Testowanie aplikacji JAVA

Laboratorium (Maven - Projekt 1)

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

Zadanie 1. Utwórz projekt **MAVEN** z zadania 2 labolatorium 2 (klasa stos). Stos ma być reprezentowany poprzez **Listę**. Dla prostoty wrzucamy **liczby całkowite**. Rozpatrz następujące przypadki:

1. Jaka jest wartość IsEmpty() na początku utworzenia stosu?
2. Metodą push włóż element, następnie metodą top sprawdź wiele razy, czy wyszedł ten sam element.
3. Poprawność działania metody pop.
4. Wrzucenie wartości największej i najmniejszej na stosie.
5. I inne ...

TERMIN: 06.03.2016 godz 23.59
Rozwiązanie ma być w repozytorium! Projekt w repozytorium nie powinien zawierać pliku **jar** oraz folderu **target** (Projekt ma być przygotowany do kompilacji po ściągnięciu).
Spóźnienia z terminem będą wiązały się z mniejszą ilością punktów. Maksymalne spóźnienie wynosi do następnych zajęć (09.03.2016) i wtedy obowiązuje 50% punktów z projektu. A więc dzień zwłoki oznacza obniżenie progu o 15%.
Pod ocenę będą brane:

1. Kompilacja i poprawne działanie kodu.
2. Przetestowanie i rozpatrzenie różnych przypadków.
3. Obsługa wyjątków.
4. Pokrycie kodu.
5. Stylistyka kodu.

Zadanie 2. Dany jest interfejs (kontrakt), który jest częścią oprogramowania pewnej gry dla dzieci na liczbach naturalnych. Interfejs należy zaimplementować jako projekt **MAVEN**, 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);
  Integer Heheszki(Integer liczba);
  bool Titit(Integer liczba_dziel);
}
```

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.
- Metoda **Heheszki** zamienia liczbę na dowolną całkowitą z przedziału <0,liczba).
- Metoda **Titit** sprawdza, czy liczba jest podzielna przez liczba_dziel.

TERMIN: 06.03.2016 godz 23.59
Rozwiązanie ma być w repozytorium! Projekt w repozytorium nie powinien zawierać pliku **jar** oraz folderu **target** (Projekt ma być przygotowany do kompilacji po ściągnięciu).
Spóźnienia z terminem będą wiązały się z mniejszą ilością punktów. Maksymalne spóźnienie wynosi do następnych zajęć (09.03.2016) i wtedy obowiązuje 50% punktów z zadania. A więc dzień zwłoki oznacza obniżenie progu o 15%.
Pod ocenę będą brane:

1. Kompilacja i poprawne działanie kodu.
2. Przetestowanie i rozpatrzenie różnych przypadków.
3. Obsługa wyjątków.
4. Pokrycie kodu.
5. Stylistyka kodu.
6. Zastosowanie biblioteki **Hamcrest**.
7. Zastosowanie **wywołania parametrycznego**.
8. Zastosowanie **suite**.

UWAGI:

1. Oczywiste jest, że klasa będzie zawierała pole z liczbą do obsługi wszystkich metod.
2. Zastosować zasadę TDD (Test Driver Development).
3. Zastanowić się nad wyjątkami i przypadkami brzegowymi.