Programozási nyelvek II. JAVA

10. gyakorlat

2017. november 20-24.

Szoftver minőségbiztosítás (ismétlés)

Adott: Specifikáció (követelmények halmaza)

Cél: A követelményeket teljesítő ("helyes") program

Megközelítések:

- **Tesztelés:** Futtatás során ellenőrizzük a követelmények teljesülését (kézi, automatikus)
- **Formális verifikáció:** A helyesség (automatikus) levezetése a (formális) specifikációból. Szimbolikus végrehajtás és model-ellenőrzés is ide sorolható.
- **Program szintézis:** Helyes program levezetése a specifikációból
- **És még mások** ...

Milyen a jó teszt? (ismétlés)

FIRST

- Fast (gyors)
- Isolated (egymástól és külvilágtól elkülönölt, egy sikertelen tesztnek pontosan egy oka lehet)
- Repeatable (megismételhető, nincsenek mellékhatások és nem-determinisztikus futás)
- Self-verifying (önellenőrző, minden teszt elbukhat, minden bukásnak pontosan egy oka lehet)
- Timely (megfelelő időben rendelkezésre álló)

Mekkora egységet tesztelünk? (ismétlés)

Egységteszt: A rendszer legkisebb önállóan működő egységeit teszteli (osztály, metódus).
Célok:

- Követelmények teljesülésének ellenőrzése kimeneten
- Funkcionalitás (vagy annak hiányának) dokumentálása
- Kód változtatásakor hibák detektálása (ld. Test driven development, Continous integration)

Elvárások:

- 1:n kapcsolat: Egy teszt pontosan 1 egységet tesztel, de 1 egységre több teszt is juthat.
- Az egységek egymástól, környezettől, felhasználótól függetlenek (nincs mellékhatás)
- Minőségmutató: lefedettség (tesztelt publikus metódusok száma)

Feketedoboz-tesztelés (ismétlés)

- A funkcionalitást teszteljük, az implementáció ismerete nélkül
- Kimerítő tesztelés: összes lehetséges input összehasonlítása a megfelelő outputokkal
- Időigény:
  - Legyenek $p_1, p_2, ..., p_n$ paraméterek típusai rendre $P_1, P_2, ..., P_n$.
  - Ekkor a kimerítő tesztelés időigénye: $T = |P_1| \cdot |P_2| \cdot ... \cdot |P_n|$, tehát $\min_i |P_i|^n \leq T \leq \max_i |P_i|^n$. (Polinomiális a típus értékhalmazának méretében, exponenciális a paraméterek számában.)
- Heurisztikák: ekvivalenciaosztályok, határértékek

Fehérdoboz-tesztelés

- Az implementációt teszteljük a kód ismeretében.
- Teljes lefedettség: Az összes végrehajtási utat teszteljük
- Időigény:
  - Beágyazott elágazások
    - Legrosszabb esetben h magas teljes bináris fával modellezhetőek.
    - Nem-levél csúcsok (elágazási pontok) száma: $O(2^h)$.
    - Levelek (végrehajtási utak) száma: $O(2^h)$.
  - Ciklusok esetében általában eldönthetetlen probléma.
  - Precízebben ld. még: ciklomatikus komplexitás
- Heurisztika: Részletes lefedettség

Teszt-lefedettségi mutatók[1]

- Metódusok lefedettsége (tesztelve van-e elég metódus)
- Utasítások lefedettsége (tesztelve van-e elég utasítás)
- Elágazások lefedettsége (tesztelve van-e elég döntési pont mindkét ága?)
- Feltételek lefedettsége (tesztelve van-e elég feltétel a logikai kifejezésekben?)
- Ciklusok lefedettsége (tesztelve van-e elég ciklus $0, 1, k$ fordulóra?)

[1] Glenford J. Myers (2004). The Art of Software Testing, 2nd edition. Wiley. ISBN 0-471-46912-2.

A tesztek szerkezete (ismétlés)

Előkészítés-Kivitelés-Ellenőrzés (Arrange-Act-Assert, AAA)

```
@Test
public void testStringIsReversed() {
  // arrange //
  String input="Hello world";

  // act //
  String result = MyLib.reverse(input);

  // assert //
  assertEquals("dlrow olleH", result);
}
```

Feladat (A `market.Fruit` osztály)

- Töltsd le a Fruit.java fájlt! Az osztály egy eladható gyümölcsöt reprezentál.
- Teszteld fehérdoboz-teszteléssel a Fruit osztályt!
- Törekedj 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)

Feladat (A `market.Fruit` osztály)

- Az osztálynak két rejtett adattagja van:
  - egy szöveges típusú `name`, amely a gyümölcs nevét tárolja
  - és egy egész szám típusú `price`, ami a gyümölcs árát tárolja.
- Az osztálynak legyen egy rejtett konstruktora, amely paraméterben megkapja a nevet és az árat, és beállítja a megfelelő adattagokat.

