CR7 - Ajándékzacskók és kuponok bevezetése

Bevezető

A JóÁr áruház pénzügyesei felsóhajtottak: kiderült, hogy a **0,5% kártyabónusz** túl bőkezű volt. A vezetőség ezért **átmenetileg 0,0%**-ra állítja a kártyabónuszt (majd *meglátjuk*). A marketing azonban nem alszik: „Ha már bónusz nincs, legalább **ingyen papírzacskó** legyen… sőt, adjunk **ajándékkuponokat** is!” – hangzott el a heti marketing megbeszélésen. Önre vár a megvalósítás!

Feladat

Az Ön feladata, hogy az eddigi megoldást a **CR7**-nek megfelelően módosítsa.
Ez a CR a sorozat **utolsó eleme** – ha idáig eljutott, **büszke lehet magára!**

Javaslatok:

- A CR6-tól bevezetett új szignatúra miatt **érdemes lehet** a korábbi egységteszteket igazítani az új felülethez.
- Használja az alfejezet végén lévő **egységteszteket** és a **hangulatvezérelt kódolás** eszközeit.

Érintett felületek

A CR6 óta a számítási belépési pont:

```
ArInfo getKosarAr(Kosar k, Idoszak i, List<String> kk, FizetesiMod m)
```

ahol a `FizetesiMod` két értéket vehet fel: `KESZPENZ`, `KARTYA`.

Változások az ÁrInfo struktúrában

A korábbi két mező:

- **fizetendő összeg** (double, forintban),
- **fel nem használt kuponok listája** (List<String>),

kiegészül az alábbiakkal:

- **ajándék papírzacskók száma** (int),
- **ajándékkuponok listája** (List<String>).

Fizetési szabályok:

- **Készpénz**: marad az **5 Ft-os kerekítés** (CR0 szabály).
- **Kártya**:
  - a **kártyabónusz = 0,0%** (CR7),
  - **nincs** 5 Ft-os kerekítés,
  - a végösszeget **10 fillérre** kell kerekíteni (0,0–4,9 fillér → lefelé, 5,0–9,9 fillér → felfelé).

Új ajándékszabályok

Papírzacskók

- **A kártyabónusz legyen 0.0%.**
- **Minden 5 kg megvásárolt termék** (alma + banán összesen) **után jár egy ajándék papírzacskó**. 5 kg-ig nem jár ilyen ajándék, mert csak minden egész 5 kg után jár ajándék.
- Vigyázat, habár az `A-FREE1` és a `B-FREE1` kupon csökkenti a fizetendő mennyiséget, de ez csak logikai csökkentés. **A papírzacskó-ajándék alapja a fizikai súly**.
- Az ajándék papírzacskók számát vissza kell adni az `ÁrInfo` struktúrában.
- Az ajándékkuponok listáját vissza kell adni az `ÁrInfo` struktúrában. Ha nem jár egy ajándékkupon sem, akkor ez a lista üres.
- A kapott darabszámot az `ArInfo.ajandekZacskokSzama` mezőben kell visszaadni.

Ajándékkuponok

- **Ha vásárlás végösszege nagyobb egyenlő, mint 20.000 forint, akkor minden 20.000 forint után jár egy ajándékkupon.** Tehát például ha a fizetendő összeg 58.000 forint, akkor 2 ajándékkupon jár. A kuponokat az alábbi valószínűség szerint sorsolja a rendszer:
  - `A10` vagy `B10`: **20% esély**, azaz **minden ötödik** ajándékkupon vagy `A10`, vagy `B10` kupon.
  - `A-FREE1` vagy `B-FREE1`: **10% esély**.
  - `A5-MAX10` vagy `B5-MAX10`: **5% esély**.
  - `X5`: **5% esély**.
  - `A5-MAX15` vagy `B5-MAX15`: **2% esély**.
  - `X10`: **2% esély**.
  - `X5-MAX10`: **1% esély**.
  - `KUPON-2000-ULTRAMAX`: **1% esély**.
  - `A5` vagy `B5`: **maradékelv**, azaz a maradék esély százalékában.

