

Instrukcja 10

Laboratorium 13

Testy akceptacyjne z wykorzystaniem narzędzia FitNesse

Cel laboratorium:**Nabywanie umiejętności przygotowywania testów akceptacyjnych za pomocą narzędzia FitNesse**

1. Wg wskazówek podanych w **Dodatk 1 p. 1**, należy zainstalować narzędzie **FitNesse** oraz wykonać projekt w środowisku **NetBeans 8.1**. Następnie, wg kolejnych przykładów w **Dodatk 1**, należy dodawać podane dalej **testy akceptacyjne** wybranych funkcji oprogramowania zaprojektowanego podczas lab. 2-11, uruchamianych z pośrednictwem klasy typu **Fasada** warstwy biznesowej. Poniżej, w kolejnych punktach instrukcji podano, ile i jakie testy należy wykonać w jedno- i dwu-osobowych grupach.
2. Należy wykonać testy akceptacyjne metod klasy typu **Fasada**, która przetwarza dane, które będą wykorzystywane w testach kolejnych metod klasy typu **Fasada**, zaproponowanych w kolejnych punktach.
W tabelce podano informację dotyczącą wyboru metod do testowania oraz przykładów rozwiązań.

Grupa	Liczba metod do testowania klasy opartej na wzorcu Fasada	Przykłady metod wybranych do testowania	Przykłady testów
		TAplikacja	Klasy testujące
1 osoba	1	Dodaj_produkt	Test_dodawanie_produktu (p.3.8, Dodatek1)
2 osoby	2	Dodaj_produkt, Dodaj_rachunek	Test_dodawanie_produktu (p.3.8, Dodatek1) Test_dodawanie_rachunku (p.3.9, Dodatek 1)

W p.3.6 **Dodatku 1** zdefiniowano tabelkę z wartościami wzorcowymi danych wejściowych i wyjściowych, wykorzystanych w testach akceptacyjnych klasy typu **TAplikacja**. Należy opracować podobny zestaw danych wzorcowych, który należy wykorzystać przy budowie własnych testów akceptacyjnych.

3. Należy wykonać testy akceptacyjne metod klasy opartej na wzorcu **Fasada**, które korzystają z wyników przetwarzania danych realizowanych przez testy z p. 2.

Grupa	Liczba metod do testowania klasy opartej na wzorcu Fasada	Przykłady wybranych do testowania	Przykłady testów
		TAplikacja	Klasy testujące
1 osoba	1	Dodaj_rachunek,	Test_dodawanie_rachunku (p.3.9, Dodatek 1)
2 osoby	2	Dodaj_Zakup, Podaj_wartosc	Test_Dodawanie_zakupu (p.3.10, Dodatek 1) Test_obliczanie_wartosci_rachunku (p.3.11, Dodatek 1)

Należy rozszerzyć zestaw danych wzorcowych z p.2, który należy wykorzystać przy budowie własnych testów akceptacyjnych w p.3.

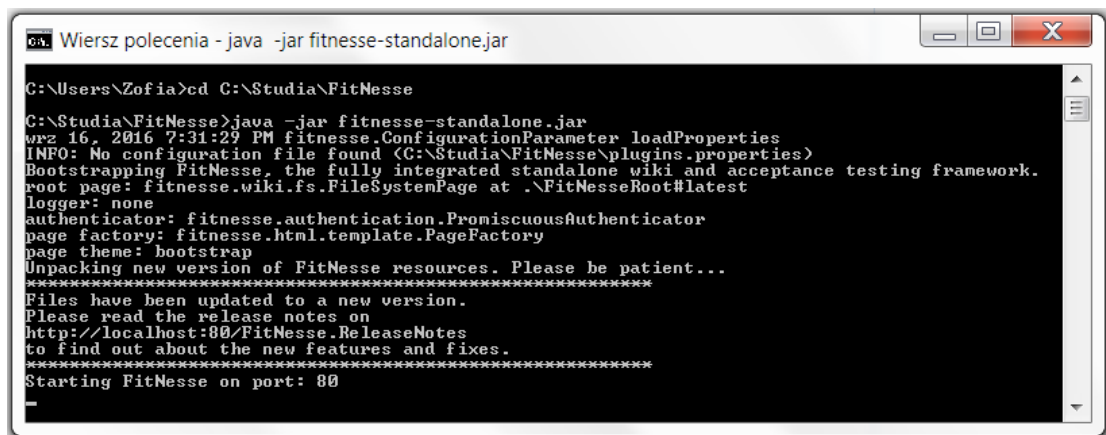
Dodatek 1

1. Instalacja narzędzia FitNesse - wersja v20160618

- 1.1. Należy pobrać oprogramowanie **FitNesse** ze strony <http://www.fitnessse.org>, które zawiera zintegrowane narzędzie do testowania oraz witrynę typu **Wiki** do tworzenia stron służących do uruchamiania oprogramowania testującego oprogramowanie.
- 1.2. Należy wykonać folder np. **FitNesse**, gdzie należy umieścić pobrany plik **fitnessse-standalone.jar** ze strony <http://www.fitnessse.org/FitNesseDownload>. Podczas tworzenia testów akceptacyjnych folder ten będzie służył również do przechowywania systemu stron służących do uruchamiania testów akceptacyjnych oprogramowania. W celu uruchomienia narzędzia należy z linii poleceń wpisać:

java -jar fitnessse-standalone.jar

Poniżej pokazano komunikaty po pierwszym uruchomieniu narzędzia.

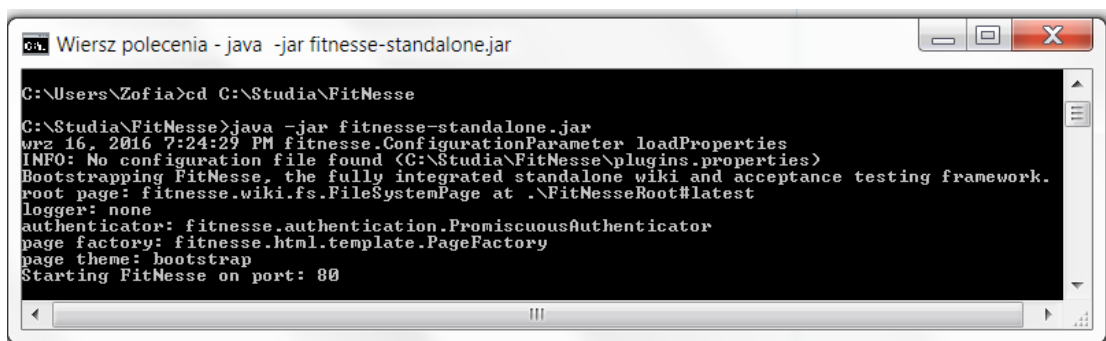


```
Wiersz polecenia - java -jar fitnessse-standalone.jar

C:\Users\Zofia>cd C:\Studia\FitNesse

C:\Studia\FitNesse>java -jar fitnessse-standalone.jar
wrz 16, 2016 7:31:29 PM fitnessse.ConfigurationParameter loadProperties
INFO: No configuration file found (C:\Studia\FitNesse\plugins.properties)
Bootstrapping FitNesse, the fully integrated standalone wiki and acceptance testing framework.
root page: fitnessse.wiki.fs.FileSystemPage at .\FitNesseRoot#latest
logger: none
authenticator: fitnessse.authentication.PromiscuousAuthenticator
page factory: fitnessse.html.template.PageFactory
page theme: bootstrap
Unpacking new version of FitNesse resources. Please be patient...
*****
Files have been updated to a new version.
Please read the release notes on
http://localhost:80/FitNesse.ReleaseNotes
to find out about the new features and fixes.
*****
Starting FitNesse on port: 80
```

Poniżej pokazano komunikaty po kolejnym uruchomieniu narzędzia **FitNesse**.

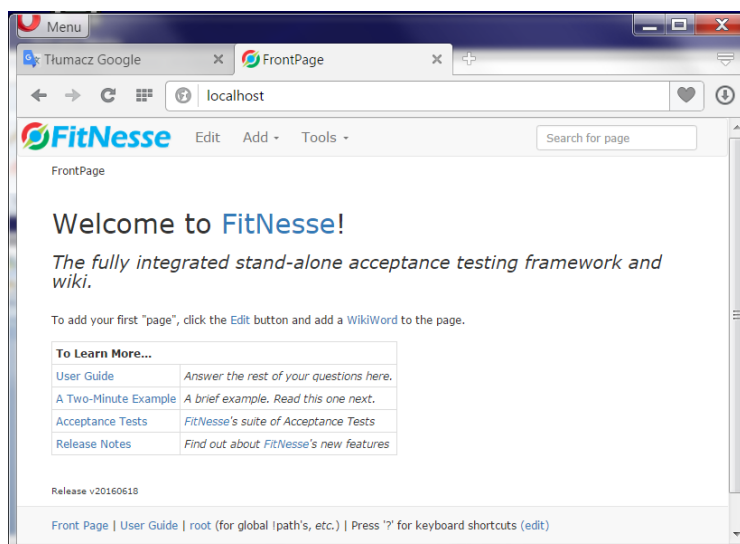


```
Wiersz polecenia - java -jar fitnessse-standalone.jar

C:\Users\Zofia>cd C:\Studia\FitNesse

C:\Studia\FitNesse>java -jar fitnessse-standalone.jar
wrz 16, 2016 7:24:29 PM fitnessse.ConfigurationParameter loadProperties
INFO: No configuration file found (C:\Studia\FitNesse\plugins.properties)
Bootstrapping FitNesse, the fully integrated standalone wiki and acceptance testing framework.
root page: fitnessse.wiki.fs.FileSystemPage at .\FitNesseRoot#latest
logger: none
authenticator: fitnessse.authentication.PromiscuousAuthenticator
page factory: fitnessse.html.template.PageFactory
page theme: bootstrap
Starting FitNesse on port: 80
```

Uruchomienie witryny typu **Wiki** nastąpi po wpisaniu w przeglądarce adresu <http://localhost>. Poniżej pokazano stronę główną uruchomionego środowiska do tworzenia i wykonania testów akceptacyjnych.



