Otwarto: środa, 16 listopada 2022, 11:20
Wymagane do: środa, 16 listopada 2022, 12:50

---

1. *(1 pkt.)* Do klas `Address` i `Patient` z zeszłego tygodnia proszę dopisać metody `public int length(String fieldName)`, które przyjmują łańcuch, będący nazwą pola klasy. W zależności od tego łańcucha metoda ma zwrócić "długość" odpowiedniego pola, którą definiujemy jako:
   - dla pól typu `String`: długość tego łańcucha,
   - dla pól typu **całkowitego**: długość binarnej reprezentacji wartości tego pola (długość łańcucha zwróconego przez metodę o nazwie `toBinaryString` z odpowiedniej klasy opakowującej),
   - dla pól typu `Address`: suma metod `length` klasy `Address` wywołanych kolejno dla każdego z pól tego obiektu,
   - -1, jeśli podany łańcuch wejściowy **nie jest nazwą żadnego z pól**.

Asercje:

2. *(3 pkt.)* W każdej metodzie/konstruktorze zaimplementuj testowanie parametrów przy użyciu **asercji**.
   Wymagania:
   - niech **przekazane obiekty nie mogą być `null`-ami**,
   - **numer domu** oraz **numer telefonu** nie mogą być zerowe ani ujemne,
   - **pesel** nie może być ujemny (dla chętnych: można sprawdzić sumę kontrolną),
   - **łańcuch** będący parametrem metody `length` z 1. zadania musi być zgodny z nazwą jednego z pól klasy, do której należy.

   Można też dopisać dodatkowe asercje według własnego pomysłu.
3. *(2 pkt.)* W obu klasach proszę dopisać metodę `main`, w której znajdą się **wywołania wszystkich metod, w których w 2. zadaniu dopisano asercje**. Do wywołań proszę przekazywać błędne argumenty, żeby zobaczyć działanie asercji.
   W trakcie pisania kolejnych wywołań trzeba będzie zakomentować poprzednie, żeby program nie kończył działania za wcześnie.

Junit:

4. *(4 pkt.)* Proszę napisać **klasę testów jednostkowych** dla klasy `Address` przy pomocy **JUnit5**. Niech testy będą wykonane dla **konstruktora** oraz **metody `length`** z 1. zadania. Proszę przynajmniej raz wykorzystać metody:
   - `assertNotNull` lub `assertNull`,
   - `assertEquals`,
   - `assertNotSame` lub `assertSame`,
   - `assertTrue`,
   - `assertFalse`

   Wszystkie powyższe metody należą do klasy `org.junit.jupiter.api.Assertions` (link do dokumentacji).

Wskazówki do JUnit – alternatywne sposoby użycia (wybieramy jeden, ale w wolnym czasie warto spróbować też innych):

I. Kompilowanie i uruchamianie z **linii poleceń**: jak w wykładzie (strona 14), przy czym używamy archiwum jar w najnowszej wersji: link do pobrania.
II. **IntelliJ IDEA** + JUnit5:
   - Ustawiamy kursor na nazwie klasy, którą chcemy testować --> naciskamy **Ctrl + Shift + T** --> "Create new Test..."
   - W oknie "Create Test" zaznaczamy "**Fix**", jeżeli JUnit5 nie ma jeszcze w projekcie, i automatycznie ściągamy bibliotekę. Można tu też np. zaznaczyć metody przeznaczone do testowania.

III. **IntelliJ IDEA** + JUnit5 + Maven/Gradle: https://www.jetbrains.com/help/idea/junit.html
IV. **Visual Studio Code** + JUnit5 + Maven/Gradle: https://code.visualstudio.com/docs/java/java-testing
V. **NetBeans** + JUnit5: wykład, od strony 16.

Status przesłanego zadania

Status przesłanego zadania	Przesłane do oceny
Stan oceniania	Ocenione