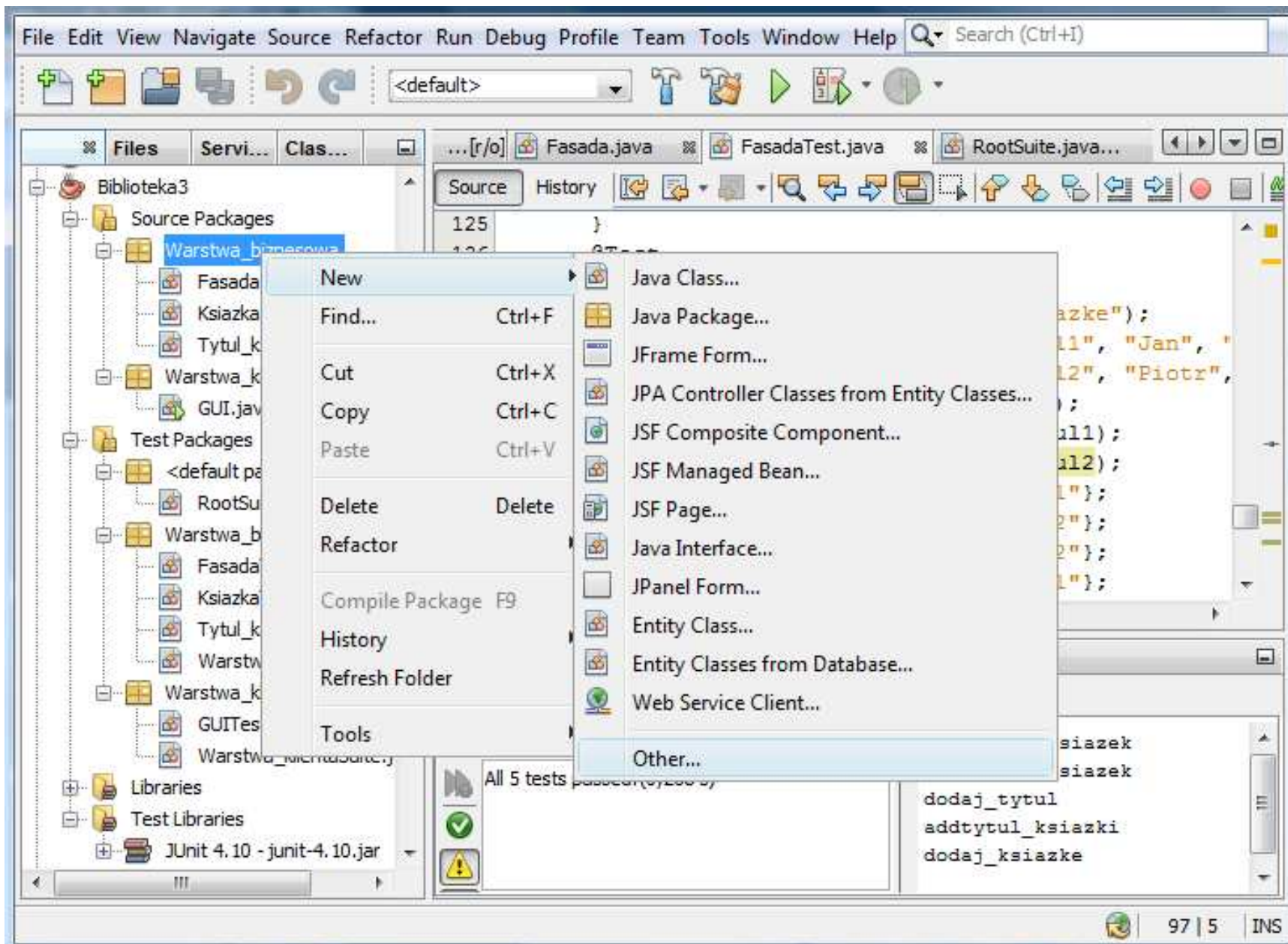
Utworzenie **zestawu testów** wykonuje się podobnie klikając prawym klawiszem na nazwę pakietu klas przeznaczonych do testowania