- 1.3. Jeśli nie startuje prawidłowo witryna typu *Wiki* narzędzia *FitNesse*, należy uruchomić to narzędzie w następujący sposób, podając **wolny numer portu** np.:

```
java -jar fitnessse-standalone.jar -p 8080
```

i w przeglądarce należy wtedy wpisać: <http://localhost:8080>.

- 1.4. Na stronie <http://localhost/FitNesse.UserGuide.WritingAcceptanceTests> jest dostępna informacja typu *Przewodnik użytkownika* w zakresie technologii testowania za pomocą *FitNesse*.

2. Modyfikacje kodu przedstawionego w Dodatku1 instrukcji 5-7 – identyczne zmiany należało wykonać podczas tworzenia testów jednostkowych.

Dodanie generowania wyjątku w przypadku niepoprawnej wartości pierwszego elementu tablicy *dane* -zmiana definicji podanej w instrukcji 6. Pełna walidacja poprawności danych wejściowych powinna być realizowana przez warstwę klienta aplikacji!

```
public class TFabryka {
    public TProdukt1 Podaj_produkt(String dane[]) {
        TProdukt1 produkt = null;
        TPromocja promocja;
        switch (Integer.parseInt(dane[0])) {
            case 0:
                produkt = new TProdukt1(dane[1], Float.parseFloat(dane[2]));
                break;
            case 1:
                promocja = new TPromocja(Float.parseFloat(dane[3]));
                produkt = new TProdukt1(dane[1], Float.parseFloat(dane[2]), promocja);
                break;
            case 2:
                produkt = new TProdukt2(dane[1], Float.parseFloat(dane[2]), Float.parseFloat(dane[3]));
                break;
            case 3:
                promocja = new TPromocja(Float.parseFloat(dane[4]));
                produkt = new TProdukt2(dane[1], Float.parseFloat(dane[2]), Float.parseFloat(dane[3]),
                    promocja);

                break;
            default:
                throw new IllegalArgumentException(0); //generowanie wyjątku z powodu
                // niepoprawnej wartości elementu tablicy dane o indeksie 0.
        }
        return produkt;
    }
}
```

W klasie *TAplikacja* dodano do definicji metod, wywołujących metodę klasy *TFabryka* dodano klauzulę `throws IllegalArgumentException` - zmiana definicji podanej w instrukcjach 6 i 7:

```
public void Dodaj_produkt(String dane[]) throws IllegalArgumentException //instrukcja 6
```

```
public void Wstaw_zakup(int nr, int ile, String dane[]) throws IllegalArgumentException //instrukcja 7
```

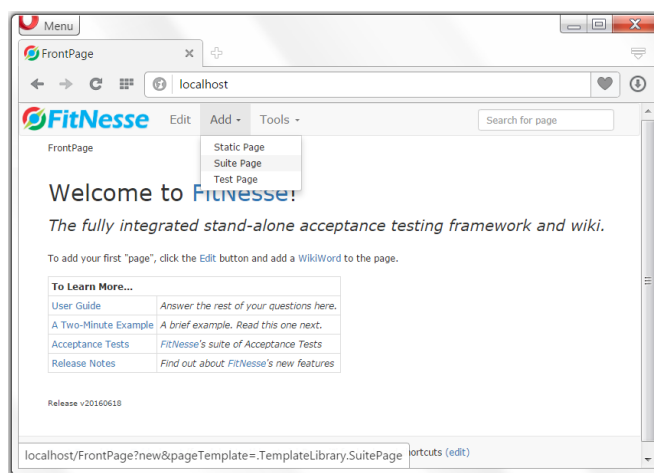
```
public static void main(String args[]) throws IllegalArgumentException //instrukcje 6 i 7
```

Dodatkowo, klasy pakietu *rachunek1*, podane w części Dodatek1 instrukcji 5 powinny być klasami publicznymi:

```
public class TRachunek
public class TZakup
public class TProdukt1
public class TProdukt2
public class TPromocja
```

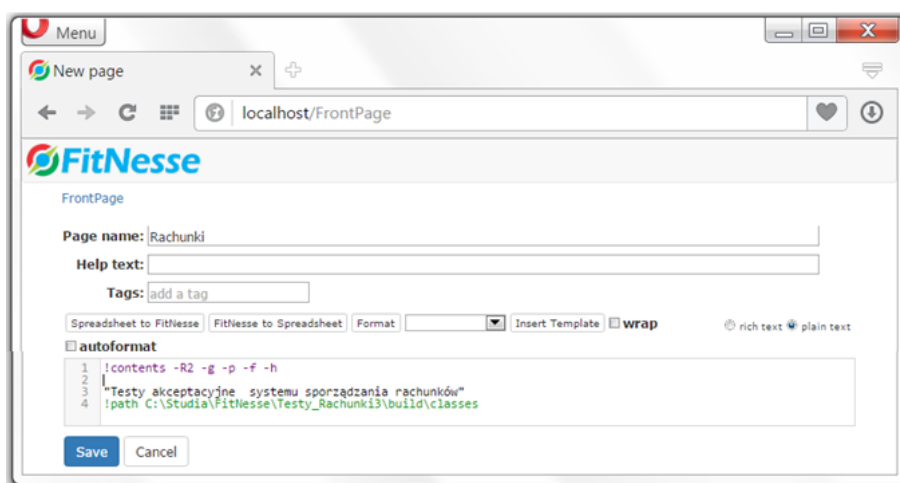
3. Przykład tworzenia testów akceptacyjnych warstwy biznesowej aplikacji

- 3.1. W celu utworzenia projektu zawierającego kod do testowania, należy wybrać w **Menu Bar** pozycję **Files**. Na tej liście kliknąć na pozycję **New Project**. W oknie **New Project**, w liście **Categories** należy wybrać pozycję **Java**, a w liście **Projects** należy wybrać pozycję **Java Class Library** i kliknąć na przycisk **Next**. W kolejnym formularzu należy wpisać nazwę projektu w polu **Project Name** i wybrać położenie projektu w polu **Project Location**. W przykładzie projekt ma nazwę **Testy_Rachunki3**.
- 3.2. W zakładce **Projects**, w folderze **Source Packages** umieścić kopię pakietu z oprogramowaniem do testowania, wykonanym podczas lab 2- lab 11. W przykładzie jest to pakiet **rachunek1**, wykorzystany podczas testów jednostkowych.
- 3.3. W zakładce **Projects**, w folderze **Source Packages** należy wykonać nowy pakiet, w którym umieszczane będą klasy realizujące testy akceptacyjne - w przykładzie jest to pakiet **testy_fitnessse_fixture**.
- 3.4. Należy wykonać stronę internetową, która będzie przechowywać połączenia do poszczególnych stron internetowych reprezentujących testy akceptacyjne poszczególnych funkcji warstwy biznesowej. W tym celu należy w **Menu Bar** narzędzia **FitNesse**, uruchomionego w p. 1.2 lub 1.3, wybrać pozycję **Add** i następnie pozycję **Suite Page** (rysunek poniżej).

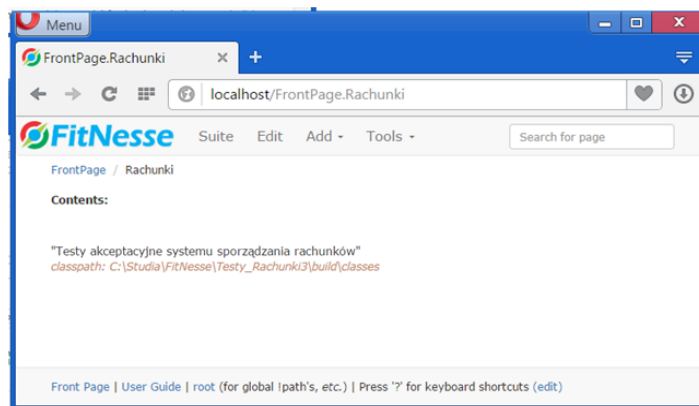


Poniżej, w formularzu strony typu **Suite Page**, jako głównej strony testów akceptacyjnych, nadano nazwę **Rachunki** w polu **Page name**. W celu tworzenia zestawu połączeń do kolejnych testów należy zachować znacznik **!contents** umożliwiający umieszczanie połączeń do kolejnych stron (tworzonych w dalszej części instrukcji) uruchamiających testy akceptacyjne. **Uwaga: Nazwy stron powinny w nazwie zawierać przynajmniej jedną dużą literę!!!**

W zawartości strony umieszczono komentarz dotyczący przedmiotu testowania oraz ścieżkę dostępu do kodu obsługującego testowanie w katalogu*build\classes* projektu.



W przykładzie wykonano projekt **Testy_Rachunki3** (p.3.1), zawierający kod do testowania w pakiecie **rachunek1** oraz kod testujący w pakiecie **testy_fitnessse_fixture**, dostępne w katalogu **C:\Studia\FitNesse\Testy_Rachunki3\build\classes**. Poniżej przedstawiono widok strony głównej systemu testów akceptacyjnych **Rachunki**, po zatwierdzeniu zawartości tej strony za pomocą przycisku **Save**.

