Laboratorium 5

Wstęp

Pisanie testów bazuje na wykorzystaniu poleceń assert (tzw assercji), które pozwalają sprawdzić czy spełnione są określone wymagania np.

- assertEquals(Obj1,Obj2) – sprawdza czy dwa obiekty są równe
- assertNull(Obj1) – sprawdza czy podana wartość jest typu null
- assertNotNull(Obj1) – sprawdza czy podana wartość nie jest null'em
- assertTrue(Boolean) – sprawdza czy podana wartość typu boolean ma wartość true
- assertFalse(Boolean) – sprawdza czy podana wartość typu boolean ma wartość false
- assertSame(Obj1,Obj2) – sprawdza czy podane referencje odwołują się do tego samego obiektu
- assertNot Same(Obj1,Obj2)
- assertArrayEquals(Obj1[],Obj2[]) – spradza równość obydwu tablic

W JUnit sterowanie wykonanie testów odbywa się poprzez definiowanie odpowiednich anotacji. Podstawowe Anotacje dot. metod to:

@BeforeClass – oznacza metodę uruchamianą raz na samym początku przed wykonaniem testu

@Before – oznacza metodę każdorazowo wykonywaną przed każdym z testów

@Test – oznacza metodę implementującą właściwy test

@After – oznacza metodę wykonywaną każdorazowo po każdym z testów

@AfterClass – oznacza metodę wykonywaną na zakończenie, po wykonaniu wszystkich testów

Przeanalizuj poniższy przykład:

```java
import org.junit.*;
import static org.junit.Assert.*;
import java.util.*;

public class JunitTest1 {

    private Collection collection;

    @BeforeClass
    public static void oneTimeSetUp() {
        System.out.println("@BeforeClass - Jednorazowe uruchamienie przy starcie");
    }

    @AfterClass
    public static void oneTimeTearDown() {
        System.out.println("@AfterClass - Jednorazowe uruchomienie przy końcu");
    }

    @Before
    public void setUp() {
        collection = new ArrayList();
        System.out.println("@Before - Każdorazowo uruchamiane przed testem");
    }

    @After
    public void tearDown() {
        collection.clear();
        System.out.println("@After - Każdorazowo uruchamiane po teście");
    }

    @Test
    public void testEmptyCollection() {
        assertTrue(collection.isEmpty());
        System.out.println("@Test - Właściwy test 1");
    }

    @Test
    public void testOneItemCollection() {
        collection.add("itemA");
        assertEquals(1, collection.size());
        System.out.println("@Test - Właściwy test 2");
    }
}
```

Wynik:

```
@BeforeClass - Jednorazowe uruchamienie przy starcie
@Before - Każdorazowo uruchamiane przed testem
@Test - Właściwy test 1
@After - Każdorazowo uruchamiane po teście
@Before - Każdorazowo uruchamiane przed testem
@Test - Właściwy test 2
@After - Każdorazowo uruchamiane po teście
@AfterClass - Jednorazowe uruchomienie przy końcu
```

Zwróć uwagę że metody z anotacją BeaforeClass i AfterClass są statyczne

Zadanie

Stwórz nowy projekt. W projekcie stwórz pakiet pl.polsl.ip.pu.testowanie. W pakiecie utwórz klasę Osoba implementującą interfejs IOsoba:

```
public interface IOsoba {
    String getImie();
    String getNazwisko();
    String getNrTelefonu();
    String info();
    void setImie(String imie);
    void setNazwisko(String nazwisko);
    void setNrTelefonu(String nrTelefonu);
    void wyswietlInfo();
}
```

Stwórz klasę BazaOsob umożliwiającą przechowywanie listy osób. W realizacji tego zadania nie korzystaj z dostępnych struktur, a jedynie z tablic. Klasa BazaOsób powinna implementować interfejs

```
public interface IBazaOsob {

    void dodaj(IOsoba o);
    int getLicznik();
    IOsoba odczytaj(int i);
    void usun(int i);
    void usun(IOsoba o);
    void wyswietl();

}
```

Gdzie metoda getLicznik powinna zwracać liczbę osób w bazie.

Konstruktor klasy BazaOsob jako parametr powinien przyjmować początkowy rozmiar tablicy. Jeżeli chcemy dodać więcej elementów niż zakładany poziom tablicy wówczas tablica automatycznie powinna zostać przepisana do nowej tablicy o zwiększonym rozmiarze. Domyslnie rozmiar tablicy powinien się zwiększać 1.5 raza.

Jeżeli tablica osiągnie rozmiar większy niż 30 elementów powinien zostać zwrócony wyjątek PrzekroczonoDopuszczalnyRozmiar(). Klasę wyjątku należy stworzyć samodzielnie.

Na podstawie przykładu z początku instrukcji napisz testy jednostkowe dla stworzonych klas w oparciu o JUnit 4.0.

W tym celu możesz wykorzystać wbudowane narzędzia Netbeansa. Kliknij prawym przyciskiem na daną klasę (w zakładce projektu) pod pozycją Tools znajdziesz polecenie CreateTest . Uzyj go do stworzenia testu.

Tworząc testy jednostkowe klas sprawdź ich odporność na wprowadzanie wartości typu null

Ponadto wykonaj testy poprawności zachowania klasy BazaOsob na wypadek dodawania i przekroczenia zakresu rozmiaru. W takim przypadku tablica Osob w klasie BazaOsob powinna automatycznie zwiększyć swój rozmiar 1.5 raza, tak aby mogła przyjąć nowe wartości

Sprawdź poprawność metod usun(), tzn czy poprawnie usuwają wartości itp