

7. feladatsor

Kezdés: jún 4, 19:21

Kvízinstrukciók

1. kérdés

0 pont

Programozási nyelvek (BSc, 18) Java 7. feladatsor

1. feladat

a

Írjunk `power()` függvényt, amely természetes számok hatványozását valósítja meg (a hatvány alap és kitevő is természetes szám). Bár az eredmény mindig egész lesz, a visszatérési érték legyen `double`, mert nemsokára továbbfejlesztjük ezt a függvényt. A 0 nulladik hatványa most legyen 1. Tesztelje fehérdozoz-testtel a `power()` függvényt.

b

Fejlesszük tovább a `power()` függvényt úgy, hogy negatív egész kitevővel is helyesen működjön.

Próbálja ki, hogy az a feladatban megírt unit tesztek továbbra is sikeresen lefutnak-e.

Bővítse a unit teszteket negatív kitevő helyességét ellenőrző esetekkel.

2. feladat

A `Book` osztály egy árverési tételt (könyv) reprezentál.

Tesztelje fehérdozoz-testtel a `Book` osztályt! Törekedjen a következő metrikák maximalizálására:

- metódusok lefedettsége
- elágazások lefedettsége (döntési pontok mindkét ága)
- feltételek lefedettsége (részfeltételek a logikai kifejezésekben)
- ciklusok lefedettsége (ciklusok, 0, 1, 2 fordulóra)

Az osztály rendelkezik egy beágyazott felsoroló osztállyal (`Book.Genre`), mely a lehetséges műfajokat tartalmazza, (azonos írásmóddal): `FANTASY`, `SATIRE`, `SCIFI`, `PHILOSOPHY`, `EDUCATIONAL`.

Az osztálynak öt rejtett adattagja van:

- egy szöveg típusú író,
- egy szöveg típusú cím,
- egy egész típusú kikiáltási ár (angolul *reserve price*),
- egy szintén egész típusú azonosító,

- illetve egy `Genre` típusú, a műfajt tároló példányváltozó.

Az osztálynak legyen egy rejtett konstruktora: paraméterként megkapja az alkotó nevét, a könyv címét, a műfajt (`Book.Genre` típusú), valamint a kikiáltási árat, és beállítja a megfelelő adattagokat. Az azonosító legyen mindig a legutolsóként használt azonosítónál 1-gyel nagyobb egész, `0`-tól indulva.

Definiáljunk egy osztályszintű `make()` nevű metódust is.

A `make()` metódus szintén az alkotó nevét, a könyv címét, valamint szöveges paraméterként a műfaj nevét, és végül a kikiáltási árát kapja meg paraméterként.

A metódus először ellenőrzi, hogy a paraméterek megfelelőek-e. Amennyiben igen, akkor létrehozza és visszaadja a paramétereknek megfelelő `Book` típusú objektumot. Ha a paraméterek nem megfelelőek, akkor a metódus `null`-t adjon vissza.

- Az író neve és a könyv címe akkor megfelelő, ha nem egy `null` referencia, legalább 2 hosszú, betűkből, számokból és szóközből áll.
- A kikiáltási ár akkor megfelelő, ha pozitív szám.
- A műfaj akkor megfelelő, ha konvertálható egy megfelelő `enum` értékke.

Írjon `isSameGenre()` osztályszintű metódust, amely a paraméterként kapott két `Book` objektumról eldönti, hogy ugyanolyan műfajú könyveket tárolnak-e.

Írjon `compare()` néven metódust, amellyel a `Book` objektumot össze lehet hasonlítani egy másik, paraméterként kapott `Book` objektummal. A metódussal csak azonos műfajú könyveket lehet összehasonlítani; ha a két könyv műfaja nem egyezik meg, akkor a függvény dobjon `IllegalArgumentException` kivételt. Két azonos műfajú könyv közül az a nagyobb, amelyik kikiáltási ára nagyobb. Ha az aktuális könyv nagyobb a paraméterként kapottnál, `1` legyen a visszatérési érték; ellenkező esetben `-1`; ha a két könyv a megadott szempont szerint egyenlő, akkor pedig `0`.

3. feladat

Az `Adder` osztály `add()` osztályszintű metódusa két számot vár paraméterként és visszaadja azok összegét. Fiktív körülmények miatt a paraméterek és a visszatérési érték típusa is karakterlánc (`String`). Az átadni kívánt számokat tehát előbb ilyen módon el kell kódolni, majd az eredményt megfelelően dekódolni kell. Ha a karakterlánc nem értelmezhető számként, a program dobjon `IllegalArgumentException` kivételt.

Az `Adder` osztály lefordított kódja itt érhető el: Fájlok/Adder.class

```
public class Adder {  
    public static String add(String a, String b){...}  
}
```

Tesztelje feketedoboz-teszteléssel az `add()` metódust, szem előtt tartva a következő szempontokat:

- összeadás helyes működése
- összeadás algebrai tulajdonságai: kommutativitás, asszociativitás, egységelem
- lebegőpontos értékekre `0.01` pontossággal helyesen működik

- kettes számrendszerbeli számokat is elfogad
- angol nyelven leírt számokat is elfogad
- a 0 körüli műveletek (+- 1) helyesek
- nem értelmes számok vagy null esetén esetén IllegalArgumentException kivételt dob
- a számok elején illetve végén lévő szóközöket figyelmen kívül hagyja

1. gyakorló feladat

Készítsen MathUtils osztályt Increment() statikus metódussal, amely biztonságos egész-szám inkrementálást valósít meg: a függvény csak akkor növeli meg a paraméterként kapott egész számot, ha az nem a legnagyobb ábrázolható egész szám. Az eredményt visszatérési értékben adja meg.