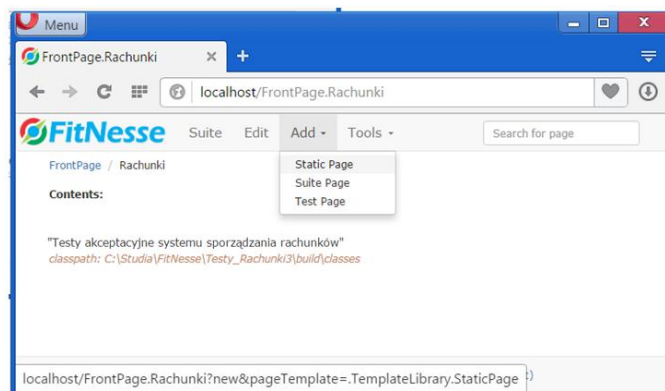


- 3.5. W projekcie utworzonym w p.3.1-3.3, w utworzonym pakiecie **testy_fitness_fixture** należy utworzyć klasę o nazwie **SetUp**, która będzie inicjowała obiekty testowane w testach akceptacyjnych. Poniżej podano kod klasy **SetUp** z przykładu:

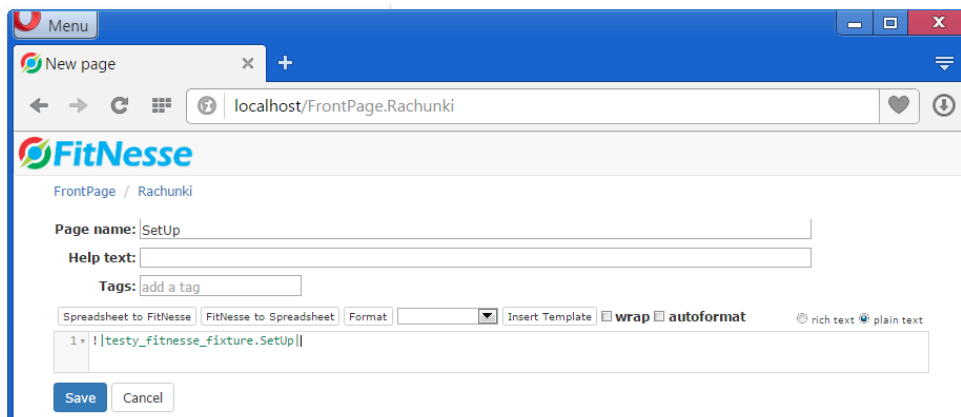
```
package testy_fitness_fixture;

import fit.Fixture;
import rachunek1.TAplikacja;
public class SetUp extends Fixture{
    static TAplikacja aplikacja;
    public SetUp() {
        aplikacja = new TAplikacja();
    }
}
```

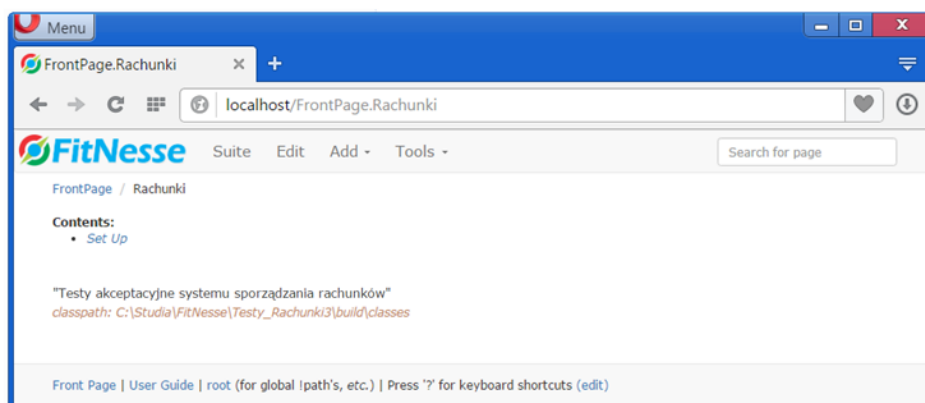
Należy wykonać stronę o nazwie **SetUp** typu **Static Page** - po przejściu na stronę główną **Rachunki** należy w **Menu Bar** wybrać pozycję **Add** i następnie pozycję **Static Page** (rysunek poniżej).



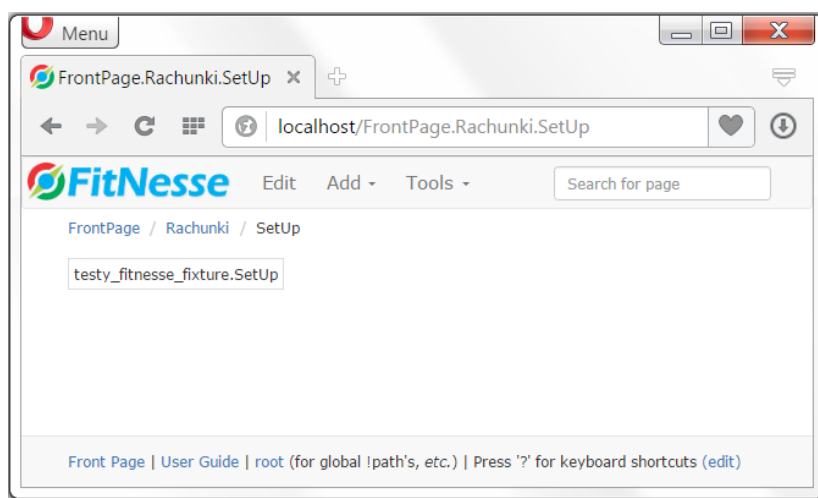
W polu **Page name** należy wpisać nazwę **SetUp** (rysunek poniżej). W zawartości strony należy wpisać ścieżkę pakietową klasy **SetUp**. Strona typu **SetUp** wskazuje na kod wykonanej klasy **SetUp** w pakiecie **testy_fitness_fixture**. Edycję strony należy zatwierdzić za pomocą przycisku **Save**.



Poniżej przedstawiono stronę główną **Rachunki** typu **Suite Page** zawierającą połączenie do strony **SetUp** typu **Static Page**.

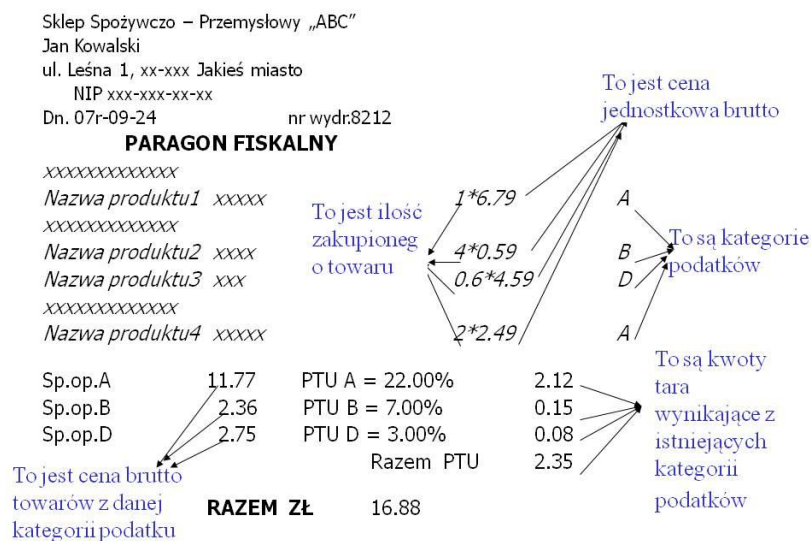


Poniżej przedstawiono widok strony **SetUp**.



- 3.6. Należy przygotować zbiór danych wzorcowych do wykonania testów akceptacyjnych z wykorzystaniem narzędzia **FitNesse**. Tabela, przedstawiona dalej, prezentuje dane wzorcowe do testowania funkcji tworzących obiekty z rodziny typu **TProdukt1**, typu **TZakup** oraz **TRachunek** za pośrednictwem klasy **TAplikacja** w procesie tworzenia rachunków. Poniżej przedstawiono przykład rachunku.

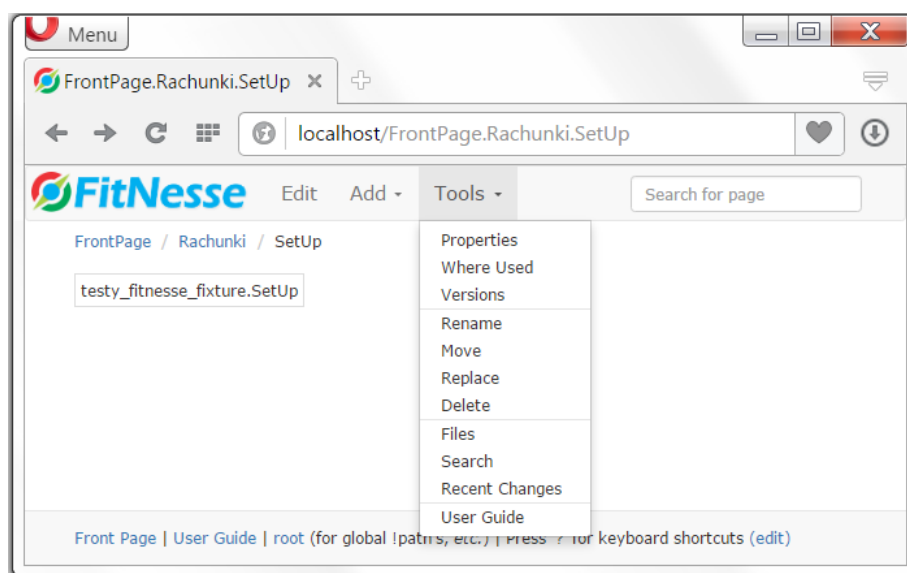
(1 cd) Obliczanie wartości rachunku



Paragon fiskalny reprezentujący wynik procesu biznesowego

Numer rachunku: 1								
Wartość brutto produktu	Ilość produkt w zakupie	Wartość brutto zakupu	Dane produktów w zakupie	Dane szczegółowe produktu w zakupie				
				Sposób tworzenia produktu	Nazwa	Cena netto produktu	Podatek %	Promocja %
1F	1/2	1F/2F	0,1,1,0,0	0	1	1	-	-
2F	2	4F	0,2,2,0,0	0	2	2	-	-
3.42F	1	3.42F	2,3,3,14,0	2	3	3	14	-
4.88F	4	19.52F	2,4,4,22,0	2	4	4	22	-
0.7F	1	0.7F	1,5,1,30	1	5	1	-	30
-	-	-	4,1,1,0,0	?	?	?	?	?
Ceny wynikające z kategorii podatkowych								
Brak podatku	Podatek: 3%	Podatek: 7%	Podatek: 14%	Podatek: 22%	Wszystkie kategorie			
6.7F	0F	0F	3.42F	19.52F	29.640001F			
Numer rachunku: 2								
Wartość brutto produktu	Ilość produkt w zakupie	Wartość brutto zakupu	Dane produktów w zakupie	Dane szczegółowe produktu w zakupie				
				Sposób tworzenia produktu	Nazwa	Cena netto produktu	Podatek %	Promocja %
0.9F	1/2	1.8F	1,6,2,50,0	1	6	2	-	50
3.99F	3	11.97F	3,7,3,3,30	3	7	3	3	30
6.48F	2	12.96F	3,8,4,7,50	3	8	4	7	50
2F	4	8F	0,2,2,0,0	0	2	2	-	-
4.88F	1	4.88F	2,4,4,22,0	2	4	4	22	-
Dane niepoprawne			4,1,1,0,0	?	?	?	?	?
Ceny wynikające z kategorii podatkowych								
Brak podatku	Podatek: 3%	Podatek: 7%	Podatek: 14%	Podatek: 22%	Wszystkie kategorie			
9.8F	11.97F	12.96F	0F	4.88F	39.61F			

3.7. Testy akceptacyjne klasy **TAplikacja** opierają się na wywołaniu głównych testowanych metod **Dodaj_produkt**, **Wstaw_rachunek**, **Szukaj_rachunek**, **Wstaw_zakup**, **Podaj_wartosc** oraz pomocniczych metod w metodach klas testujących, dziedziczących po klasie **ColumnFixture**. Opis tworzenia testów akceptacyjnych w środowisku uruchomionego narzędzia FitNesse w 1.2 lub 1.3 jest dostępny z **MenuBar/Tools/User Guide** (rysunek poniżej), czyli: <http://localhost/FitNesse.UserGuide.WritingAcceptanceTests>.