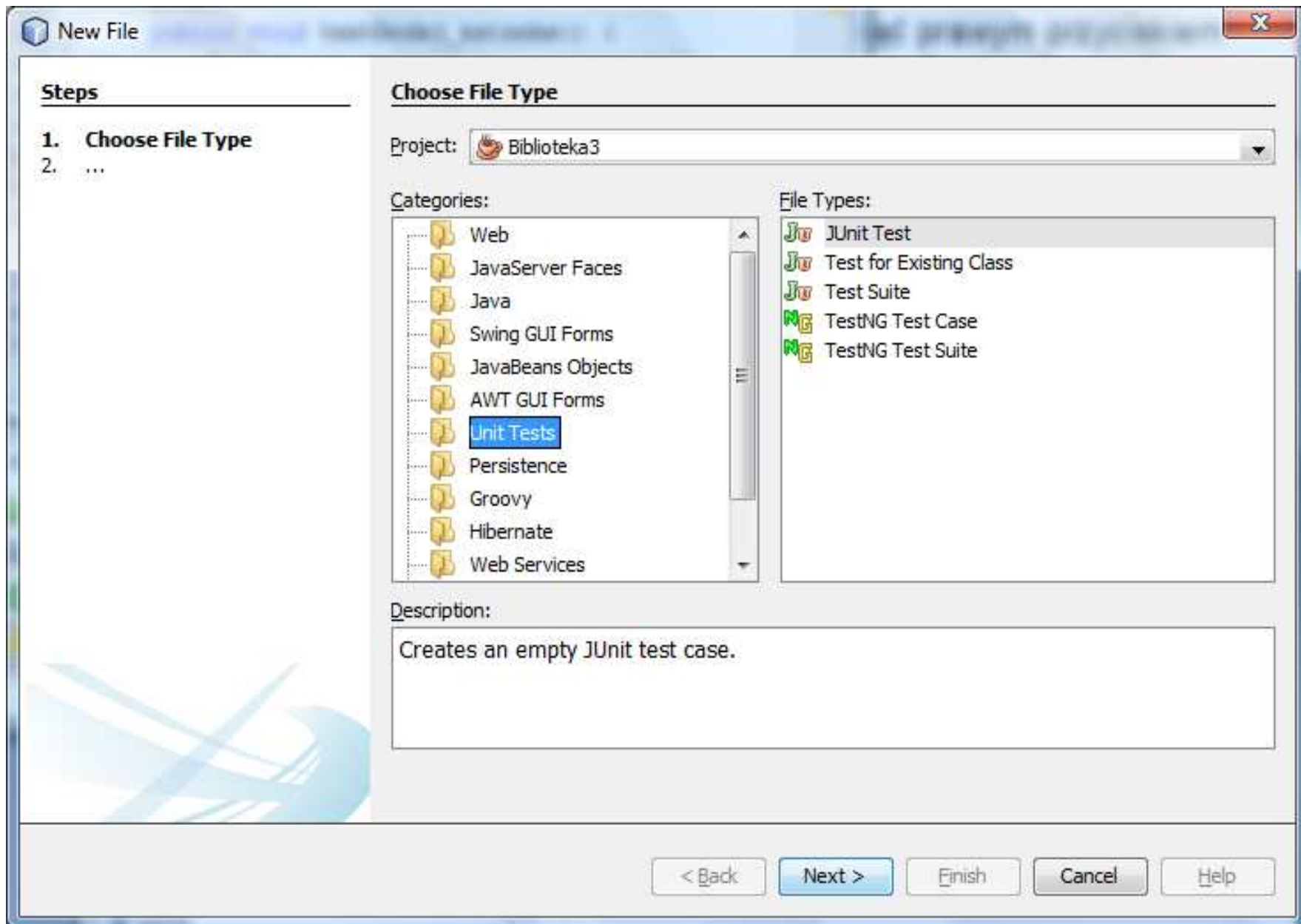
Należy wybrać framework typu **JUnit**. Na wybranym formularzu należy zaznaczyć właściwe opcje testowania.



2. Klasy testowe można utworzyć drugim sposobem wykorzystując **File Wizard** – wybór pozycji **Other...**



Wybór kategorii pliku: **Unit Test** oraz typ pliku testowego (prawa część formularza)