- Persze ezek az esélyek a jövőben változhatnak.
- A kapott típusokat az `ArInfo.ajandekKuponok` listában kell visszaadni; ha nem jár kupon, a lista **üres**.

Példák

1. **Zacskó határérték**
   Kosár: 4,9 kg alma + 0,1 kg banán → **5,0 kg**
   Zacskók: **1 db** (fizikai súly alapján).
   Ajándékkupon: a végösszegtől függ (20 000 Ft-onként 1 db).
2. **Több többszörös**
   Kosár: 6,8 kg alma + 3,5 kg banán → **10,3 kg → 2 db zacskó**.
3. **FREE kupon nem csökkenti a fizikai súlyt**
   Kosár: 5,2 kg alma; Kuponok: [A-FREE1]
   Zacskók: **1 db** (nem 0), mert a fizikai súly 5,2 kg.
4. **Ajándékkupon darabszám**
   Készpénz, nagy kosár → végösszeg 50 600 Ft → **2 db kupon** (mind a listából sorsolva).
5. **Ajándékkupon határ**
   Készpénz: ~20 305 Ft → **1 db kupon**.
   Kártya: ~19 999,9 Ft → **0 db kupon** (CR7-ben 0% bónusz, csak 10 filléres kerekítés él).
6. **ULTRAMAX kinulláz**
   Kosár: 3 kg alma; Kupon: [KUPON-2000-ULTRAMAX] → **0 Ft**
   Ajándékkupon: **0 db** (mert < 20 000 Ft).
   Zacskó: **0 db** (fizikai súly 3,0 kg).
7. **Nagy fizikai súly, sok zacskó**
   Kosár: 60,2 kg alma + 63,2 kg banán → **123,4 kg → 24 db zacskó**.
   Ajándékkupon: végösszeg/20 000 Ft szerint.

Fejlesztési követelmények (összefoglaló)

1. **ÁrInfo bővítése**:
   - `int ajandekZacskokSzama`
   - `List<String> ajandekKuponok`
2. **Kártya logika**: kártyabónusz **0,0%**, kerekítés **10 fillérre** (CR6 alapján, de bónusz nélkül).
3. **Zacskók számítása**: `floor((alma_kg + banan_kg) / 5.0)` – **fizikai** súly alapján.
4. **Ajándékkuponok számítása**: `floor(vegosszeg / 20000.0)`; típusok sorsolása a megadott eloszlással.
5. **Determinálhatóság teszthez**: javasolt a véletlen sorsolást **injektálható RNG**-re vagy **seed**-re bízni, hogy a tesztek reprodukálhatók legyenek.
6. **Visszafelé kompatibilitás**: a korábbi CR-ek logikája (mennyiségi kedvezmények, kuponok, MAX, X, ULTRAMAX, kerekítések) változatlanul működjön.

Egységtesztek

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import java.util.List;
import java.util.Set;

import org.aruhaz.*;

class AruhazCR7Tesztek {

    static Aruhaz target;
    static Idoszak normal;

    // Ajándékkuponok megengedett típuskódjai (specifikáció szerint)
    static final Set<String> ENGED_ajandek = Set.of(
            "A10", "B10",
            "A-FREE1", "B-FREE1",
            "A5-MAX10", "B5-MAX10",
            "X5", "X10", "X5-MAX10",
            "A5-MAX15", "B5-MAX15",
            "KUPON-2000-ULTRAMAX",
            "A5", "B5"
    );

    @BeforeAll
    static void init() {
        target = new Aruhaz();
        normal = new Idoszak("Normál");
        normal.setEgysegAr(Termek.ALMA, 500.0);
        normal.setEgysegAr(Termek.BANAN, 450.0);
        normal.setKedvezmeny(Termek.ALMA, 5.0, 0.1);
        normal.setKedvezmeny(Termek.ALMA, 20.0, 0.15);
        normal.setKedvezmeny(Termek.BANAN, 2.0, 0.1);
        target.addIdoszak(normal);
    }