Przed wywołaniem każdej metody testującej lub grup metod testujących tworzony jest obiekt typu **SetUp**, który tworzy obiekt typu **TAplikacja**, oparty na koncepcji klasy typu **Fasada** warstwy biznesowej testowanej aplikacji. Dalej przedstawiono definicję klas testujących poszczególne funkcje oprogramowania oraz stron uruchamiających te testy.

- 3.8. Jako pierwszy, należy dodać test akceptacyjny dodawania produktu, czyli metody **Dodaj_produkt** klasy **TAplikacja**. W projekcie utworzonym w p.3.1-3.3 należy dodać nową klasę **Test_dodawanie_produktu** do pakietu **testy_fitness_fixture**. Test ten sprawdza liczbę utworzonych obiektów z rodziny **TProdukt1**, sprawdzając zachowanie spójności danych za pomocą metody **liczba_produktow** oraz kontrolę poprawności danych przekazanych do klasy **TFabryka**, przechwytyjąc wyjątek generowany przez klasę **TFabryka** w przypadku niepoprawnych danych. **Kolorem czerwonym zaznaczono nazwy metod i atrybutu, zastosowane dalej przy budowie tablicy decyzyjnej testu na stronie Dodawanie_produktu.**

```
package testy_fitness_fixture;

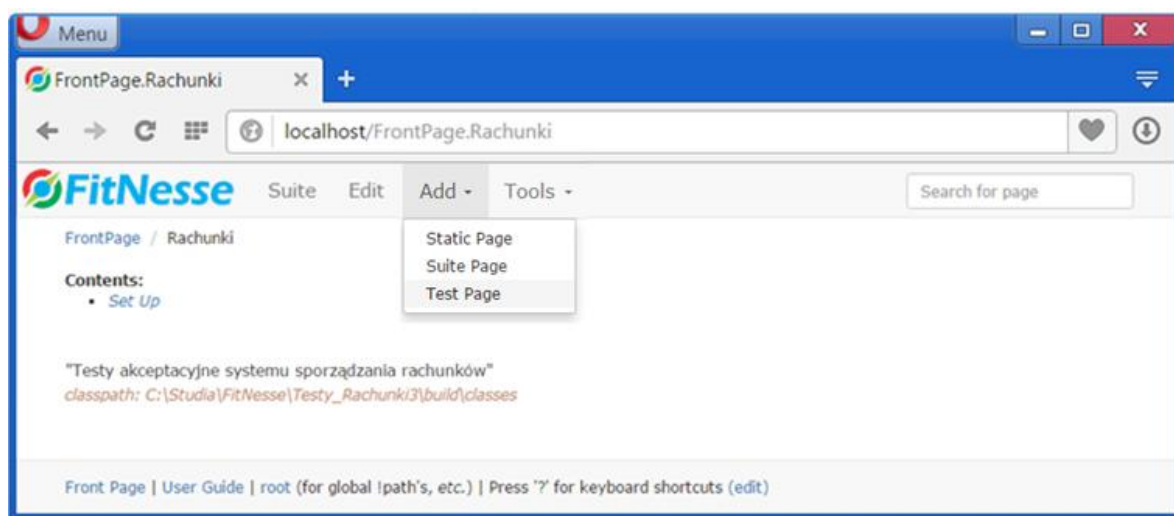
import fit.ColumnFixture;
import java.util.IllegalFormatException;

public class Test_dodawanie_produktu extends ColumnFixture{
    String dane[];

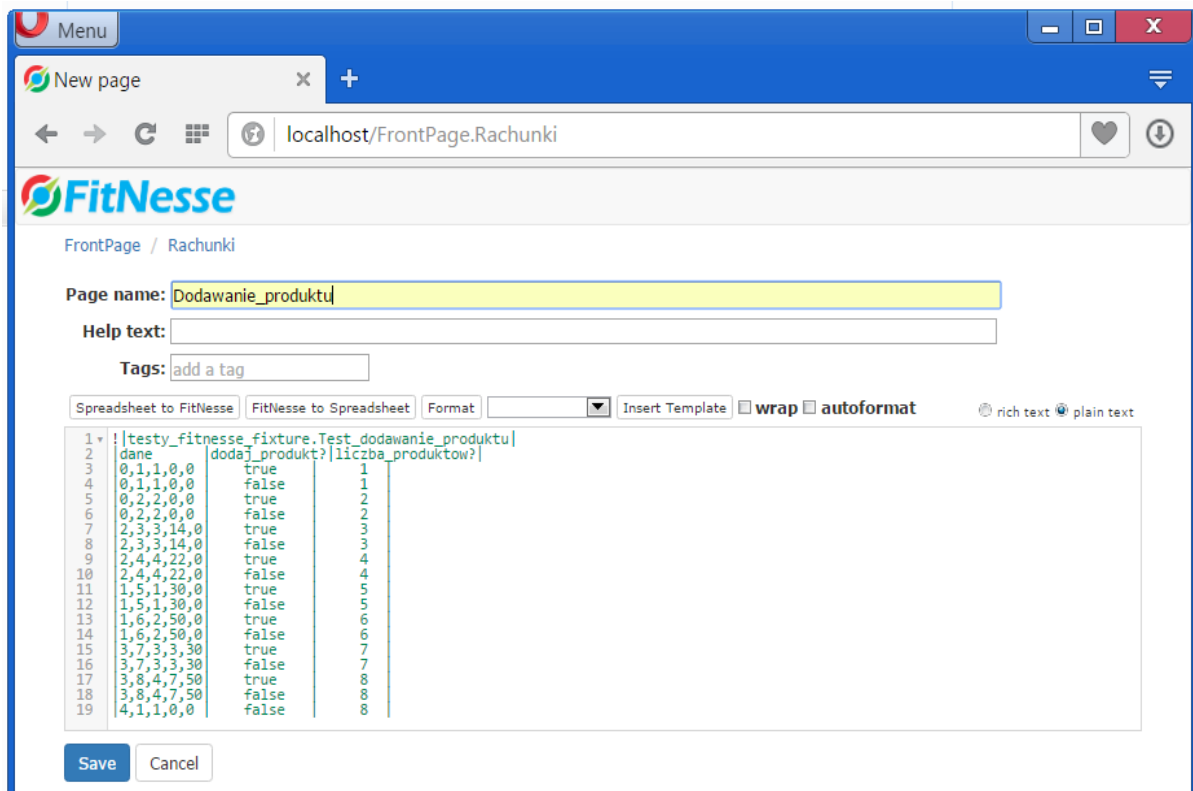
    public boolean dodaj_produkt() throws IllegalFormatException {
        int s1=liczba_produktow();
        try{
            SetUp.aplikacja.Dodaj_produkt(dane);
            int s2=liczba_produktow();
            return s1!=s2;
        } catch(IllegalFormatException e) {
        }
        return false;
    }

    public int liczba_produktow() {
        return SetUp.aplikacja.getProdukty().size();
    }
}
```

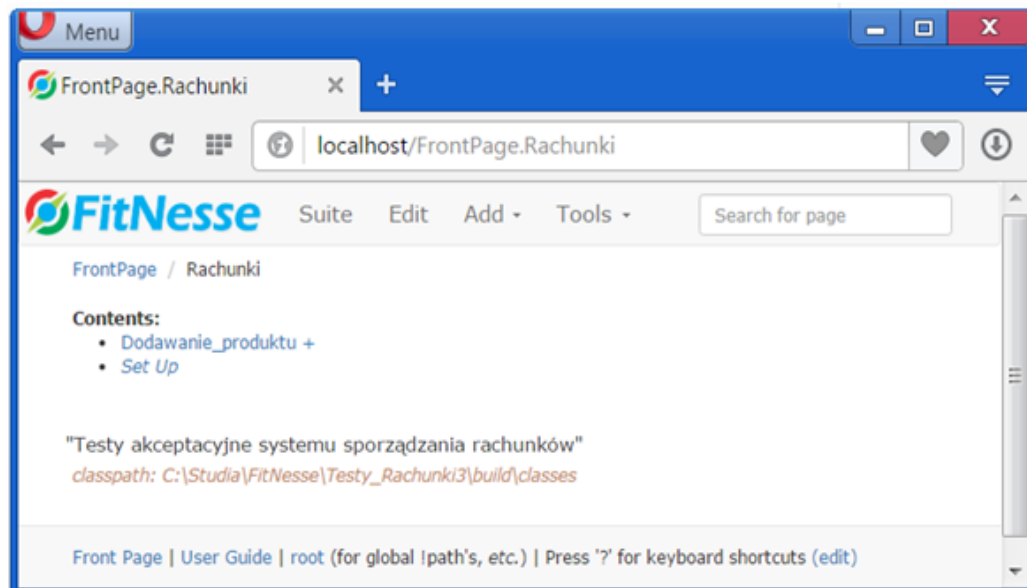
Należy dodać stronę **Dodawanie_produktu** z testem akceptacyjnym dodawania produktu przez wybór na stronie **Rachunki** z listy **Menu Bar/Add** pozycji **Test Page** (rysunek poniżej). Strona ta umożliwi uruchomienie testu akceptacyjnego realizowanego przez klasę **Test_dodawanie_produktu**, dodaną do pakietu **testy_fitness_fixture**. Wykonanie strony **Dodawanie_produktu** uruchamiającej test akceptacyjny realizowany przez klasę **Test_dodawanie_produktu** pokazano na kolejnych rysunkach p.3.8. Testy wykonano w oparciu o dane wzorcowe z tabeli z p.3.6.