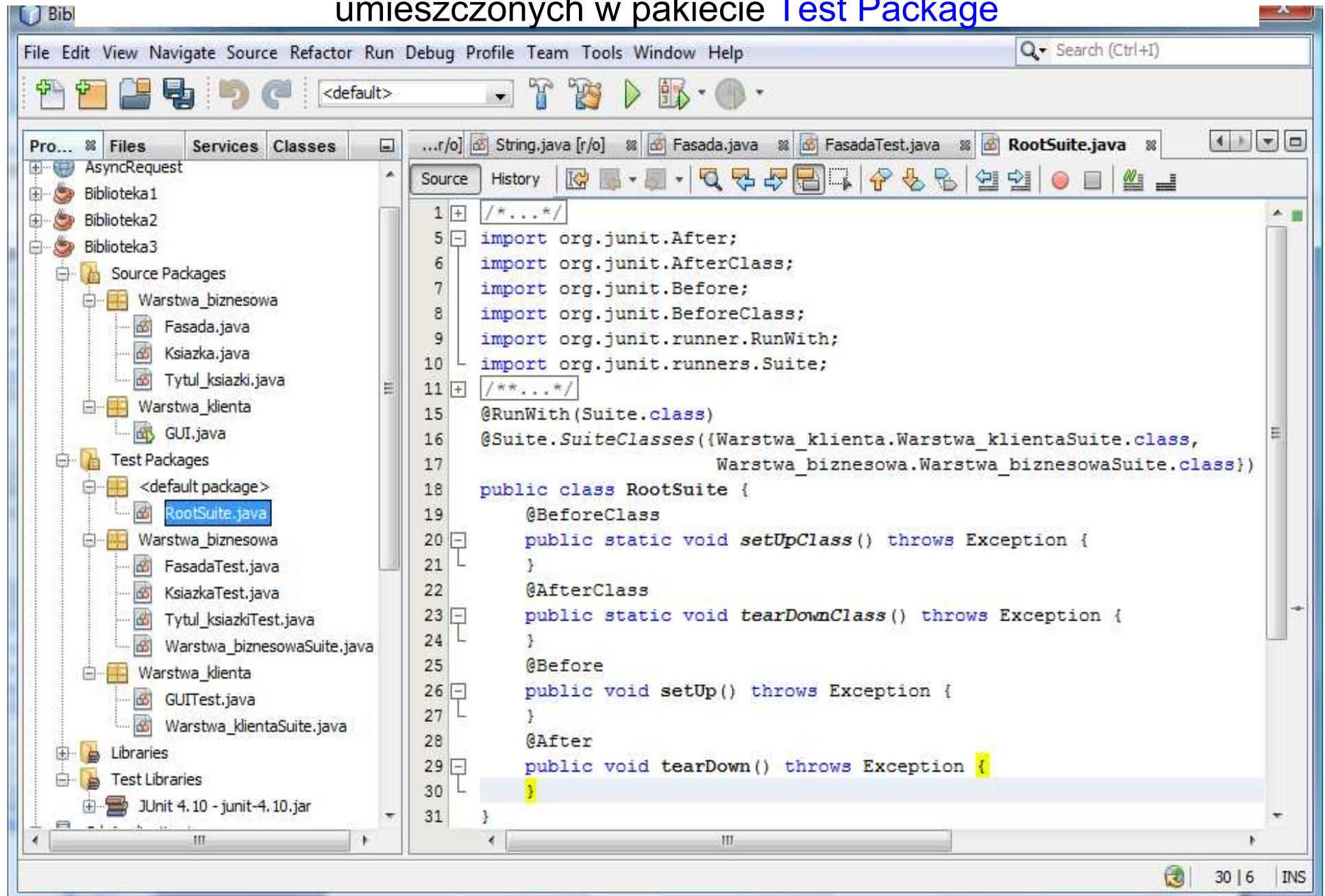


3. Wygenerowany zestaw szkieletu testów dla pakietu **Source Package**

The screenshot shows an IDE interface with a project structure on the left and a code editor on the right. The project structure includes a 'Test Packages' folder highlighted with a red box, containing sub-packages for 'Warstwa_biznesowa' and 'Warstwa_klienta', each with a suite and test files. The code editor displays the source code for 'Fasada.java' in the 'Warstwa_biznesowa' package, showing a class with a private field, a constructor, and three public methods.

```
1 package Warstwa_biznesowa;
2 /*...*/
6 import java.util.ArrayList;
7 /*...*/
11 public class Fasada {
12
13     private ArrayList<Tytul_książki> tytulys_książek =
14         new ArrayList<Tytul_książki>();
15
16     public Fasada() {
17     }
18
19     public ArrayList<Tytul_książki> getTytuly_książek() {
20         return tytulys_książek;
21     }
22
23     public void setTytuly_książek(ArrayList<Tytul_książki> val) {
24         this.tytulys_książek = val;
25     }
26
27     public void dodaj_tytul(String dane_tytul[]) {
28         Tytul_książki tytul_książki = new Tytul_książki(
29             dane_tytul[0],
30             dane_tytul[1]);
```

3.1. Plik zestawu wszystkich testów pakietu Source Package umieszczonych w pakiecie Test Package



The screenshot shows an IDE window with the following structure:

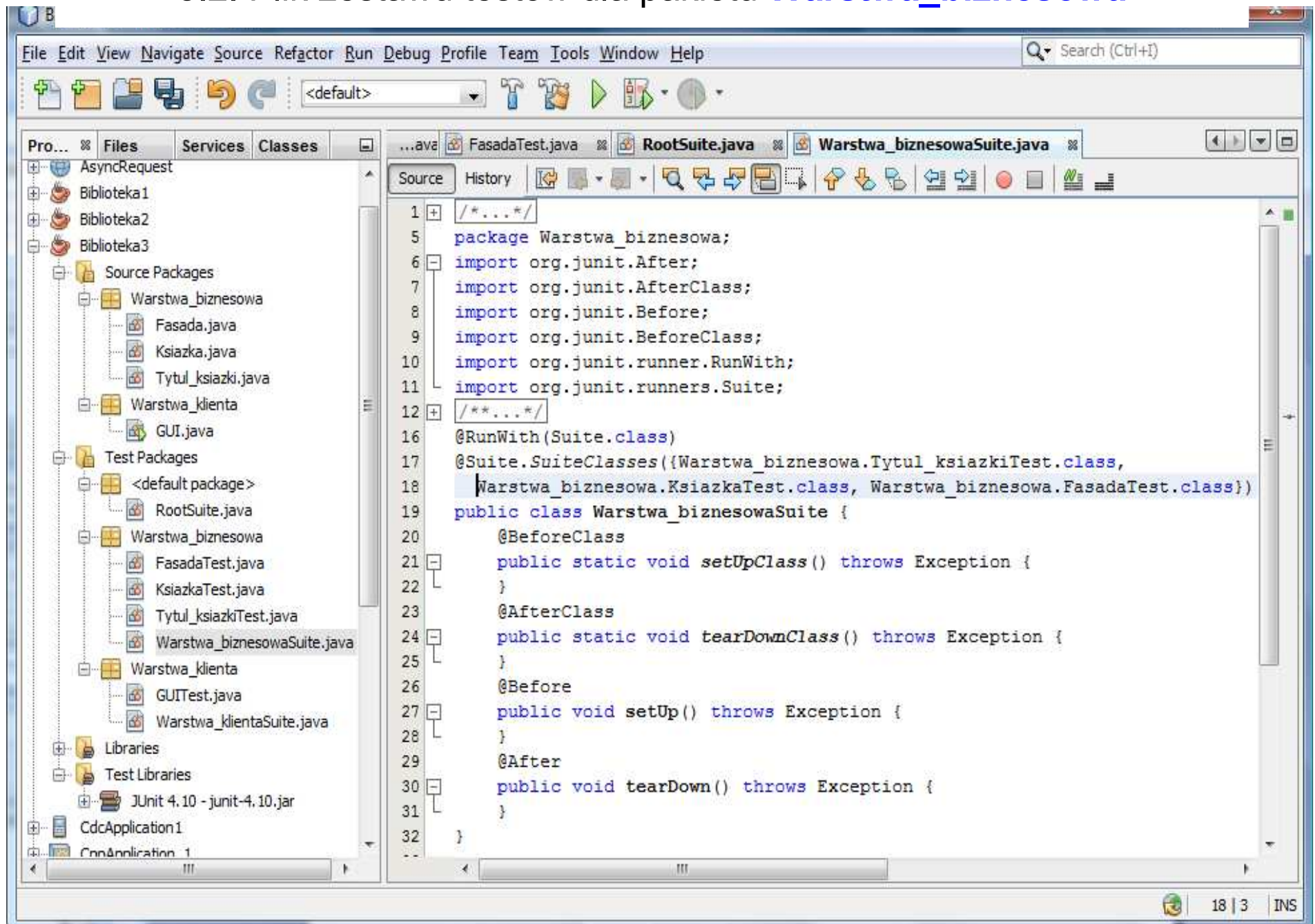
- Project Explorer (left):
 - AsyncRequest
 - Biblioteka1
 - Biblioteka2
 - Biblioteka3
 - Source Packages
 - Warstwa_biznesowa
 - Fasada.java
 - Ksiazka.java
 - Tytul_ksiazki.java
 - Warstwa_klienta
 - GUI.java
 - Test Packages
 - <default package>
 - RootSuite.java
 - Warstwa_biznesowa
 - FasadaTest.java
 - KsiazkaTest.java
 - Tytul_ksiazkiTest.java
 - Warstwa_biznesowaSuite.java
 - Warstwa_klienta
 - GUITest.java
 - Warstwa_klientaSuite.java
 - Libraries
 - Test Libraries
 - JUnit 4.10 - junit-4.10.jar