    // --- Segéd: 5 Ft-os kerekítés (készpénz) a CR0/CR1 logika szerint
    private double kerekites5re(double osszeg) {
        double maradek = osszeg % 10.0;
        double base = Math.floor(osszeg / 10.0) * 10.0;
        if (maradek < 2.5) return base;
        if (maradek < 5.0) return base + 5.0;
        if (maradek < 7.5) return base + 5.0;
        return base + 10.0;
    }

    // --- Segéd: 10 fillérre kerekítés (kártya) a CR6 logika szerint
    private double kerekites10Fillerre(double osszeg) {
        return Math.round(osszeg * 10.0) / 10.0;
    }

    // 1) Ajándék papírzacskó: első teljes 5 kg után 1 db (fizikai súly
alapján!)
    @Test
    void teszt_cr7_1_zacskok_5kg_hatar() {
        // 4.9 kg alma + 0.1 kg banán = 5.0 kg összesen -> 1 zacskó
        Kosar kosar = new Kosar(List.of(
                new Tetel(Termek.ALMA, 4.9),
                new Tetel(Termek.BANAN, 0.1)
        ));
        ArInfo ar = target.getKosarAr(kosar, normal, List.of(),
```

```
FizetesiMod.KESZPENZ);
        assertEquals(1, ar.getAjandekZacskokSzama());
    }

    // 2) Ajándék papírzacskó: több többszörös eset (10.3 kg -> 2 db)
    @Test
    void teszt_cr7_2_zacskok_tobb_tobbszoros() {
        Kosar kosar = new Kosar(List.of(
                new Tetel(Termek.ALMA, 6.8),
                new Tetel(Termek.BANAN, 3.5) // összesen 10.3 kg ->
floor(10.3/5)=2
        ));
        ArInfo ar = target.getKosarAr(kosar, normal, List.of(),
FizetesiMod.KESZPENZ);
        assertEquals(2, ar.getAjandekZacskokSzama());
    }

    // 3) "FREE" kupon csak logikailag csökkenti a fizetendő mennyiséget,
    // de a zacskó SZÁMÍTÁSNÁL a fizikai súly számít
    @Test
    void teszt_cr7_3_zacskok_free_kupon_nem_csokkenti_a_fizikai_sulyt() {
        // 5.2 kg alma, A-FREE1 lecsökkentheti a fizetendőt max 1 kg-gal,
de
        // zacskónál továbbra is 5.2 kg -> 1 zacskó jár
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA, 5.2)));
        ArInfo ar = target.getKosarAr(kosar, normal, List.of("A-FREE1"),
FizetesiMod.KESZPENZ);
        assertEquals(1, ar.getAjandekZacskokSzama());
    }

    // 4) Ajándékkupon darabszám: 58 000 Ft végösszeg -> 2 db ajándékkupon
    // (fizetendő összeg alapú)
    @Test
    void teszt_cr7_4_ajandekkupon_darabszam_58000() {
        // Egyszerűen állítsunk elő ~58 000 Ft körüli végösszeget kuponok
nélkül:
        // Pl. 100 kg alma * 500 = 50 000 Ft és 20 kg banán * 450 = 9 000
Ft -> 59 000 Ft
        // Mennyiségi kedvezmények miatt az alma 20kg felett -15%:
100*500=50000 -> 42500
        // Banán 20 kg felett 10% (2 kg-tól) -> 20*450=9000 -> 8100
        // Össz: 42500 + 8100 = 50600 -> kerekítés készpénznél 5 Ft-ra
(50600 marad)
        Kosar kosar = new Kosar(List.of(
                new Tetel(Termek.ALMA, 100.0),
                new Tetel(Termek.BANAN, 20.0)
        ));
        ArInfo ar = target.getKosarAr(kosar, normal, List.of(),
FizetesiMod.KESZPENZ);
        // FIZETENDŐ a CR-implementáció szerint (itt az elméleti kalkuláció
alapján ~50600):
        int varhatoDb = (int) Math.floor(ar.getAr() / 20000.0);
        assertEquals(varhatoDb, ar.getAjandekKuponok().size());
        // és minden kupon típuskódja legyen engedélyezett

assertTrue(ar.getAjandekKuponok().stream().allMatch(ENGED_ajandek::contains
));
        // Ebben a konkrét példában 50600 -> 2 db ajándékkupon
        assertEquals(2, ar.getAjandekKuponok().size());
    }
```