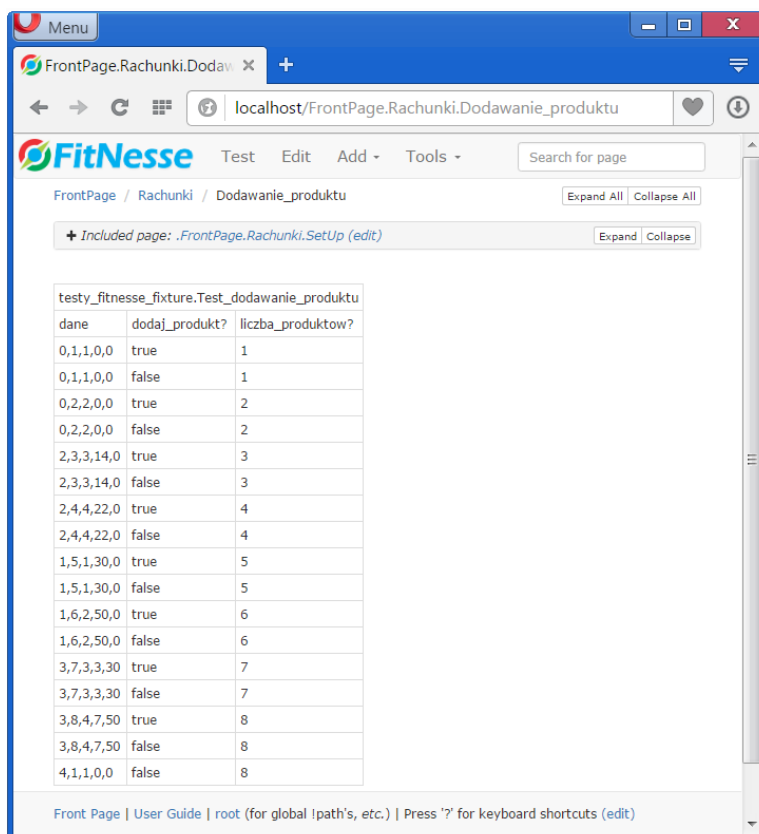
Utworzenie tabeli testującej wyniki zwracane przez metody **dodaj_produkt** oraz **liczba_produktow** po przekazaniu danych w tablicy **dane** przedstawiono na rysunku poniżej – kolumna dane zawiera **dane** z tabeli danych wzorcowych z p. 3.6.



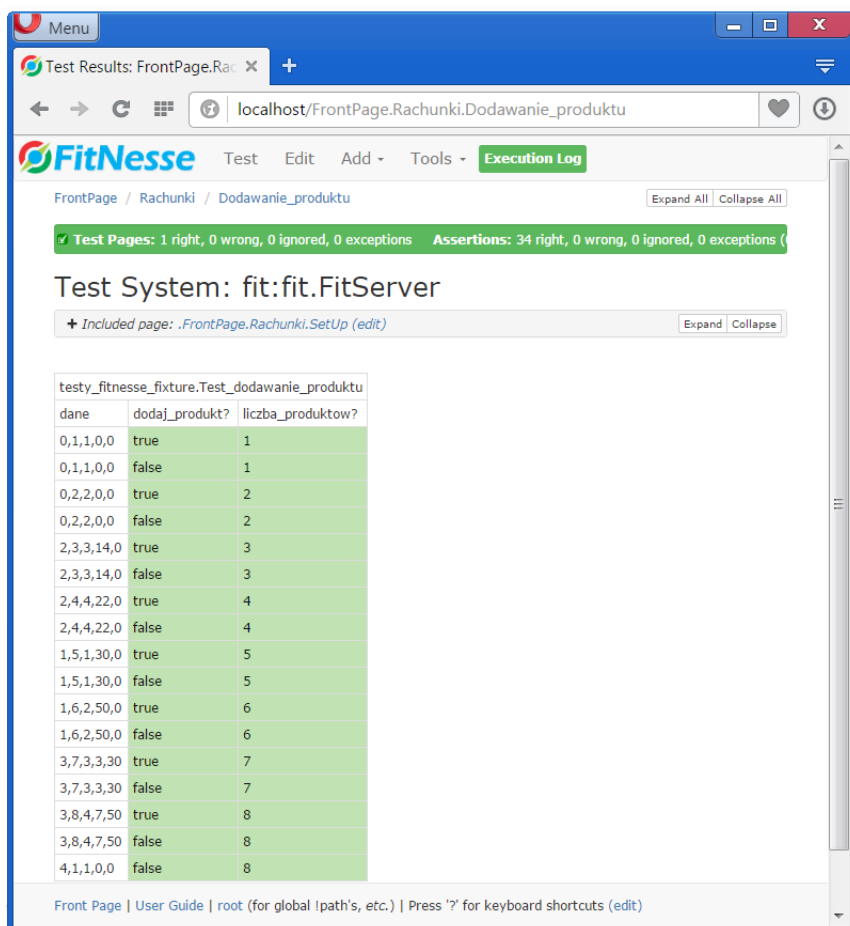
Dalej przedstawiono widok strony głównej **Rachunki** po zatwierdzeniu zawartości strony **Dodawanie_produkту** po naciśnięciu przycisku **Save**.



Na kolejnym rysunku pokazano widok zawartości wybranej strony testowej **Dodawanie_produkту** z listy połączeń strony głównej **Rachunki**.



Widok strony testowej po uruchomieniu testu dodawania produktu za pomocą pozycji **Test** z **Menu Bar** strony **Dodawanie produktu** przedstawiono poniżej.



- 3.9. Wykonanie testu akceptacyjnego dodawania rachunku przez testowaną aplikację – definicja klasy **Test_dodawanie_rachunku** zawierającej kod do testowania tej funkcji, czyli metody **Wstaw_rachunek** klasy **TAplikacja**. Kolorem czerwonym zaznaczono nazwy metod i atrybutu, zastosowane przy budowie tablicy decyzyjnej testu na stronie **Dodawanie_rachunku**.

```
package testy_fitnessse_fixture;

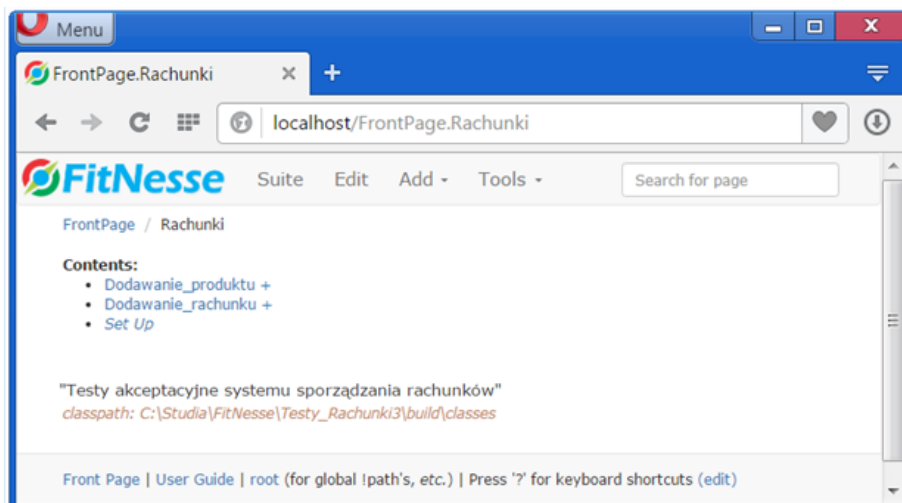
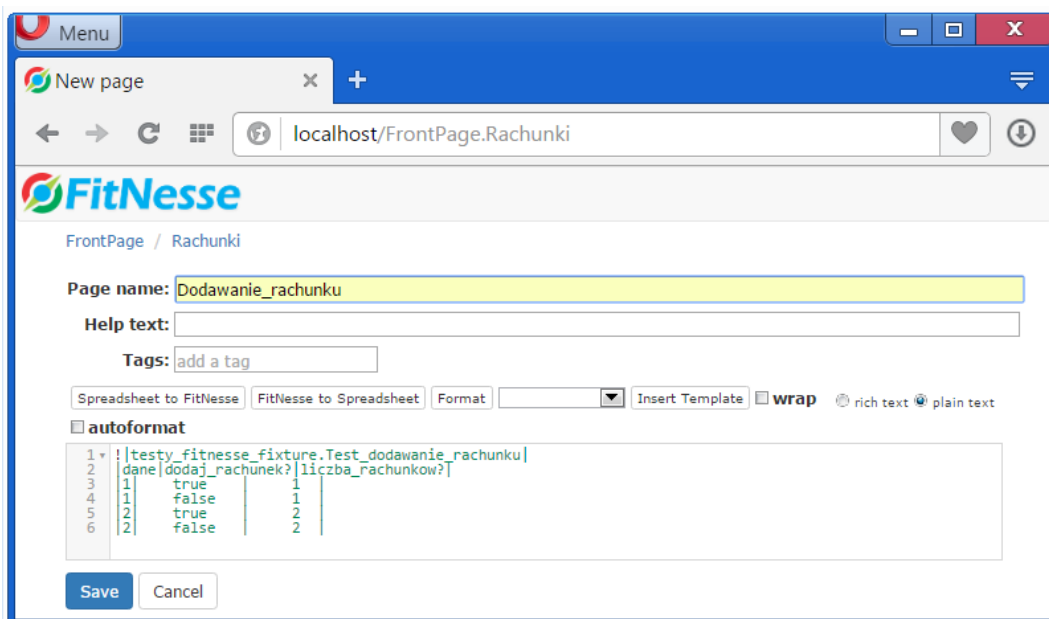
import fit.ColumnFixture;
public class Test_dodawanie_rachunku extends ColumnFixture {

    int dane;

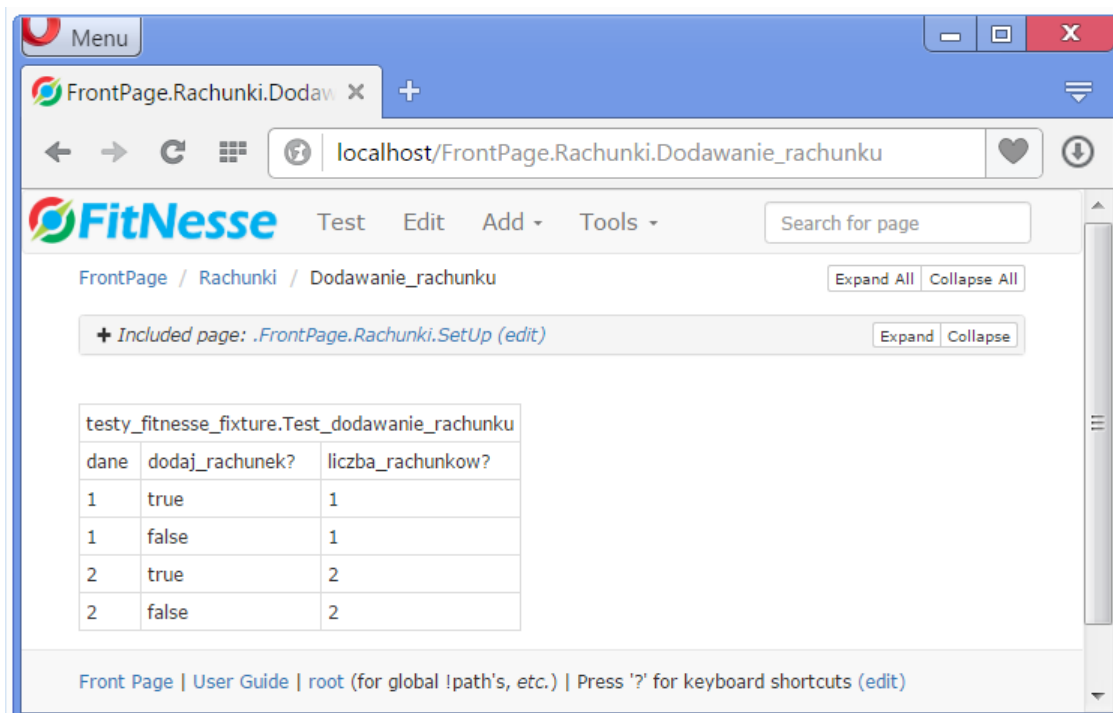
    public boolean dodaj_rachunek() {
        int s1 = liczba_rachunkow();
        SetUp.aplikacja.Wstaw_rachunek(dane);
        int s2 = liczba_rachunkow();
        return (SetUp.aplikacja.Szukaj_rachunek(dane)) != null && s1!=s2;
    }

    public int liczba_rachunkow() {
        return SetUp.aplikacja.getRachunki().size(); }
}
```

