CR6 - A kártyás fizetés bevezetése

Bevezető

A JóÁr áruház ötletládája egy feneketlen zsákhoz hasonlít: csak gyűlnek és gyűlnek benne az üzenetek. A legtöbb panasz így szól: *„Miért nem lehet kártyával fizetni?!”*
A vezetőség végül meghozta a döntést: **igen, legyen kártyás fizetés is!** Sőt, hogy a vásárlók még jobban örüljenek, kártyás fizetés esetén extra kedvezmény is jár.

Feladat

Az Ön feladata, hogy az eddigi megoldást a **CR6**-nak megfelelően módosítsa!

Érdemes lehet az eddigi CR-ek egységtesztjeit átalakítani a követelményekből adódó új felületnek megfelelően.
Javasoljuk, hogy használja az alfejezet végén lévő egységteszteket, illetve a hangulatvezérelt kódolás eszközeit a feladat megoldásához!

Követelmények

A `getKosarAr` metódus szignatúrája mostantól négy paramétert fogad:

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

A `FizetesiMod` kétféle lehet:

- **készpénzes**
- **kártyás**

Eddig csak a készpénzes fizetés volt elfogadott, de most már támogatni kell a kártyás fizetést is.

A kártyás fizetésre az alábbi követelmények vonatkoznak:

1. **Kártyás fizetésre nem érvényes** a készpénzes fizetésnél leírt **5 forintra kerekítés** szabálya.
2. **Kártyás fizetésnél** a végső összegből adunk még **0,5% kedvezményt**.
    - Ez a *kártyabónusz*, amely a későbbiekben paraméterezhető lehet.
    - A kedvezményt a **végső összegből** kell levonni, tehát miután **minden kedvezmény és kupon** érvényesült.
3. A kártyás kedvezmény levonása után a fizetendő összeget **10 fillérre kell kerekíteni** az alábbi szabályok szerint:
    - **0,0–4,9 fillér között** → lefelé a legközelebbi 10 fillérre
    - **5,0–9,9 fillér között** → felfelé a legközelebbi 10 fillérre

Példák és egységtesztek

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.BeforeAll;
```

```java
import org.junit.jupiter.api.Test;
import java.util.List;
import org.aruhaz.*;

class AruhazCR6Tesztek {

    static Aruhaz target;
    static Idoszak normal;

    @BeforeAll
    static void init() {
        target = new Aruhaz();
        normal = new Idoszak("Normál");
        // A CR0/CR1 szerinti normál időszak beállításai
        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-ra kerekítés (készpénz)
    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)
    // 0,0–4,9 fillér → lefelé; 5,0–9,9 fillér → felfelé
    private double kerekites10Fillérre(double osszeg) {
        // 1 Ft = 100 fillér, 10 fillér = 0.1 Ft
        double tizedFt = Math.round(osszeg * 10.0); // először közelítünk
tized Ft pontosságra
        // De a specifikáció szerint 5.0–9.9 fillér esetén felfelé kell
kerekíteni a legközelebbi 10 fillérre.
        // Ez pontosan megfelel a "tized Ft-re" kerekítésnek.
        return Math.round(osszeg * 10.0) / 10.0;
    }

    // 1) Készpénz: viselkedés változatlan, 5 Ft-os kerekítés (1 kg alma)
    @Test
    void teszt_cr6_keszpenz_valtozatlan_1kg_alma() {
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA, 1.0))); //
500 Ft
        ArInfo ar = target.getKosarAr(kosar, normal, List.of(),
FizetesiMod.KESZPENZ);
        assertEquals(kerekites5re(500.0), ar.getAr(), 0.001); // 500
        assertEquals(List.of(), ar.getFelNemHasznaltKuponok());
    }

    // 2) Kártya: nincs 5 Ft-os kerekítés, 0.5% kedvezmény a végén, majd 10
fillérre kerekítés (1 kg alma)
    @Test
    void teszt_cr6_kartya_1kg_alma_bonusz_es_10filler_kerekites() {
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA, 1.0))); //
500 Ft
```

```
        // Kártyabónusz a VÉGÖSSZEGEN: 500 * 0.995 = 497.5 Ft → 10 fillérre
kerekítve: 497.5 Ft (már tized Ft)
        ArInfo ar = target.getKosarAr(kosar, normal, List.of(),
FizetesiMod.KARTYA);
        assertEquals(kerekites10Fillérre(500.0 * 0.995), ar.getAr(),
0.0001); // 497.5
        assertEquals(List.of(), ar.getFelNemHasznaltKuponok());
    }

    // 3) Kártya + mennyiségi kedvezmény (2 kg banán → 10% kedvezmény) +
0.5% bónusz a végén
    @Test
    void teszt_cr6_kartya_mennyisegi_kedvezmeny_utan_bonusz() {
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.BANAN, 2.0))); //
900 Ft → 10% = 810 Ft
        double utanKedv = 810.0;
        double kartya = utanKedv * 0.995; // 810 * 0.995 = 805.95
        ArInfo ar = target.getKosarAr(kosar, normal, List.of(),
FizetesiMod.KARTYA);
        assertEquals(kerekites10Fillérre(kartya), ar.getAr(), 0.0001); //
806.0 Ft
    }