```
    // 5) Ajándékkupon: pontos határ - 20000 Ft -> 1 db, 19999.9 Ft -> 0 db
    @Test
    void teszt_cr7_5_ajandekkupon_hatar_ertekek() {
        // 20000 Ft környéke: konstruáljunk:
        // 40 kg alma 20kg felett -15%: 40*500=20000 -> 17000 Ft (!), ez
nem jó.
        // Legyen 20 kg alma: 20*500=10000 -> -15% = 8500
        // + 25 kg banán: 25*450=11250 -> -10% = 10125
        // Össz: 18625 -> kevés. Adjunk hozzá 3 kg almát: +1500 (most 23 kg
alma -> -15% még mindig)
        // Új alma bruttó: 23*500=11500 -> -15% = 9775; banán változatlan
10125; össz: 19900.
        // Még +1 kg banán: +450 bruttó -> kedv után +405 -> 20305
(készpénz kerekít: 20305)
        Kosar kosar = new Kosar(List.of(
                new Tetel(Termek.ALMA, 23.0),
                new Tetel(Termek.BANAN, 26.0)
        ));
        ArInfo ar = target.getKosarAr(kosar, normal, List.of(),
FizetesiMod.KESZPENZ);
        // 20305 -> 1 db kupon
        int db = (int) Math.floor(ar.getAr() / 20000.0);
        assertEquals(db, ar.getAjandekKuponok().size());
        assertEquals(1, ar.getAjandekKuponok().size());

assertTrue(ar.getAjandekKuponok().stream().allMatch(ENGED_ajandek::contains
));

        // 19999.9 eset (kártyával szépen modellezhető, mert nincs 5 Ft-os
kerekítés és 0% bónusz CR7-ben):
        // Konstruáljunk egy kosarat, amely kártyán pont 19999.9-re adódik
(itt inkább azt teszteljük, hogy <20000 -> 0 db).
        // Válasszunk egy kisebb kosarat és skálázzuk:  (Elegendő az elv:
<20000 esetén 0 db.)
        Kosar k2 = new Kosar(List.of(
                new Tetel(Termek.ALMA, 10.0),    // 10*500=5000 -> 10% kedv?
Nem, 10 kg almánál normálban 5% csak 20-tól 15%. 5kg-tól 10%. 10kg -> 10%
-> 4500
                new Tetel(Termek.BANAN, 20.0)    // 20*450=9000 -> -10% =
8100
        ));
        // össz: 4500 + 8100 = 12600 -> jóval 20000 alatt
        ArInfo ar2 = target.getKosarAr(k2, normal, List.of(),
FizetesiMod.KARTYA);
        assertTrue(ar2.getAr() < 20000.0);
        assertEquals(0, ar2.getAjandekKuponok().size());
    }

    // 6) Ajándékkupon lista üres, ha a végösszeg 0 Ft (ULTRAMAX-szal
kinullázva)
    @Test
    void teszt_cr7_6_ultramax_nullaz_es_ajandekkupon_nincs() {
        // Olyan kosár, amit az ULTRAMAX kinulláz
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA, 3.0))); //
1500 Ft
        ArInfo ar = target.getKosarAr(kosar, normal, List.of("KUPON-2000-
ULTRAMAX"), FizetesiMod.KESZPENZ);
        assertEquals(0.0, ar.getAr(), 0.0001);
        assertEquals(List.of(), ar.getAjandekKuponok());
        // Zacskó ettől még járhat (fizikai súly alapján 3.0 -> 0 db)
        assertEquals(0, ar.getAjandekZacskokSzama());
```

```
    }

    // 7) Ajándékkuponok – csak engedélyezett kódok szerepelhetnek
    @Test
    void teszt_cr7_7_ajandekkuponok_megengedett_halmazbol() {
        // Konstruáljunk nagy végösszeget, hogy kapjunk több kupont
        Kosar kosar = new Kosar(List.of(
                new Tetel(Termek.ALMA, 200.0),  // 200*500=100000 -> -15% =
85000
                new Tetel(Termek.BANAN, 50.0)   // 50*450=22500 -> -10% =
20250
        ));
        ArInfo ar = target.getKosarAr(kosar, normal, List.of(),
FizetesiMod.KESZPENZ);
        int db = (int) Math.floor(ar.getAr() / 20000.0);
        assertEquals(db, ar.getAjandekKuponok().size());

assertTrue(ar.getAjandekKuponok().stream().allMatch(ENGED_ajandek::contains
));
    }