Feladat (A `market.Fruit` osztály)

Legyen egy statikus `make` metódus, amely szintén egy nevet és egy árat kap.

- A metódusnak ellenőriznie kell a paramétereket, és amennyiben azok megfelelőek, akkor hozza létre, és adja vissza a paramétereknek megfelelő Fruit objektumot.
- Ha a paraméterek nem jók, akkor a metódus null-t adjon vissza.
  - A nevet tartalmazó paraméter akkor megfelelő, ha csak betűt tartalmaz és legalább 2 karakter hosszú.
  - Az árat tartalmazó paraméter pedig akkor helyes, ha pozitív, de legfeljebb 5000, továbbá és 0-ra vagy 5-re végződik.

Feladat (A `market.Fruit` osztály)

A `make` metódust terheljük túl egy másik metódussal, mely szövegek egy listáját várja paraméterül, és `Fruit` objektumok listájával tér vissza. Az érvényes szövegobjektumok mindegyike 2 db, vesszővel elválasztott mezőt tartalmaz: a gyümölcs nevét, és az árát. Például:

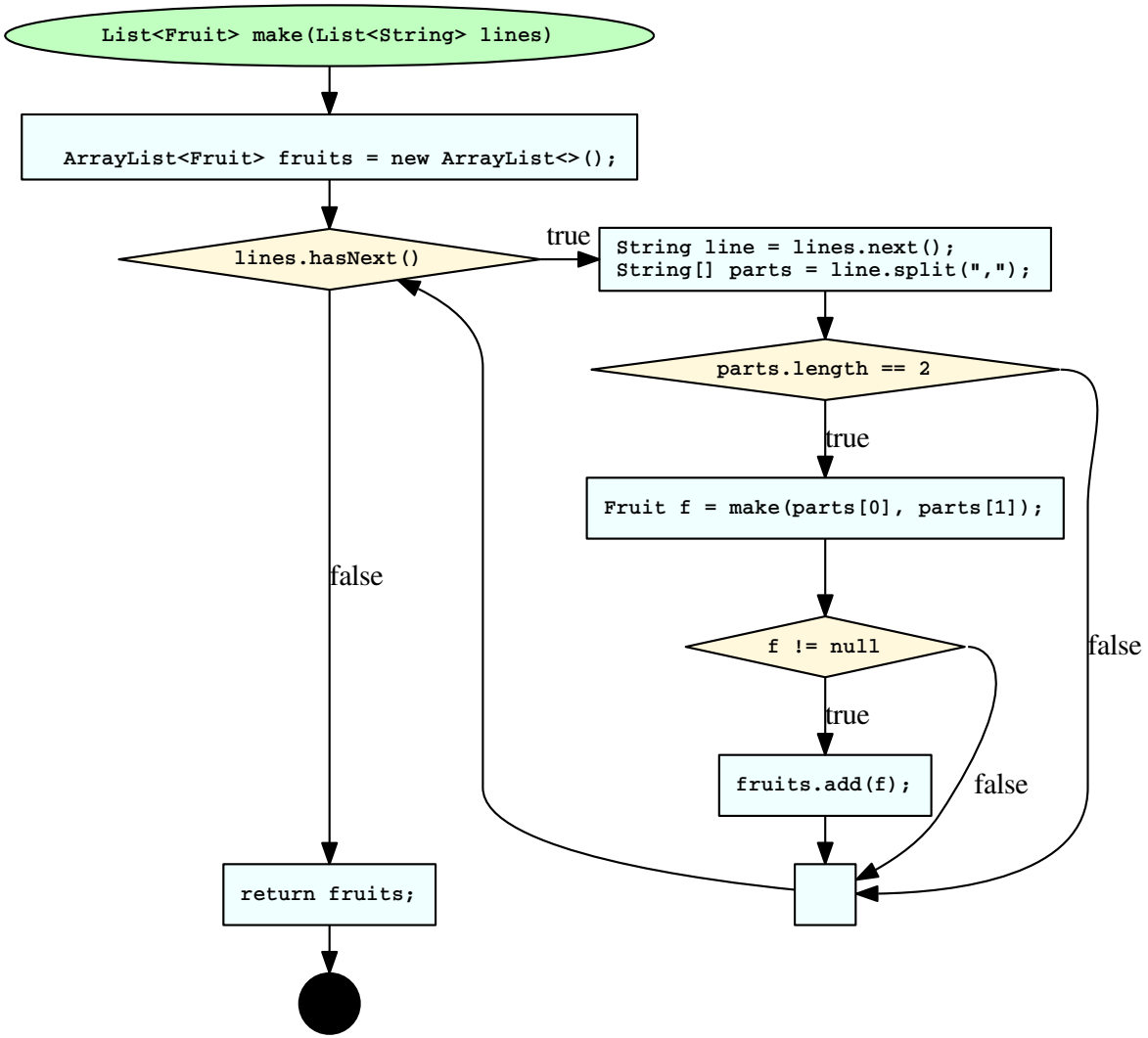
```
korte, 130
```

Ha a metódus ilyen sort olvas be a listából, a `make` metódus másik túlterhelt példánya segítségével létrehozza a gyümölcs objektumokat, majd egy listában visszaadja azokat. Amennyiben a sor érvénytelen, az utóbbi metódus null referenciát ad vissza.