- Editor (right):
- String.java [r/o]
- Fasada.java
- FasadaTest.java
- RootSuite.java (active)

```
1  /*...*/
5  import org.junit.After;
6  import org.junit.AfterClass;
7  import org.junit.Before;
8  import org.junit.BeforeClass;
9  import org.junit.runner.RunWith;
10 import org.junit.runners.Suite;
11 /***...*/
15 @RunWith(Suite.class)
16 @Suite.SuiteClasses({Warstwa_klienta.Warstwa_klientaSuite.class,
17                      Warstwa_biznesowa.Warstwa_biznesowaSuite.class})
18 public class RootSuite {
19     @BeforeClass
20     public static void setUpClass() throws Exception {
21     }
22     @AfterClass
23     public static void tearDownClass() throws Exception {
24     }
25     @Before
26     public void setUp() throws Exception {
27     }
28     @After
29     public void tearDown() throws Exception {
30     }
31 }
```

Page 30 | 6 | INS

3.2. Plik zestawu testów dla pakietu **Warstwa_biznesowa**



The screenshot displays an IDE window with the following components:

- Project Explorer (Left):** Shows a project structure with 'Source Packages' containing 'Warstwa_biznesowa' (with files: Fasada.java, Ksiazka.java, Tytul_ksiazki.java) and 'Test Packages' containing 'Warstwa_biznesowa' (with files: FasadaTest.java, KsiazkaTest.java, Tytul_ksiazkiTest.java, Warstwa_biznesowaSuite.java).
- Editor (Right):** Displays the source code for 'Warstwa_biznesowaSuite.java'. The code is as follows:

```
1  /*...*/
5  package Warstwa_biznesowa;
6  import org.junit.After;
7  import org.junit.AfterClass;
8  import org.junit.Before;
9  import org.junit.BeforeClass;
10 import org.junit.runner.RunWith;
11 import org.junit.runners.Suite;
12 /*...*/
16 @RunWith(Suite.class)
17 @Suite.SuiteClasses({Warstwa_biznesowa.Tytul_ksiazkiTest.class,
18 Warstwa_biznesowa.KsiazkaTest.class, Warstwa_biznesowa.FasadaTest.class})
19 public class Warstwa_biznesowaSuite {
20     @BeforeClass
21     public static void setUpClass() throws Exception {
22     }
23     @AfterClass
24     public static void tearDownClass() throws Exception {
25     }
26     @Before
27     public void setUp() throws Exception {
28     }
29     @After
30     public void tearDown() throws Exception {
31     }
32 }
```
- Bottom Status Bar:** Shows '18 | 3 | INS'.

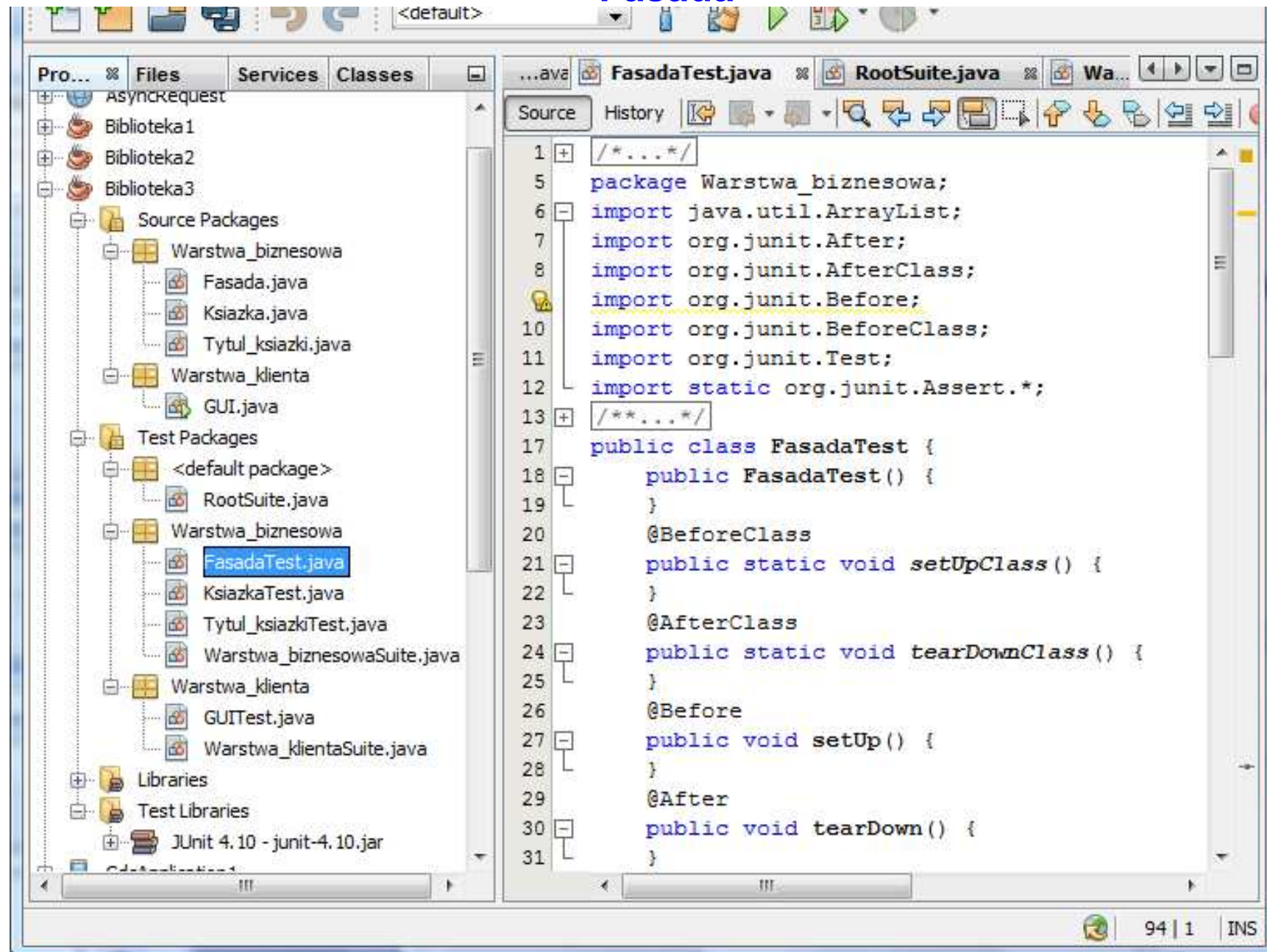
3.3. Plik zestawu testów dla pakietu **Warstwa_klienta**

