Testowanie aplikacji JAVA

Laboratorium (JUnit oraz Hamcrest)

Tutorial do biblioteki **Hamcrest** jest **tutaj**.
Przykład użycia biblioteki **Hamcrest** jest **tutaj**.
Ściąga metod w bibliotece **Hamcrest** jest **tutaj**.
Projekt **introductionHamcrest** jest **tutaj**.
Wiki do testów **parametrycznych** jest **tutaj**.

Zadanie 1. Do kalkulatora z zadania 2 (Labolatorium 1) powtórzyć testy jednostkowe, używając biblioteki **Hamcrest**.

Zadanie 2. Napisać klasę MyStack implementujący stos. Zaimplementować podstawowe operację na stosie:

- MyPush (wkładanie na stos)
- MyPop (zdejmowanie ze stosu)
- IsNull (pusty stos)
- MyTop (wypisanie szczytu stosu)

Przy pomocy biblioteki **Hamcrest** oraz **JUnit** przetestować klasę MyStack.

Zadanie 3. Używając dowolnej metody sortowania napisać klasę Sorting, która będzie zawierała tablicę o nazwie **table** oraz metodę **SortArray** (Sortowanie rosnące tablicy) następnie przetestować klasę Sorting przy użyciu **JUnit** oraz biblioteki **Hamcrest**. Rozważ różne przypadki.

Zadanie 4. Dany jest interfejs (kontrakt), który jest częścią oprogramowania pewnej gry dla dzieci. Interfejs należy zaimplementować, a następnie przetestować przy użyciu **JUnit** oraz biblioteki **Hamcrest**. Interfejs wygląda następująco:

```
public Interface Psikus{
  Integer CyfroKrad(Integer liczba);
  Integer HultajChochla(Integer liczba) throws NieudanyPsikusException;
  Integer Niekształtek (Integer liczba);
}
```

Metody w klasie **Kontrakt** mają działać następująco:

- Metoda **CyfroKrad** działa w ten sposób, że w liczbie przekazanej jako argument usuwa losowo jedną cyfrę. Gdy jednocyfrowa zwraca null.
- Metoda **HultajChochla** działa w ten sposób, że losowo zmienia wystąpienia dwóch cyfr w liczbie będącej argumentem.
- Metoda **Niekształtek** działa w ten sposób, że w zadanej liczbie dokonuje jednej losowej zmiany cyfr według schematu:
  - $3 \rightarrow 8$
  - $7 \rightarrow 1$
  - $6 \rightarrow 9$

  Gdy powyższe liczby nie występują w liczbie będącej argumentem, metoda zwraca argument.

UWAGI:

1. Zastosować zasadę TDD (Test Driver Development).
2. Zastosować **wywołanie parametryczne**, utworzyć **suite** oraz z korzystać z biblioteki **Hamcrest**.
3. Zwrócić uwagę na wyjątki i przypadki brzegowe.