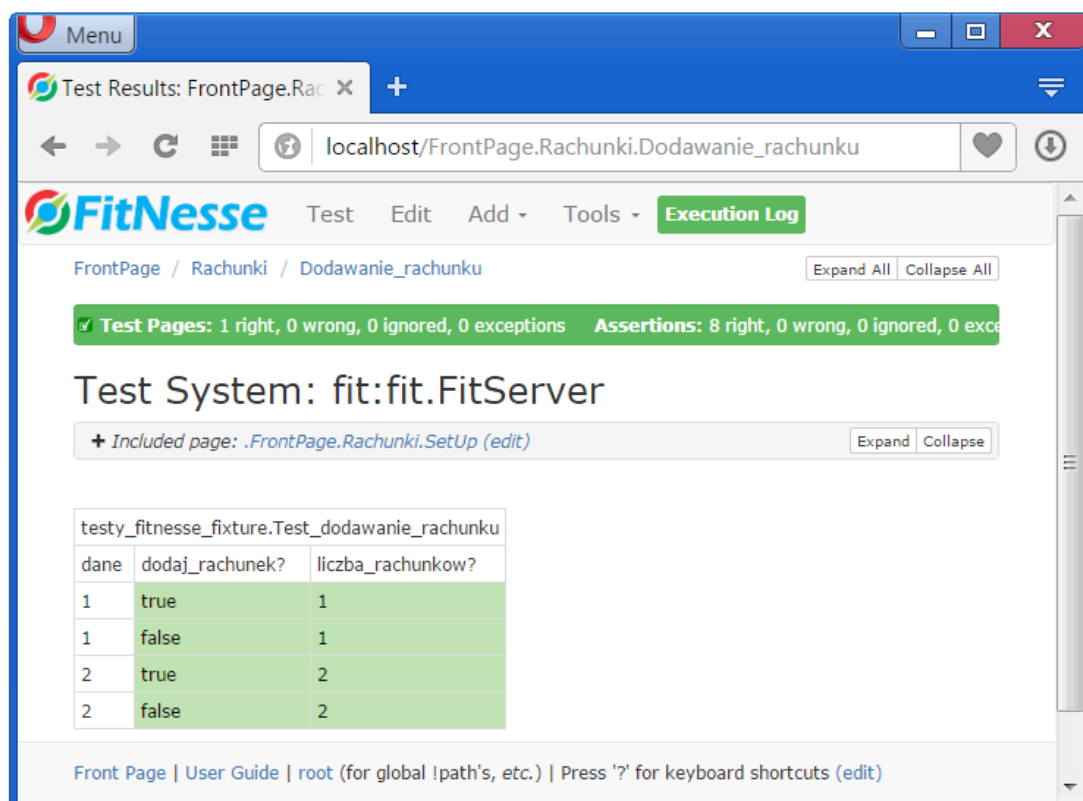
Definicja strony **Dodawanie_rachunku** do testowania dodawania nowego rachunku jest przedstawiona na kolejnych rysunkach p. 3.9 (czynności takie same, jak przy tworzeniu testu dodawania produktu w p.3.8).



Na kolejnym rysunku pokazano widok zawartości wybranej strony testowej **Dodawanie_rachunku** z listy połączeń strony głównej **Rachunki** (poprzedni rysunek).



Widok strony testowej po uruchomieniu testu dodawania nowego rachunku za pomocą pozycji **Test z Menu Bar** strony **Dodawanie_rachunku** pokazano na rysunku poniżej.