The screenshot displays an IDE window with the following components:

- Project Explorer (Left):** Shows a project structure with 'Source Packages' and 'Test Packages'. Under 'Test Packages', the file 'Warstwa_klientaSuite.java' is selected and highlighted in blue.
- Source Editor (Right):** Displays the code for 'Warstwa_klientaSuite.java'. The code is as follows:

```
1  /*...*/
5  package Warstwa_klienta;
6  import org.junit.After;
7  import org.junit.AfterClass;
8  import org.junit.Before;
9  import org.junit.BeforeClass;
10 import org.junit.runner.RunWith;
11 import org.junit.runners.Suite;
12 /*...*/
16 @RunWith(Suite.class)
17 @Suite.SuiteClasses({Warstwa_klienta.GUITest.class})
18 public class Warstwa_klientaSuite {
19     @BeforeClass
20     public static void setUpClass() throws Exception {
21     }
22     @AfterClass
23     public static void tearDownClass() throws Exception {
24     }
25     @Before
26     public void setUp() throws Exception {
27     }
28     @After
29     public void tearDown() throws Exception {
30     }
31 }
```
- Bottom Status Bar:** Shows '32 | 1 | INS'.

3.4. Wygenerowane metody testowe w klasie **FasadaTest** dla klasy typu **Fasada**



The screenshot displays an IDE window with the following components:

- Project Explorer (Left):** Shows a project structure with packages like `Warstwa_biznesowa` and `Test Packages`. The file `FasadaTest.java` is highlighted in the `Test Packages` folder.
- Source Editor (Right):** Shows the source code of `FasadaTest.java`. The code is as follows:

```
1  /*...*/
5  package Warstwa_biznesowa;
6  import java.util.ArrayList;
7  import org.junit.After;
8  import org.junit.AfterClass;
9  import org.junit.Before;
10 import org.junit.BeforeClass;
11 import org.junit.Test;
12 import static org.junit.Assert.*;
13 /*...*/
17 public class FasadaTest {
18     public FasadaTest() {
19     }
20     @BeforeClass
21     public static void setUpClass() {
22     }
23     @AfterClass
24     public static void tearDownClass() {
25     }
26     @Before
27     public void setUp() {
28     }
29     @After
30     public void tearDown() {
31     }
```
- Page-Footer (Bottom Right):** Shows the page number `94 | 1` and the text `INS`.

3.5. Wygenerowane metody testowe w klasie **FasadaTest** dla klasy typu **Fasada** cd

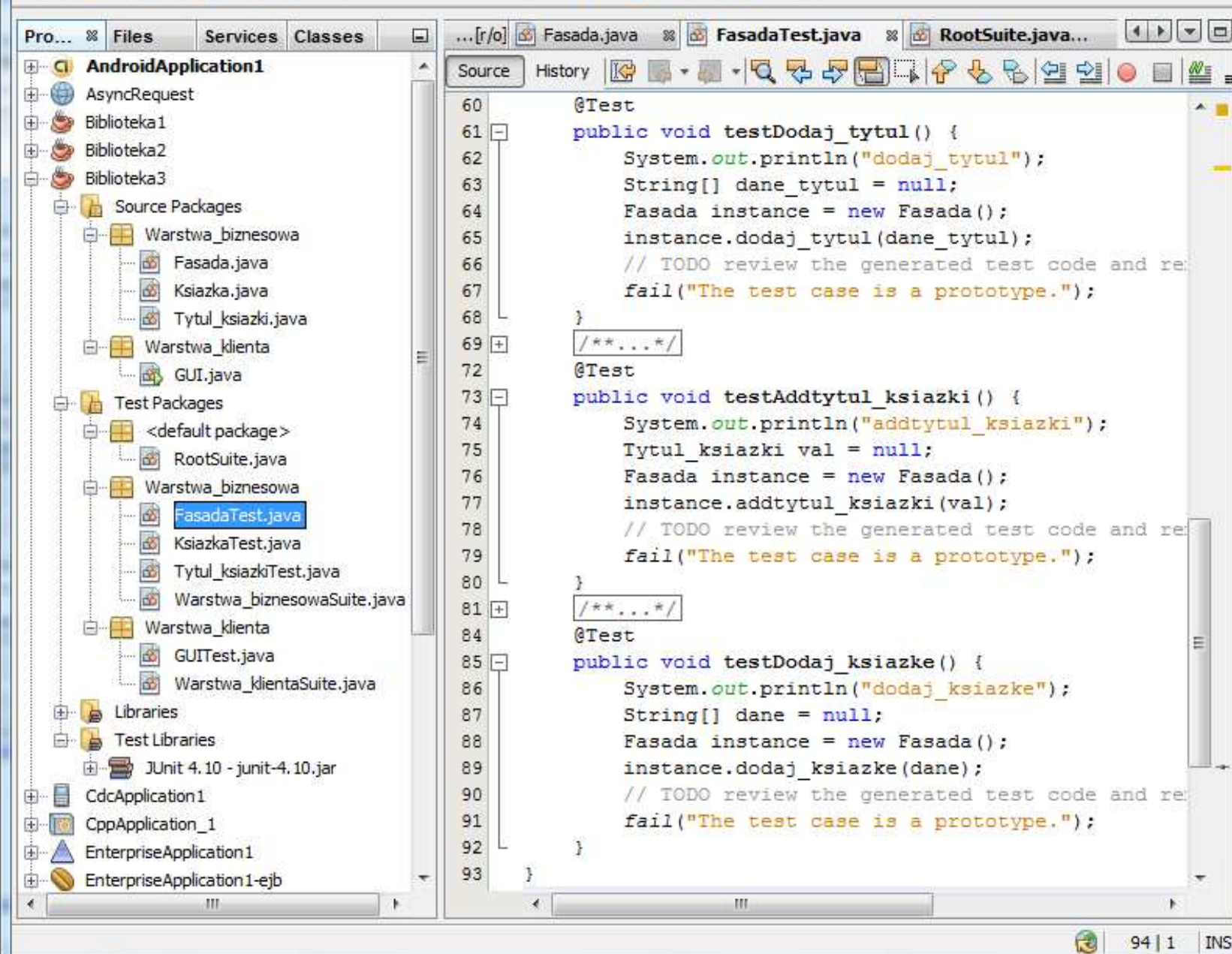
The screenshot shows an IDE window with the following structure:

- Project: Biblioteka3
- Source Packages:
 - Warstwa_biznesowa
 - Fasada.java
 - Ksiazka.java
 - Tytul_ksiazki.java
 - Warstwa_klienta
 - GUI.java
- Test Packages:
 - <default package>
 - RootSuite.java
 - Warstwa_biznesowa
 - FasadaTest.java** (highlighted)
 - KsiazkaTest.java
 - Tytul_ksiazkiTest.java
 - Warstwa_biznesowaSuite.java
 - Warstwa_klienta
 - GUITest.java
 - Warstwa_klientaSuite.java
- Libraries:
 - JUnit 4.10 - junit-4.10.jar

The code in **FasadaTest.java** is as follows:

```
32  /**...*/
35  @Test
36  public void testGetTytuly_ksiazek() {
37      System.out.println("getTytuly_ksiazek");
38      Fasada instance = new Fasada();
39      ArrayList expResult = null;
40      ArrayList result = instance.getTytuly_ksiazek();
41      assertEquals(expResult, result);
42      // TODO review the generated test code and
43      // remove the default call to fail.
44      fail("The test case is a prototype.");
45  }
46  /**...*/
49  @Test
50  public void testSetTytuly_ksiazek() {
51      System.out.println("setTytuly_ksiazek");
52      ArrayList<Tytul_ksiazki> val = null;
53      Fasada instance = new Fasada();
54      instance.setTytuly_ksiazek(val);
55      // TODO review the generated test code and remove
56      fail("The test case is a prototype.");
57  }
```

3.6. Wygenerowane metody testowe w klasie **FasadaTest** dla klasy typu **Fasada** cd



The screenshot shows an IDE window with the following structure:

- Project: AndroidApplication1
 - Source Packages
 - Warstwa_biznesowa
 - Fasada.java
 - Ksiazka.java
 - Tytul_ksiazki.java
 - Warstwa_klienta
 - GUI.java
 - Test Packages
 - <default package>
 - RootSuite.java
 - Warstwa_biznesowa
 - FasadaTest.java** (selected)
 - KsiazkaTest.java
 - Tytul_ksiazkiTest.java
 - Warstwa_biznesowaSuite.java
 - Warstwa_klienta
 - GUITest.java
 - Warstwa_klientaSuite.java
 - Libraries
 - Test Libraries
 - JUnit 4.10 - junit-4.10.jar

```
60     @Test
61     public void testDodaj_tytul() {
62         System.out.println("dodaj_tytul");
63         String[] dane_tytul = null;
64         Fasada instance = new Fasada();
65         instance.dodaj_tytul(dane_tytul);
66         // TODO review the generated test code and re:
67         fail("The test case is a prototype.");
68     }
69     /**...*/
72     @Test
73     public void testAddtytul_ksiazki() {
74         System.out.println("addtytul_ksiazki");
75         Tytul_ksiazki val = null;
76         Fasada instance = new Fasada();
77         instance.addtytul_ksiazki(val);
78         // TODO review the generated test code and re:
79         fail("The test case is a prototype.");
80     }
81     /**...*/
84     @Test
85     public void testDodaj_ksiazke() {
86         System.out.println("dodaj_ksiazke");
87         String[] dane = null;
88         Fasada instance = new Fasada();
89         instance.dodaj_ksiazke(dane);
90         // TODO review the generated test code and re:
91         fail("The test case is a prototype.");
92     }
93 }
```

Page 94 | 1 | INS

4. Adnotacje określające sposób i moment testowania

<http://www.vogella.com/articles/JUnit/article.html>

Annotation	Description
@Test public void method().	The annotation @Test identifies that a method is a test method.
@Before public void method()	Will execute the method before each test. This method can prepare the test environment (e.g. read input data, initialize the class).
@After public void method()	Will execute the method after each test. This method can cleanup the test environment (e.g. delete temporary data, restore defaults).
@BeforeClass public void method()	Will execute the method once, before the start of all tests. This can be used to perform time intensive activities, for example to connect to a database.
@AfterClass public void method()	Will execute the method once, after all tests have finished. This can be used to perform clean-up activities, for example to disconnect from a database.
@Ignore	Will ignore the test method. This is useful when the underlying code has been changed and the test case has not yet been adapted. Or if the execution time of this test is too long to be included.
@Test (expected = Exception.class)	Fails, if the method does not throw the named exception.
@Test(timeout=100)	Fails, if the method takes longer than 100 milliseconds.