A `Fruit.make(List<String> lines)` metódus kódja

```
...
    public static List<Fruit> make(List<String> lines) {
        ArrayList<Fruit> fruits = new ArrayList<>();
        for (String line : lines) {
            String[] parts = line.split(",");
            if (parts.length == 2) {
                Fruit f = make(parts[0], parts[1]);
                if (f != null) {
                    fruits.add(f);
                }
            }
        }
        return fruits;
    }
...
```

A `Fruit.make(List<String> lines)` metódus vezérlésfolyam-gráfja



Feladat (Az `auction.Book` osztály)

- Töltsd le a Book.java fájlt! Az osztály egy árverési tételt (könyv) reprezentál.
- Teszteld fehérdoboz-teszteléssel a Book osztályt!
- Törekedj 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)

Feladat (Az `auction.Book` osztály)

Az osztály rendelkezik egy beágyazott felsoroló osztállyal (`auction.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 leütési ár(angolul hammer price),
- egy szintén egész típusó azonosító,
- illetve egy Genre típusú, a műfajt tároló példányváltozó.

Feladat (Az `auction.Book` osztály)

Az osztálynak legyen egy rejtett konstruktora, amely paraméterként megkapja az alkotó nevét, a könyv címét, a kikiáltási árat, valamint a műfajt (`Genre` típusú), és beállítja a megfelelő adattagokat (a leütési ár legyen a kikiáltási ár). Az azonosító legyen mindig a legutolsóként használt azonosítónál egyel nagyob egész, 0-tól indulva.

Feladat (Az `auction.Book` osztály)

Definiáljunk egy osztályszintű `make` nevű metódust is.

- A `make` metódus szintén az alkotó nevét, a könyv címét és a kikiáltási árát kapja meg paraméterként, valamint szöveges paraméterként a műfaj nevét.
- A metódus először ellenőrzi, hogy a paraméterek megfelelőek. 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 szóközökből áll.
  - A kikiáltási ár akkor megfelelő, ha pozitív szám.
  - A műfaj akkor megfelelő, ha konvertálható egy megfelelő enum értékké.

Feladat (Az `auction.Book` osztály)

A `make` metódust terheljük túl egy másik metódussal, mely szövegek egy listáját várja paraméterül, és `Book` objektumok listájával tér vissza. Az érvényes szövegobjektumok mindegyike 4 db, pontosvesszővel elválasztott mezőt tartalmaz: az alkotó nevét, a könyv címét, a kikiáltási árat, valamint a műfajt. Például:

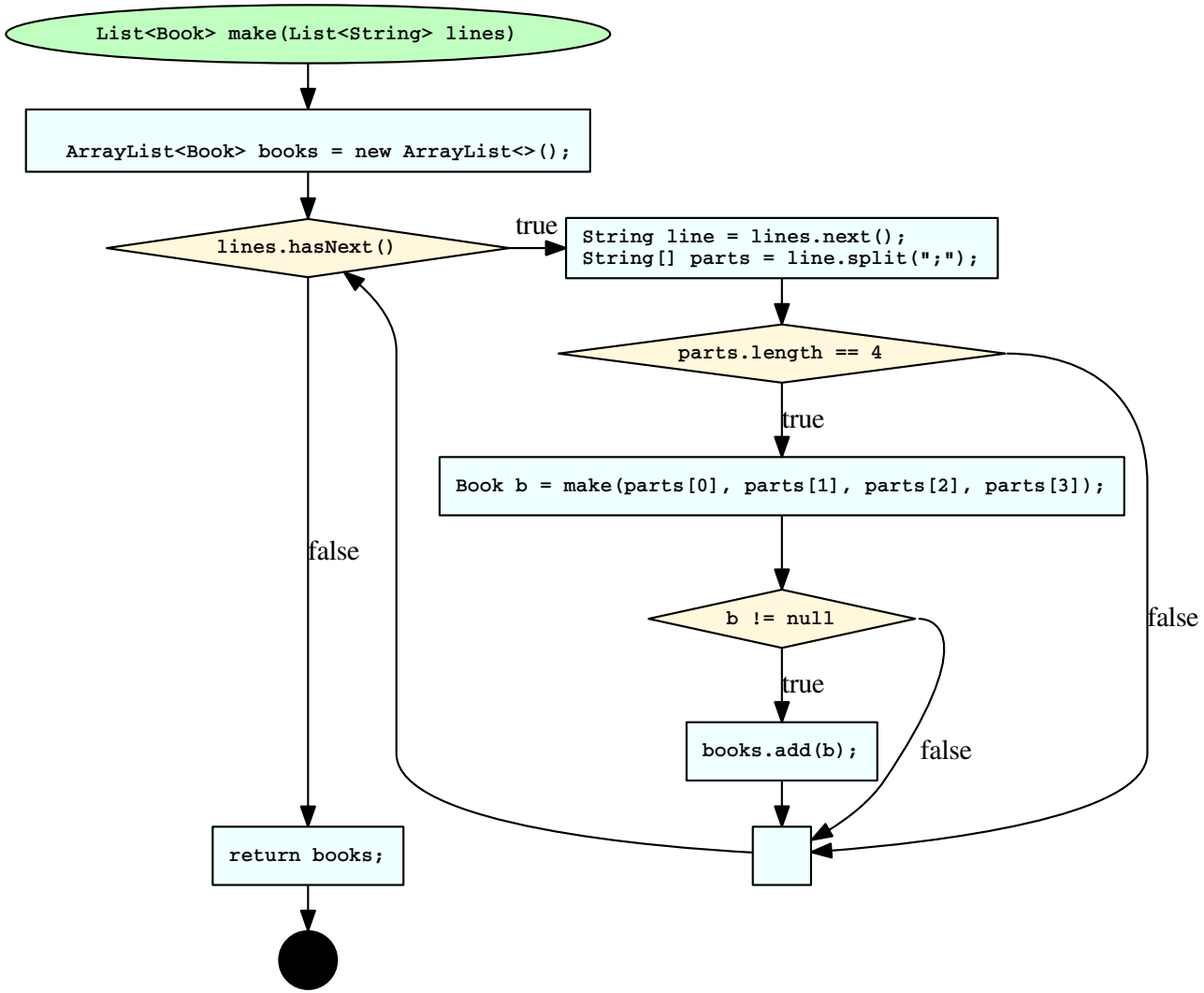
```
Giles Andreae;GIRAFFES CAN'T DANCE;PHILOSOPHY;50000
```

Ha a metódus ilyen sort olvas be a listából, a `make` metódus másik túlterhelt példánya segítségével létrehozza a könyv objektumokat, majd egy listában visszaadja azokat. Amennyiben a sor érvénytelen, az utóbbi metódus null referenciát ad vissza.

A `Book.make(List<String> lines)` metódus kódja

```
...
public static List<Book> make(List<String> lines) {
  ArrayList<Book> books = new ArrayList<>();
  for (String line : lines) {
    String[] parts = line.split(";");
    if (parts.length == 4) {
      Book b = make(parts[0], parts[1], parts[2], parts
          [3]);
      if (b != null) {
        books.add(b);
      }
    }
  }
  return books;
}
...
```

A `Book.make(List<String> lines)` metódus vezérlésfolyam-gráfja



Feladat (A `GTNGame` osztály)

Töltsd le a GTNGame osztályt, mely a jól ismert (*"Guess the number"*) számkitalálos játékot valósítja meg.

- A főprogram parancssori argumentumként egy pozitív `maxNum` számot vár a felhasználótól, ami a megoldás felső korlátját jelenti majd.
- Amennyiben a felhasználó megfelelő számú argumentumot ad meg, vagy nem egy pozitív számot ad, a program írjon ki üzenetet.

Feladat (A `GTNGame` osztály)

Ezután a program "gondoljon" egy számra 1 és maxNum között (használható a `java.util.Random` osztály `nextInt` metódusa). Ezt követően ciklusban kérjen be számokat a felhasználótól.

- Amennyiben a felhasználó eltalálja melyik számra gondolt a program, a program lépjen ki a ciklusból és írjon ki gratuláló üzenetet.
- Amennyiben a felhasználó nem egy pozitív számot ad, írjon ki üzenetet, majd kérjen be újra számot.
- Amennyiben a felhasználó valid számot ad meg, de nem azt, amelyikre gondolt a program, írja ki, hogy kisebb vagy nagyobb volt a tipp a megoldásnál.

Feladat (Az `GTNGame` osztály)

- Töltsd le a megoldást tartalmazó GTNGame.java fájlt!
- Rajzold meg az osztály vezérlésfolyam-gráfját.
- Teszteld fehérdoboz-teszteléssel a GTNGame osztályt!
- Törekedj 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)