CR1 – Időszakok és időszakfüggő akciók kezelése

Ahogy arra számítani lehetett, itt a CR1.
Az Ön feladata, hogy az eddigi megoldást a CR1-nek megfelelően módosítsa. A JóÁr áruház vezetősége bevezette az **időszakok** fogalmát, amelyekben **különböző alapárak és mennyiségi kedvezmények** érvényesek.

Metódus módosítás

A `getKosárÁr` metódus szignatúrája mostantól két paramétert fogad:

```
double getKosárÁr(Kosár kosár, Időszak időszak)
```

Az új paraméter, az **időszak**, határozza meg az adott vásárláskor érvényes árakat és akciókat.

Időszakok

A JóÁr áruház többféle időszakot kezelhet. A rendszerben tárolni és konfigurálni kell az időszakokat, valamint az ezekhez tartozó akciókat.

A jelenlegi példában három időszak van:

1. Normál időszak (alapállapot, megegyezik az eredeti alapfeladattal)

- **Alapárak:**
  - Alma: 500 Ft/kg
  - Banán: 450 Ft/kg
- **Mennyiségi kedvezmények:**
  - Alma:
    - Legalább 5 kg vásárlása esetén → 10% kedvezmény
    - Legalább 20 kg vásárlása esetén → 15% kedvezmény
  - Banán:
    - Legalább 2 kg vásárlása esetén → 10% kedvezmény

Vegyük észre, hogy a normál időszak alapára és mennyiségi kedvezményei pontosan megegyeznek az alapfeladatban (CR0) megadott érékekkel. Ez nem véletlen! Ez direkt van így!

2. Tavaszi akciós időszak

- **Alapárak:**
  - Alma: 600 Ft/kg
  - Banán: 550 Ft/kg
- **Mennyiségi kedvezmények:**
  - Alma:
    - Legalább 2 kg vásárlása esetén → 15% kedvezmény
    - Legalább 5 kg vásárlása esetén → 20% kedvezmény

    - Banán:
        - Legalább 4 kg vásárlása esetén → 20% kedvezmény
        - Legalább 7.5 kg vásárlása esetén → 25% kedvezmény

3. Téli akciós időszak

- **Alapárak:**
    - Alma: 400 Ft/kg
    - Banán: 400 Ft/kg
- **Mennyiségi kedvezmények:**
    - Alma:
        - Legalább 10 kg vásárlása esetén → 5% kedvezmény
        - Legalább 20 kg vásárlása esetén → 10% kedvezmény
    - Banán:
        - Legalább 5 kg vásárlása esetén → 10% kedvezmény

Fejlesztési követelmények

- Az **időszakokat és a hozzájuk tartozó árakat, valamint akciós szabályokat paraméterezhetővé** kell tenni, azaz a JóÁr áruház később is bővíthesse vagy módosíthassa az időszakok listáját és azok tartalmát.
- A `getKosárÁr` metódus működésének minden esetben az adott időszakhoz tartozó adatok alapján kell számolnia az árakat és kedvezményeket.
- Az akciók **mennyiségi határokhoz kötöttek**, és **termék-specifikusak** (külön alma és külön banán esetén).

Megjegyzés

A jelenlegi három időszak (normál, tavaszi, téli) csak **példaként szolgál**, a jövőben az áruház más időszakokat is bevezethet (pl. őszi leárazás, nyári kiárusítás stb.). A rendszernek készen kell állnia ezek fogadására és kezelésére.

Elképzelhető, hogy a JóÁr áruház vezetősége további CR-eket fog kérni a jövőben!