5. Metody wspomagające ocenę wyniku testu

<http://www.vogella.com/articles/JUnit/article.html>

Statement	Description
fail(String)	Let the method fail. Might be used to check that a certain part of the code is not reached. Or to have failing test before the test code is implemented.
assertTrue(true) / assertFalse(false)	Will always be true / false. Can be used to predefine a test result, if the test is not yet implemented.
assertTrue([message],boolean condition)	Checks that the boolean condition is true.
assertEquals([String message], expected, actual)	Tests that two values are the same. Note: for arrays the reference is checked not the content of the arrays.
assertEquals([String message], expected, actual, tolerance)	Test that float or double values match. The tolerance is the number of decimals which must be the same.
assertNull([message], object)	Checks that the object is null.
assertNotNull([message], object)	Checks that the object is not null.
assertSame([String], expected, actual)	Checks that both variables refer to the same object.
assertNotSame([String], expected, actual)	Checks that both variables refer to different objects.

6. Pomocnicze klasy zdefiniowane przez użytkownika do przeprowadzenia testów jednostkowych metod `dodaj_tytul` i `addtytul_książki` klasy `Fasada`

```
Source History [Icons]
39
40     public Tytul_książki Tytul(String dane_tytul[]) {
41         Tytul_książki tytul_książki = new Tytul_książki();
42         tytul_książki.setTytul(dane_tytul[0]);
43         tytul_książki.setNazwisko(dane_tytul[1]);
44         tytul_książki.setImie(dane_tytul[2]);
45         tytul_książki.setISBN(dane_tytul[3]);
46         tytul_książki.setWydawnictwo(dane_tytul[4]);
47         return tytul_książki;
48     }
49     public ArrayList<Tytul_książki> Tytuly() {
50         ArrayList<Tytul_książki> tytuly = new ArrayList();
51         String dane_tytulu1[] = {"tytul1", "Jan", "Kowalski", "12345", "W1"};
52         String dane_tytulu2[] = {"tytul2", "Piotr", "Nowak", "67891", "W2"};
53         tytuly.add(Tytul(dane_tytulu1));
54         tytuly.add(Tytul(dane_tytulu2));
55         return tytuly;
56     }
```

Definicja metody tworzącej wzorcowy obiekt typu `Tytul_książki`

Definicja metody tworzącej wzorcową kolekcję typu `ArrayList` z elementami typu `Tytul_książki`

Utworzenie wzorcowego obiektu typu `Tytul_książki`

Utworzenie wzorcowej kolekcji typu `ArrayList` zawierającej elementy - obiekty typu `Tytul_książki`

6.2. Uzupełnione metody do testowania metod `dodaj_tytul` i `addtytul_książki` klasy `Fasada`

```
85     @Test
86     public void testDodaj_tytul() {
87         System.out.println("dodaj_tytul");
88         String[] dane_tytul = {"tytul1", "Jan", "Kowalski", "12345", "W1"};
89         Fasada instance = new Fasada();
90         instance.dodaj_tytul(dane_tytul);
91         Tytul_książki expectedResult = Tytul(dane_tytul);
92         Tytul_książki result = instance.getTytuly_książek().get(0);
93         assertEquals(expectedResult, result);
94         // TODO review the generated test code and remove the default call to fail()
95         //fail("The test case is a prototype.");
96     }
97     @Test
98     public void testAddtytul_książki() {
99         System.out.println("addtytul_książki");
100        String[] dane_tytul = {"tytul1", "Jan", "Kowalski", "12345", "W1"};
101        Tytul_książki val = Tytul(dane_tytul);
102        Fasada instance = new Fasada();
103        instance.addtytul_książki(val);
104        Tytul_książki result = instance.getTytuly_książek().get(0);
105        assertEquals(val, result);
106        instance.addtytul_książki(val);
107        int result1 = instance.getTytuly_książek().size();
108        assertTrue("Bład", 1 == result1);
109        // TODO review the generated test code and remove the default call to fail()
110        //fail("The test case is a prototype.");
111    }
```

Porównanie
dodanego
tytułu
pobranego
z aplikacji z
tytułem
wzorcowym

Sprawdzenie
,czy po
próbie
dodania tego
samego
tytułu liczba
tytułów nie
zmieni się

Porównanie
dodanego
tytułu
pobranego
z aplikacji z
tytułem
wzorcowym

6.3. Pomocnicze klasy zdefiniowane przez użytkownika do przeprowadzenia testów jednostkowych metody `dodaj_ksiazke` klasy `Fasada`

```
116 public Ksiazka Ksiazka(String dane1[], Tytul_ksiazki dane2) {
117     Ksiazka ksiazka = new Ksiazka();
118     ksiazka.setNumer(Integer.parseInt(dane1[1]));
119     ksiazka.setTytul_ksiazki(dane2);
120     return ksiazka;
121 }
122 public ArrayList<Ksiazka> Ksiazki(String dane_tytulu[], String[]... dane) {
```

Definicja metody tworzącej wzorcowy obiekt typu `Ksiazka`

Definicja metody tworzącej wzorcową kolekcję typu `ArrayList` z elementami typu `Ksiazka`

Utworzenie wzorcowej kolekcji typu `ArrayList` zawierającej elementy - obiekty typu `Ksiazka`

Utworzenie wzorcowego obiektu typu `Ksiazka`

Zmienna liczba parametrów typu `String[]`, przekazywana do metody w postaci tablicy `dane` takich elementów

6.4. Uzupełniona metoda do testowania metody `dodaj_książke` klasy **Fasada**

```
@Test
public void testDodaj_książke() {
    System.out.println("dodaj_książke");
    String[] dane_tytul1 = {"tytul1", "Jan", "Kowalski", "12345", "W1"};
    String[] dane_tytul2 = {"tytul2", "Piotr", "Nowak", "67891", "W2"};
    Fasada instance = new Fasada();
    instance.dodaj_tytul(dane_tytul1);
    instance.dodaj_tytul(dane_tytul2);
    String dane_1[] = {"12345", "1"};
    String dane_2[] = {"67891", "2"};
    String dane_3[] = {"12345", "2"};
    String dane_4[] = {"54321", "1"};
    instance.dodaj_książke(dane_1);
    instance.dodaj_książke(dane_1);
    instance.dodaj_książke(dane_2);
    instance.dodaj_książke(dane_3);
    instance.dodaj_książke(dane_4);
    assertEquals(Książki(dane_tytul1, dane_1, dane_3),
        instance.getTytuly_książek().get(0).getKsiążka());
    assertEquals(Książki(dane_tytul2, dane_2),
        instance.getTytuly_książek().get(1).getKsiążka());
    // TODO review the generated test code and remove the default call to
    //fail("The test case is a prototype.");
}
```