- 3.10. Wykonanie testu akceptacyjnego dodawania zakupu przez testowaną aplikację – dalej podano definicję klasy **Test_dodawanie_zakupu** zawierającej kod do testowania funkcji **Wstaw_zakup** klasy **TAplikacja**. Kolorem czerwonym zaznaczono nazwy metod i atrybutu, zastosowane przy budowie tablicy decyzyjnej testu na stronie **Dodawanie_zakupu**.

```
package testy_fitnessse_fixture;

import fit.ColumnFixture;
import java.util.IllegalFormatCodePointException;
import rachunek1.TRachunek;
import rachunek1.TZakup;

public class Test_dodawanie_zakupu extends ColumnFixture {

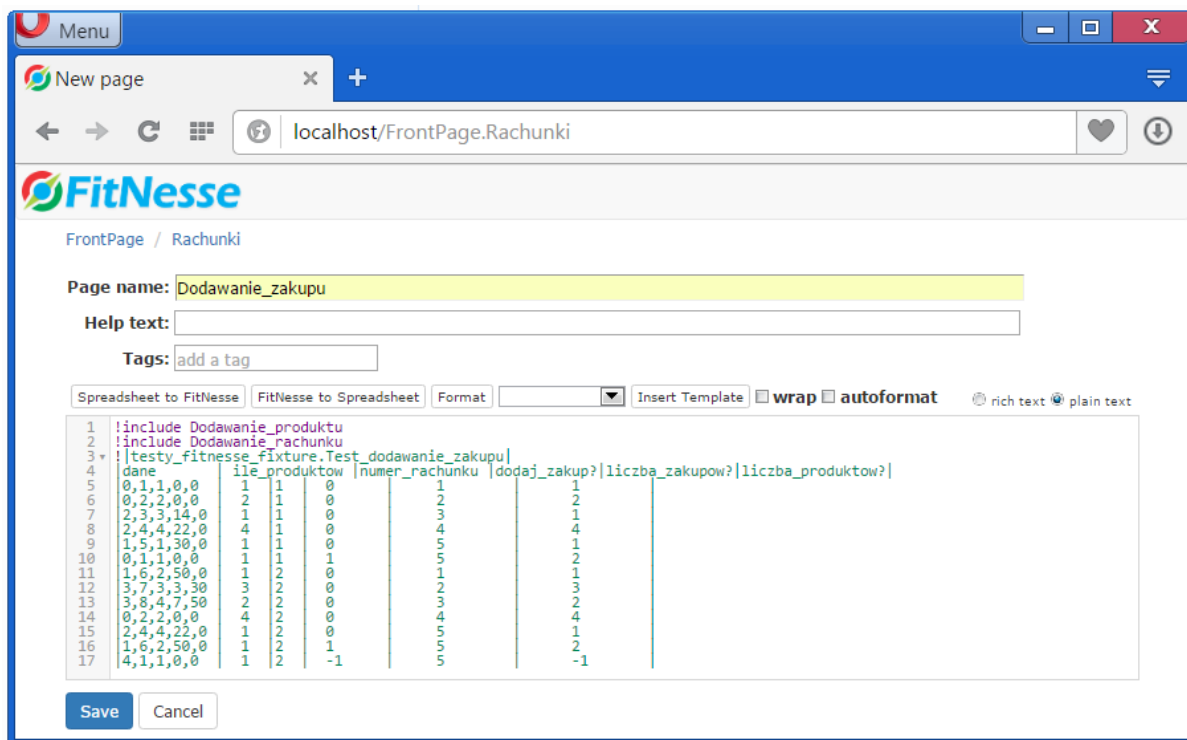
    String dane[];
    int ile_produktow, numer_rachunku, wynik;

    public int dodaj_zakup() {
        int s1 = liczba_zakupow();
        try {
            SetUp.aplikacja.Wstaw_zakup(numer_rachunku, ile_produktow, dane);
            int s2 = liczba_zakupow();
            if (s1 != s2)
                return wynik = 0;
            else
                return wynik = 1;
        } catch (IllegalFormatCodePointException e) {
        }
        return wynik = -1;
    }

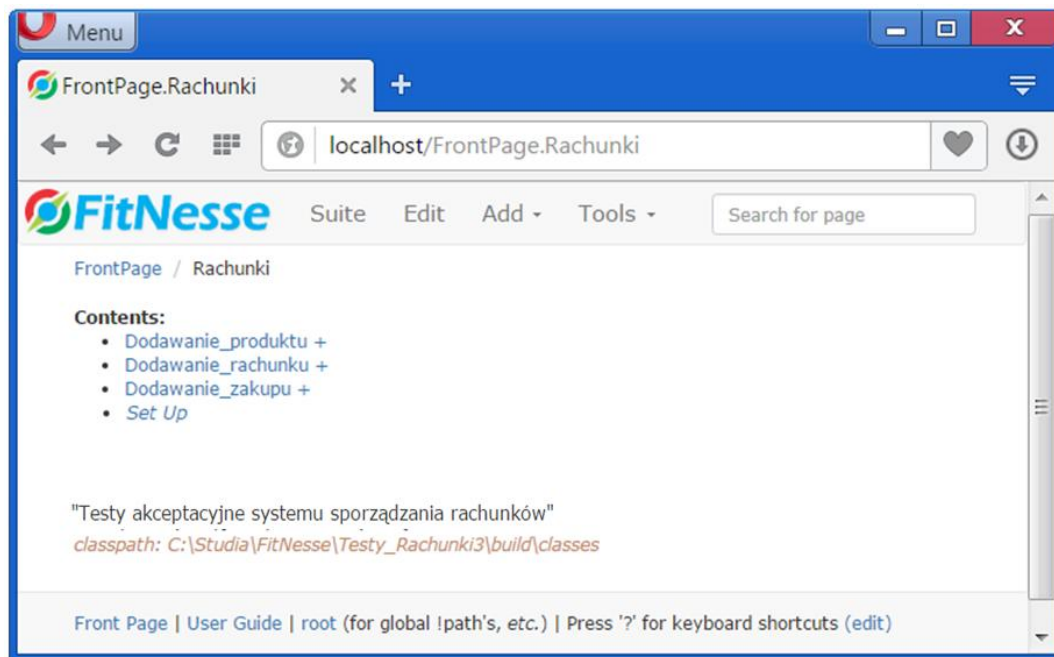
    public int liczba_produktow() {
        TRachunek rachunek = SetUp.aplikacja.getRachunki().get(numer_rachunku - 1);
        int s = rachunek.getZakupy().size();
        TZakup zakup;
        if (wynik == 0)
            zakup = rachunek.getZakupy().get(s - 1);
        else
            if (wynik == 1)
                zakup = rachunek.getZakupy().get(0);
            else return wynik = -1;
        return zakup.Podaj_ilosc();
    }

    public int liczba_zakupow() {
        return SetUp.aplikacja.getRachunki().get(numer_rachunku - 1).getZakupy().size();
    }
}
```

Definicja strony **Dodawanie_zakupu** do testowania dodawania nowego zakupu (rysunek poniżej) opiera się na czynnościach takich samych, jak przy tworzeniu testu dodawania produktu w p.3.8. W treści strony należy uruchomić dwa poprzednie testy dotyczące dodawania produktów i rachunku, aby na podstawie wprowadzonych danych podczas testowania wprowadzania produktów i rachunków wykonać testy akceptacyjne procesu dodawania zakupów tych produktów. Dokonano tego dodając znaczniki **!include** i podając nazwy stron testowych: **Dodawanie_produkty** oraz **Dodawanie_rachunku**. Dane wzorcowe testu pobrano z tabeli z p.3.6.



Poniżej podano widok strony głównej **Rachunki** po dodaniu nowej strony testowej **Dodawanie_zakupu**.



Poniżej pokazano widok strony testowej **Dodawanie_zakupu** po uruchomieniu ze strony **Rachunki**.

FrontPage.Rachunki.Dodawanie_zakupu

FitNesse Test Edit Add - Tools - Search for page

FrontPage / Rachunki / Dodawanie_zakupu

+ Included page: FrontPage.Rachunki.SetUp (edit) Expand Collapse

- Included page: Dodawanie_produkty (edit) Expand Collapse

dane	dodaj_produkty?	liczba_produkty?
0,1,1,0,0	true	1
0,1,1,0,0	false	1
0,2,2,0,0	true	2
0,2,2,0,0	false	2
2,3,3,14,0	true	3
2,3,3,14,0	false	3
2,4,4,22,0	true	4
2,4,4,22,0	false	4
1,5,1,30,0	true	5
1,5,1,30,0	false	5
1,6,2,50,0	true	6
1,6,2,50,0	false	6
3,7,3,3,30	true	7
3,7,3,3,30	false	7
3,8,4,7,50	true	8
3,8,4,7,50	false	8
4,1,1,0,0	false	8

- Included page: Dodawanie_rachunku (edit) Expand Collapse

dane	dodaj_rachunek?	liczba_rachunkow?
1	true	1
1	false	1
2	true	2
2	false	2

dane	ile_produkty	numer_rachunku	dodaj_zakup?	liczba_zakupow?	liczba_produkty?
0,1,1,0,0	1	1	0	1	1
0,2,2,0,0	2	1	0	2	2
2,3,3,14,0	1	1	0	3	1
2,4,4,22,0	4	1	0	4	4
1,5,1,30,0	1	1	0	5	1
0,1,1,0,0	1	1	1	5	2
1,6,2,50,0	1	2	0	1	1
3,7,3,3,30	3	2	0	2	3
3,8,4,7,50	2	2	0	3	2
0,2,2,0,0	4	2	0	4	4
2,4,4,22,0	1	2	0	5	1
1,6,2,50,0	1	2	1	5	2
4,1,1,0,0	1	2	-1	5	-1

Front Page | User Guide | root (for global !path's, etc.) | Press '?' for keyboard shortcuts (edit)

Wykonanie testu dodawania nowego zakupu po uruchomieniu testu za pomocą pozycji **Test z Menu Bar** strony **Dodawanie_zakupu** pokazano na rysunku poniżej.

The screenshot shows the FitNesse Test Results interface for the test 'FrontPage.Rachunki.Dodawanie_zakupu'. The interface includes a menu bar, a browser address bar showing 'localhost/FrontPage.Rachunki.Dodawanie_zakupu', and a status bar indicating 'Test Pages: 1 right, 0 wrong, 0 ignored, 0 exceptions' and 'Assertions: 81 right, 0 wrong, 0 ignored, 0 exceptions (0,508 seconds)'.

The test system is identified as 'fit:fit.FitServer'. The included page is 'FrontPage.Rachunki.SetUp (edit)'. The test results are displayed in a table format, showing the test name, the data used, and the results of the assertions.

Test Results: FrontPage.Rachunki.Dodawanie_zakupu

Test Pages: 1 right, 0 wrong, 0 ignored, 0 exceptions Assertions: 81 right, 0 wrong, 0 ignored, 0 exceptions (0,508 seconds)

Test System: fit:fit.FitServer

Included page: FrontPage.Rachunki.SetUp (edit)

Included page: Dodawanie_produkty (edit)

dane	dodaj_produkty?	liczba_produkty?
0,1,1,0,0	true	1
0,1,1,0,0	false	1
0,2,2,0,0	true	2
0,2,2,0,0	false	2
2,3,3,14,0	true	3
2,3,3,14,0	false	3
2,4,4,22,0	true	4
2,4,4,22,0	false	4
1,5,1,30,0	true	5
1,5,1,30,0	false	5
1,6,2,50,0	true	6
1,6,2,50,0	false	6
3,7,3,3,30	true	7
3,7,3,3,30	false	7
3,8,4,7,50	true	8
3,8,4,7,50	false	8
4,1,1,0,0	false	8

Included page: Dodawanie_rachunku (edit)

dane	dodaj_rachunek?	liczba_rachunkow?
1	true	1
1	false	1
2	true	2
2	false	2

testy_fitnesse_fixture.Test_dodawanie_zakupu

dane	ile_produkty	numer_rachunku	dodaj_zakup?	liczba_zakupow?	liczba_produkty?
0,1,1,0,0	1	1	0	1	1
0,2,2,0,0	2	1	0	2	2
2,3,3,14,0	1	1	0	3	1
2,4,4,22,0	4	1	0	4	4
1,5,1,30,0	1	1	0	5	1
0,1,1,0,0	1	1	1	5	2
1,6,2,50,0	1	2	0	1	1
3,7,3,3,30	3	2	0	2	3
3,8,4,7,50	2	2	0	3	2
0,2,2,0,0	4	2	0	4	4
2,4,4,22,0	1	2	0	5	1
1,6,2,50,0	1	2	1	5	2
4,1,1,0,0	1	2	-1	5	-1

Front Page | User Guide | root (for global !path's, etc.) | Press '?' for keyboard shortcuts (edit)