    // 8) CR7: kártyabónusz = 0.0% (CR6-ban 0.5% volt) → kártyán most nincs
plusz kedvezmény
    // Összevetjük készpénz vs. kártya: különbség CSAK a kerekítésből
adódhat.
    @Test
    void teszt_cr7_8_kartya_bonusz_nullazva() {
        // Válasszunk olyan összeget, ahol látszik a különbség:
        // 1.333 kg alma -> 1.333*500 = 666.5
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA, 1.333)));
        ArInfo cash = target.getKosarAr(kosar, normal, List.of(),
FizetesiMod.KESZPENZ);
        ArInfo card = target.getKosarAr(kosar, normal, List.of(),
FizetesiMod.KARTYA);

        // Készpénz: 5 Ft-ra kerekítés (666.5 -> 665.0 a CR0 szabály
szerint)
        assertEquals(kerekites5re(666.5), cash.getAr(), 0.001);

        // Kártya: NINCS 0.5% bónusz, csak 10 fillér kerekítés -> 666.5
marad 666.5
        assertEquals(kerekites10Fillerre(666.5), card.getAr(), 0.0001);

        // Ellenőrizzük, hogy tényleg nincs 0.5% levonás (az 666.5 * 0.995
= 663.1675 lenne, ami NEM egyezik)
        assertNotEquals(kerekites10Fillerre(666.5 * 0.995), card.getAr(),
0.0001);
    }

    // 9) Nagy kosár: sok zacskó és megfelelő számú ajándékkupon együtt
    @Test
    void teszt_cr7_9_sok_zacskok_es_ajandekkuponok() {
        // 123.4 kg összsúly (pl. 60.2 alma + 63.2 banán) ->
floor(123.4/5)=24 zacskó
        Kosar kosar = new Kosar(List.of(
                new Tetel(Termek.ALMA, 60.2),
                new Tetel(Termek.BANAN, 63.2)
        ));
        ArInfo ar = target.getKosarAr(kosar, normal, List.of(),
FizetesiMod.KESZPENZ);
        assertEquals(24, ar.getAjandekZacskokSzama());
```

```
        // Kupon darabszám: fizetendő/20000 lefelé egészítve
        int db = (int) Math.floor(ar.getAr() / 20000.0);
        assertEquals(db, ar.getAjandekKuponok().size());

assertTrue(ar.getAjandekKuponok().stream().allMatch(ENGED_ajandek::contains
));
    }

    // 10) Vegyes: ULTRAMAX + nagy fizikai súly
    // Végösszeg lehet alacsony (emiatt 0 ajándékkupon), de zacskó jár a
fizikai súlyra.
    @Test
    void teszt_cr7_10_ultramax_es_sok_suly() {
        Kosar kosar = new Kosar(List.of(
                new Tetel(Termek.ALMA, 8.0),    // 8*500=4000 -> -10%
(>=5kg) = 3600
                new Tetel(Termek.BANAN, 7.0)    // 7*450=3150 -> -10%
(>=2kg) = 2835
        ));
        // Össz: 6435; két ULTRAMAX -> első: 4435 -> második: 2435 ->
készpénz 5 Ft kerek.: 2435
        ArInfo ar = target.getKosarAr(kosar, normal, List.of("KUPON-2000-
ULTRAMAX", "KUPON-2000-ULTRAMAX"), FizetesiMod.KESZPENZ);
        assertTrue(ar.getAr() >= 0.0);
        // Fizikai súly: 8 + 7 = 15 kg -> 3 zacskó
        assertEquals(3, ar.getAjandekZacskokSzama());
        // Ajándékkupon csak akkor, ha >= 20000 -> itt nincs
        assertEquals(0, ar.getAjandekKuponok().size());
    }
}
```