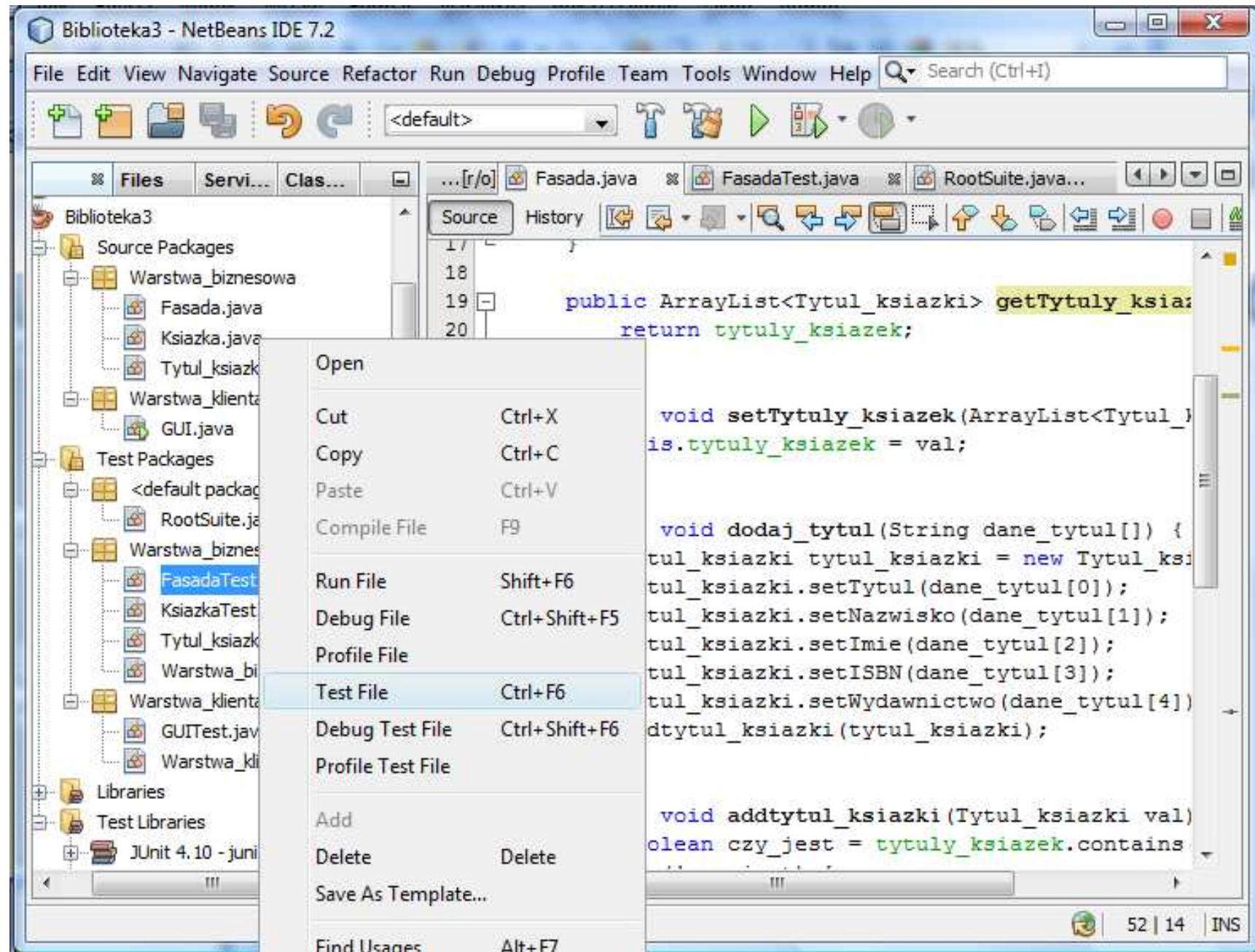
Metoda `Książki` zwraca wzorcowe kolekcje książek: dla tytułu z ISBN=12345 (2 książki) oraz ISBN=67891 (1 książka)

Danie 2 tytułów: z ISBN=12345 i ISBN=67891

Dodanie 3 książek: 2 do tytułu z ISBN=12345 i 1 do tytułu o ISBN=67891

Metoda zwraca kolekcję książek pobranych z aplikacji dla tytułu z ISBN=12345 oraz ISBN=67891

7. Uruchomienie testów



8. Wynik testowania

The screenshot displays the NetBeans IDE 7.2 interface for a project named 'Biblioteka3'. The left sidebar shows the project structure with 'Test Packages' containing 'FasadaTest.java'. The central editor shows a snippet of Java code for a test method:

```
125     }
126     @Test
127     public void testDodaj_ksiazke() {
128         System.out.println("dodaj_ksiazke");
129         String[] dane_tytul1 = {"tytul1", "Jan", "Kowalski", "12"};
130         String[] dane_tytul2 = {"tytul2", "Bioty", "Nowak", "678"};
```

The bottom output window, titled 'Output - Biblioteka3 (test)', shows the following test results:

```
Ant suite
100,00 %
All 5 tests passed.(0,035 s)

[TestNG] Running:
  Command line suite

getTytuly_ksiazek
setTytuly_ksiazek
dodaj_tytul
addtytul_ksiazki
dodaj_ksiazke

-----
Command line suite
Total tests run: 5, Failures: 0, Skips: 0
```

The status bar at the bottom right indicates '139 | 40 | INS'.