3.11. Wykonanie testu akceptacyjnego obliczania wartości zakupu przez testowaną aplikację – definicja klasy **Test_obliczanie_wartosci_rachunku** zawierającej kod do testowania tej funkcji, czyli metody **Podaj_wartosc** klasy **TAplikacja**. Kolorem czerwonym zaznaczono nazwy metod i atrybutu, zastosowane przy budowie tablicy decyzyjnej testu na stronie **Obliczanie_wartosci_rachunku**.

```
package testy_fitnessse_fixture;

import fit.ColumnFixture;

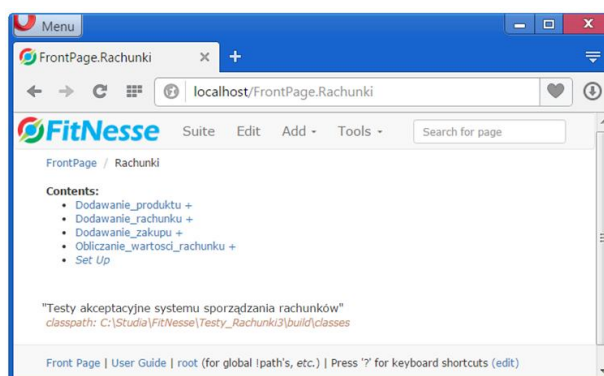
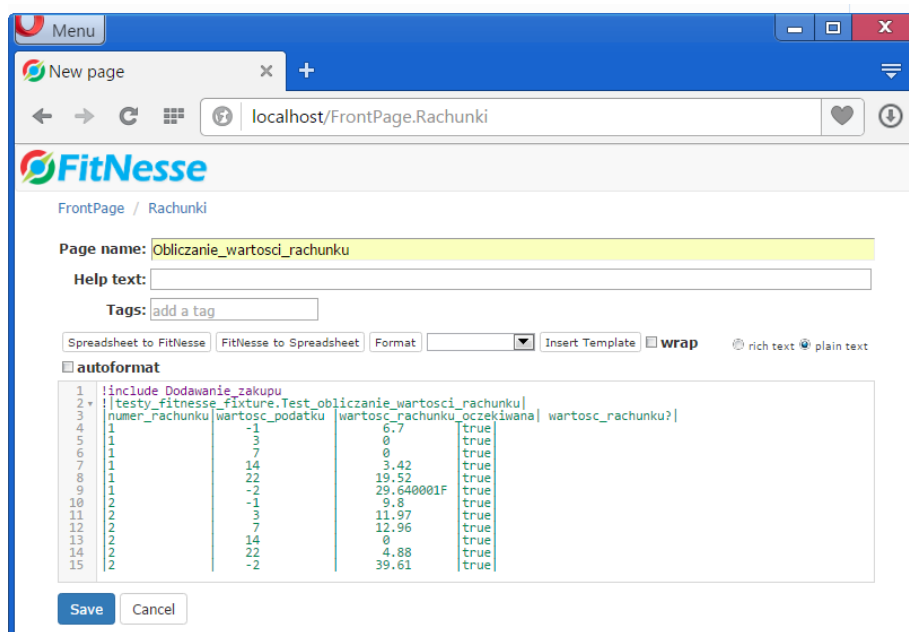
public class Test_obliczanie_wartosci_rachunku extends ColumnFixture{

    int numer_rachunku, wartosc_podatku;
    float wartosc_rachunku_oczekiwana;

    public boolean wartosc_rachunku() {
        return SetUp.aplikacja.Podaj_wartosc(numer_rachunku, wartosc_podatku)==
            wartosc_rachunku_oczekiwana; }

}
```

Definicja strony **Obliczanie_wartosci_rachunku** do testowania obliczania wartości wybranego rachunku wg kategorii ceny (rysunek poniżej) opiera się na czynnościach takich samych, jak przy tworzeniu testu dodawania produktu w p.3.8. W treści strony należy uruchomić trzy poprzednie testy dotyczące dodawania produktów, rachunku oraz zakupów, aby na podstawie wprowadzonych danych podczas testowania wprowadzania produktów, rachunków i zakupów wykonać testy akceptacyjne procesu obliczania wartości rachunków wypełnionych zakupami (wprowadzonych podczas testowania akceptacyjnego procesu dodawania zakupów z p.3.10). Dokonano tego dodając znacznik **!include** i podając nazwy strony testowej: **Dodawanie_zakupu**. Podczas uruchamiania tej strony testowej zostaną uruchomione strony **Dodawanie_produktu**, **Dodawanie_rachunku** (p. 3.10). Dane wzorcowe testu pobrano z tabeli z p.3.6.



Poniżej podano widok strony testowej **Obliczanie_wartosci_rachunku** uruchomionej ze strony **Rachunki** (poprzedni rysunek).

The screenshot shows a web application interface with a blue header bar containing a 'Menu' button and window controls. The browser address bar shows 'localhost/FrontPage.Rachunki.Obliczanie_wartosci_rachunku'. The application has a 'FitNesse' logo and a search bar. The main content area displays a tree view of test results, with the following tables:

Included page: Dodawanie_produkty (edit)

dane	dodaj_produkty?	liczba_produkty?
0,1,1,0,0	true	1
0,1,1,0,0	false	1
0,2,2,0,0	true	2
0,2,2,0,0	false	2
2,3,3,14,0	true	3
2,3,3,14,0	false	3
2,4,4,22,0	true	4
2,4,4,22,0	false	4
1,5,1,30,0	true	5
1,5,1,30,0	false	5
1,6,2,50,0	true	6
1,6,2,50,0	false	6
3,7,3,3,30	true	7
3,7,3,3,30	false	7
3,8,4,7,50	true	8
3,8,4,7,50	false	8
4,1,1,0,0	false	8

Included page: Dodawanie_rachunku (edit)

dane	dodaj_rachunek?	liczba_rachunkow?
1	true	1
1	false	1
2	true	2
2	false	2

testy_fitnesse_fixture.Test_dodawanie_zakupu

dane	ile_produkty	numer_rachunku	dodaj_zakup?	liczba_zakupow?	liczba_produkty?
0,1,1,0,0	1	1	0	1	1
0,2,2,0,0	2	1	0	2	2
2,3,3,14,0	1	1	0	3	1
2,4,4,22,0	4	1	0	4	4
1,5,1,30,0	1	1	0	5	1
0,1,1,0,0	1	1	1	5	2
1,6,2,50,0	1	2	0	1	1
3,7,3,3,30	3	2	0	2	3
3,8,4,7,50	2	2	0	3	2
0,2,2,0,0	4	2	0	4	4
2,4,4,22,0	1	2	0	5	1
1,6,2,50,0	1	2	1	5	2
4,1,1,0,0	1	2	-1	5	-1

testy_fitnesse_fixture.Test_obliczanie_wartosci_rachunku

numer_rachunku	wartosc_podadu	wartosc_rachunku_oczekiwana	wartosc_rachunku?
1	-1	6.7	true
1	3	0	true
1	7	0	true
1	14	3.42	true
1	22	19.52	true
1	-2	29.640001F	true
2	-1	9.8	true
2	3	11.97	true
2	7	12.96	true
2	14	0	true
2	22	4.88	true
2	-2	39.61	true

Front Page | User Guide | root (for global lpath's, etc.) | Press '?' for keyboard shortcuts (edit)

Test Results: FrontPage.Rachunki

localhost/FrontPage.Rachunki.Obliczanie_wartosci_rachunku

FitNesse Test Edit Add - Tools - Execution Log

FrontPage / Rachunki / Obliczanie_wartosci_rachunku

Test Pages: 1 right, 0 wrong, 0 ignored, 0 exceptions Assertions: 93 right, 0 wrong, 0 ignored, 0 exceptions (0.792 seconds)

Test System: fit:fit.FitServer

Included page: FrontPage.Rachunki.SetUp (edit)

Included page: Dodawanie_zakupu (edit)

Included page: Dodawanie_produktu (edit)

testy_fitnesse_fixture.Test_dodawanie_produktu			
dane	dodaj_produkt?	liczba_produktow?	
0,1,1,0,0	true	1	
0,1,1,0,0	false	1	
0,2,2,0,0	true	2	
0,2,2,0,0	false	2	
2,3,3,14,0	true	3	
2,3,3,14,0	false	3	
2,4,4,22,0	true	4	
2,4,4,22,0	false	4	
1,5,1,30,0	true	5	
1,5,1,30,0	false	5	
1,6,2,50,0	true	6	
1,6,2,50,0	false	6	
3,7,3,3,30	true	7	
3,7,3,3,30	false	7	
3,8,4,7,50	true	8	
3,8,4,7,50	false	8	
4,1,1,0,0	false	8	

Included page: Dodawanie_rachunku (edit)

testy_fitnesse_fixture.Test_dodawanie_rachunku			
dane	dodaj_rachunek?	liczba_rachunkow?	
1	true	1	
1	false	1	
2	true	2	
2	false	2	

testy_fitnesse_fixture.Test_dodawanie_zakupu					
dane	ile_produktow	numer_rachunku	dodaj_zakup?	liczba_zakupow?	liczba_produktow?
0,1,1,0,0	1	1	0	1	1
0,2,2,0,0	2	1	0	2	2
2,3,3,14,0	1	1	0	3	1
2,4,4,22,0	4	1	0	4	4
1,5,1,30,0	1	1	0	5	1
0,1,1,0,0	1	1	1	5	2
1,6,2,50,0	1	2	0	1	1
3,7,3,3,30	3	2	0	2	3
3,8,4,7,50	2	2	0	3	2
0,2,2,0,0	4	2	0	4	4
2,4,4,22,0	1	2	0	5	1
1,6,2,50,0	1	2	1	5	2
4,1,1,0,0	1	2	-1	5	-1

testy_fitnesse_fixture.Test_obliczanie_wartosci_rachunku				
numer_rachunku	wartosc_podatku	wartosc_rachunku_oczekiwana	wartosc_rachunku?	
1	-1	6.7	true	
1	3	0	true	
1	7	0	true	
1	14	3.42	true	
1	22	19.52	true	
1	-2	29.640001F	true	
2	-1	9.8	true	
2	3	11.97	true	
2	7	12.96	true	
2	14	0	true	
2	22	4.88	true	
2	-2	39.61	true	

Front Page | User Guide | root (for global lpath's, etc.) | Press '?' for keyboard shortcuts (edit)

Wykonanie testu wyznaczania wartości rachunku pokazano na rysunku powyżej.