    // 4) Kártya: kerekítés 10 fillérre – lefelé (példa: 1.789 kg alma)
    @Test
    void teszt_cr6_kartya_10filler_lefele() {
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA, 1.789))); //
1.789*500 = 894.5
        double utanKuponok = 894.5; // nincs mennyiségi kedvezmény
        double kartya = utanKuponok * 0.995; // 894.5 - 0.5% = 890.0275 →
2.75 fillér → lefelé 890.0
        ArInfo ar = target.getKosarAr(kosar, normal, List.of(),
FizetesiMod.KARTYA);
        assertEquals(kerekites10Fillérre(kartya), ar.getAr(), 0.0001); //
890.0
    }

    // 5) Kártya: kerekítés 10 fillérre – felfelé (példa: 1.5 kg alma)
    @Test
    void teszt_cr6_kartya_10filler_felfele() {
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA, 1.5))); //
750.0
        double kartya = 750.0 * 0.995; // 746.25 → 25 fillér (5.0–9.9 →
felfelé) → 746.3
        ArInfo ar = target.getKosarAr(kosar, normal, List.of(),
FizetesiMod.KARTYA);
        assertEquals(kerekites10Fillérre(kartya), ar.getAr(), 0.0001); //
746.3
    }

    // 6) Kártya + kupon: A10 almára, utána 0.5% bónusz, majd 10 fillérre
kerekítés
    @Test
    void teszt_cr6_kartya_kuponok_utan_bonusz() {
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA, 1.0))); //
500 Ft
        // A10 → 450.0, kártyabónusz: 450 * 0.995 = 447.75 → 10 fillérre:
447.8
        ArInfo ar = target.getKosarAr(kosar, normal, List.of("A10"),
FizetesiMod.KARTYA);
        assertEquals(kerekites10Fillérre(450.0 * 0.995), ar.getAr(),
```

```
0.0001); // 447.8
        assertEquals(List.of(), ar.getFelNemHasznaltKuponok());
    }

    // 7) Kártya + X10 (nem összevonható) + más kuponok: csak X10
érvényesül, utána 0.5% bónusz
    @Test
    void teszt_cr6_kartya_x10_nem_osszevonhato_bonusz_utan() {
        Kosar kosar = new Kosar(List.of(
                new Tetel(Termek.ALMA, 2.0), // 1000
                new Tetel(Termek.BANAN, 1.0) // 450
        )); // össz: 1450
        // X10 → 1305.0; kártyabónusz: 1305 * 0.995 = 1298.475 → 10
fillérre: 1298.5
        ArInfo ar = target.getKosarAr(kosar, normal, List.of("X10", "A5",
"B5"), FizetesiMod.KARTYA);
        assertEquals(kerekites10Fillérre(1450.0 * 0.90 * 0.995),
ar.getAr(), 0.0001); // 1298.5
        // A nem összevonható kuponok visszajárnak X10 mellett
        assertEquals(List.of("A5", "B5"), ar.getFelNemHasznaltKuponok());
    }

    // 8) Összehasonlítás: ugyanaz a kosár készpénzzel vs. kártyával (A10
almára)
    @Test
    void teszt_cr6_keszpenz_vs_kartya_azonos_kosar() {
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA, 1.0))); //
500
        // Készpénz + A10: 450 → 5 Ft-os kerekítés: 450
        ArInfo cash = target.getKosarAr(kosar, normal, List.of("A10"),
FizetesiMod.KESZPENZ);
        assertEquals(kerekites5re(450.0), cash.getAr(), 0.001); // 450.0

        // Kártya + A10: 450 → 0.5% = 447.75 → 10 fillérre: 447.8
        ArInfo card = target.getKosarAr(kosar, normal, List.of("A10"),
FizetesiMod.KARTYA);
        assertEquals(kerekites10Fillérre(450.0 * 0.995), card.getAr(),
0.0001); // 447.8
    }

    // 9) Kártya: nagyobb kosár mennyiségi kedvezménnyel (banán 3 kg →
10%), majd bónusz és 10 fillérre kerekítés
    @Test
    void teszt_cr6_kartya_nagyobb_kosar_mennyisegi_kedvezmeny() {
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.BANAN, 3.0))); //
1350 → 10% = 1215
        double kartya = 1215.0 * 0.995; // 1208.925 → 10 fillérre: 1208.9
(mivel 92.5 fillér → 90)
        ArInfo ar = target.getKosarAr(kosar, normal, List.of(),
FizetesiMod.KARTYA);
        assertEquals(kerekites10Fillérre(kartya), ar.getAr(), 0.0001); //
1208.9
    }

    // 10) Kártya + "ingyen" típusú kupon (pl. A-FREE1), a bónusz a végső
összegből jön le
    @Test
    void teszt_cr6_kartya_free_kupon_utan_bonusz() {
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA, 0.5))); //
0.5 kg alma
        // A-FREE1 → 0 Ft; kártyabónusz a végső összegen → 0 Ft; 10 fillér
```

```
kerekítés → 0.0 Ft
        ArInfo ar = target.getKosarAr(kosar, normal, List.of("A-FREE1"),
FizetesiMod.KARTYA);
        assertEquals(0.0, ar.getAr(), 0.0001);
        assertEquals(List.of(), ar.getFelNemHasznaltKuponok());
    }
}
```