Tesztelje fehérdoboz-teszteléssel az Increment() metódust.

2. gyakorló feladat

Készítse el a verem [_ \(https://hu.wikipedia.org/wiki/Verem_\(adatszerkezet\)\)](https://hu.wikipedia.org/wiki/Verem_(adatszerkezet)) Stack adatszerkezet egy egyszerű implementációját!

- A Stack csak int -eket tud tárolni.
- Két fő művelete van:
- push(): behelyez egy elemet a verem tetejére,
- pop(): eltávolítja az utoljára behelyezett elemet a veremből is vissza adja annak az értékét.
- Továbbá ellenőrizhető, hogy üres-e a verem (empty()) és lekérdezhető a mérete (size()).
- Amennyiben egy üres veremből próbálunk adatot kivenni, váltódjon ki NoSuchElementException (java.util csomag)!

Tesztelje le a verem implementációt (a tanult tesztkörnyezetben) a következőképp:

- ellenőrizze, hogy az utoljára behelyezett elem az első, amit kikerül,
- egy frissen létrehozott verem üres,
- egy elemmel rendelkező verem nem üres,
- üres veremből nem lehet elemet kivenni (teszteljük, hogy dobódik-e kivétel),
- jól működik a size() metódus extrémális értékeken (0,1),
- ha az utolsó elem is kikerül, a verem üres.

3. gyakorló feladat

Implementálja a Caesar-kódolást [_ \(https://hu.wikipedia.org/wiki/Caesar-rejtjel\)](https://hu.wikipedia.org/wiki/Caesar-rejtjel) az angol ábécéhez! Ehhez készítse el a Caesar osztályt, mely következőképp működik:

- az eltolási távolságot konstruktor paraméterként kapja meg (n) és tárolja el,
- ha nem igaz, hogy n eleme [1, 25]-nek, akkor dobjon IllegalArgumentException -t,
- legyen cipher() művelete, ami elkódol egy angol szöveget n eltolással,
- valamint egy decipher() művelet, mely dekódol egy megfelelő Caesar kódot (itt nem kell hibát kezelni),
- a cipher() művelet kivételt dob, ha nem angol ábécébeli karaktert észlel.

Tesztelje megoldását adatvezérelt módon, **bemeneti szöveg - elvárt kód** párok felsorolásával (legalább 5db)! Ehhez használja a JUnit környezet [Parameterized tests](https://github.com/junit-team/junit4/wiki/Parameterized-tests) (<https://github.com/junit-team/junit4/wiki/Parameterized-tests>) megoldását!

Az ellenőrzésekhez (*assertions*) próbálja meg a [Hamcrest](http://hamcrest.org/JavaHamcrest/tutorial) (<http://hamcrest.org/JavaHamcrest/tutorial>) tesztelő könyvtárat használni (az alap JUnit `assert*` műveletek helyett)!

Egészítse ki a fenti paramétereket az eltolás mértékével (`n`), és tesztelje legalább 3 különböző `n` értékre!

Készítsen *negatív* teszteseteket is, amik azt ellenőrzik, hogy a hibás paraméterek esetén is jól működik az osztály (kivételt dob)!

4. gyakorló feladat

Tesztelje le a **6. feladatsor 2/b. feladatának** megoldását JUnit segítségével! Ehhez szükség lesz a beolvasó, összegző, valamint kiíró logika szétválasztására. Ezeket rendezze statikus metódusokba, úgy, hogy ne függjenek egymástól, csupán a paraméterek/visszatérési érték segítségével tudjanak kommunikálni egymással!

Mindhárom művelethez készítsen külön teszt-osztályt, és ügyeljen rá, hogy az egyes teszt-osztályok csak az adott logikát tesztelik.

A teszteléshez hozzon létre több minta bemenetet (szöveges fájlokat)! Kerülje a kódismétlést a paraméterezett tesztek használatával! Törekedjen a komplex ellenőrzések tömör megfogalmazására (Hamcrest matcher könyvtár)!

A szummázó/kiíró logika tesztelése során feltehetjük, hogy a beolvasás már működik, így azt felhasználhatjuk arra, hogy az bemenetet beolvassuk. A bemeneti fájlok tartalmát a teszt metódusok lefutása előtt olvassa be egy megfelelő mezőbe a beolvasó metódus segítségével! Ehhez használja a `@Before` annotációt ([segítség](https://www.baeldung.com/junit-before-beforeclass-beforeeach-beforeall) (<https://www.baeldung.com/junit-before-beforeclass-beforeeach-beforeall>))!

A tesztelés során ügyeljen rá, hogy a fellépő kivételeket, csak az arra szánt tesztekben kezelje le! Egyéb esetben azt kell biztosítani, hogy tesztelés során megnyitott fájlok bezárásra kerüljenek (**tipp** [_https://www.baeldung.com/java-wrapping-vs-rethrowing-exceptions_](https://www.baeldung.com/java-wrapping-vs-rethrowing-exceptions)).

Próbáljon meg minél több lehetséges adatot letesztelni figyelmet fordítva a szélsőséges estekre (egy elemű sor, üres sor, üres fájl, stb)!

Feltöltés

Fájl kiválasztása

Nincs elmentve

Kvíz beadása