Hogy még világosabb legyen feladat, megadjuk a feladatnak megfelelő unit teszteket is:

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

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

class AruhazCR1Tesztek {
    static Aruhaz target;
    static Idoszak normal, tavaszi, teli;
    static double almaEgysegArNormal = 500.0;
    static double bananEgysegArNormal = 450.0;
    static double almaHatar1Normal = 5.0;
```

```
    static double almaKedvezmeny1Normal = 0.1;
    static double almaHatar2Normal = 20.0;
    static double almaKedvezmeny2Normal = 0.15;
    static double bananHatar1Normal = 2.0;
    static double bananKedvezmeny1Normal = 0.1;
    static double almaEgysegArTavaszi = 600.0;
    static double bananEgysegArTavaszi = 550.0;
    static double almaHatar1Tavaszi = 2.0;
    static double almaKedvezmeny1Tavaszi = 0.15;
    static double almaHatar2Tavaszi = 5.0;
    static double almaKedvezmeny2Tavaszi = 0.20;
    static double bananHatar1Tavaszi = 4.0;
    static double bananKedvezmeny1Tavaszi = 0.20;
    static double bananHatar2Tavaszi = 7.5;
    static double bananKedvezmeny2Tavaszi = 0.25;
    static double almaEgysegArTeli = 400.0;
    static double bananEgysegArTeli = 400.0;
    static double almaHatar1Teli = 10.0;
    static double almaKedvezmeny1Teli = 0.05;
    static double almaHatar2Teli = 20.0;
    static double almaKedvezmeny2Teli = 0.10;
    static double bananHatar1Teli = 5.0;
    static double bananKedvezmeny1Teli = 0.10;
    @BeforeAll
    public static void initAruhaz() {
        target = new Aruhaz();
        normal = new Idoszak("Normál");
        normal.setEgysegAr(Termek.ALMA, almaEgysegArNormal);
        normal.setEgysegAr(Termek.BANAN, bananEgysegArNormal);
        normal.setKedvezmeny(Termek.ALMA, almaHatar1Normal,
almaKedvezmeny1Normal);
        normal.setKedvezmeny(Termek.ALMA, almaHatar2Normal,
almaKedvezmeny2Normal);
        normal.setKedvezmeny(Termek.BANAN, bananHatar1Normal,
bananKedvezmeny1Normal);
        target.addIdoszak(normal);
        tavaszi = new Idoszak("Tavaszi");
        tavaszi.setEgysegAr(Termek.ALMA, almaEgysegArTavaszi);
        tavaszi.setEgysegAr(Termek.BANAN, bananEgysegArTavaszi);
        tavaszi.setKedvezmeny(Termek.ALMA, almaHatar1Tavaszi,
almaKedvezmeny1Tavaszi);
        tavaszi.setKedvezmeny(Termek.ALMA, almaHatar2Tavaszi,
almaKedvezmeny2Tavaszi);
        tavaszi.setKedvezmeny(Termek.BANAN, bananHatar1Tavaszi,
bananKedvezmeny1Tavaszi);
        tavaszi.setKedvezmeny(Termek.BANAN, bananHatar2Tavaszi,
bananKedvezmeny2Tavaszi);
        target.addIdoszak(tavaszi);
        teli = new Idoszak("Téli");
        teli.setEgysegAr(Termek.ALMA, almaEgysegArTeli);
        teli.setEgysegAr(Termek.BANAN, bananEgysegArTeli);
        teli.setKedvezmeny(Termek.ALMA, almaHatar1Teli,
almaKedvezmeny1Teli);
        teli.setKedvezmeny(Termek.ALMA, almaHatar2Teli,
almaKedvezmeny2Teli);
        teli.setKedvezmeny(Termek.BANAN, bananHatar1Teli,
bananKedvezmeny1Teli);
        target.addIdoszak(teli);
    }
    @Test
    void teszt_cr1_normal_alma_hatar1() {
```

```
        Idoszak idoszak = normal;
        Termek termek = Termek.ALMA;
        double egysegAr = almaEgysegArNormal;
        double mennyiseg = almaHatar1Normal;
        double kedvezmeny = almaKedvezmeny1Normal;
        double expected = kerekites5re(
                egysegAr * mennyiseg * (1 - kedvezmeny));
        Kosar kosar = new Kosar(List.of(new Tetel(termek, mennyiseg)));
        double actual = target.getKosarAr(kosar, idoszak);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr1_normal_alma_hatar2() {
        Idoszak idoszak = normal;
        Termek termek = Termek.ALMA;
        double egysegAr = almaEgysegArNormal;
        double mennyiseg = almaHatar2Normal;
        double kedvezmeny = almaKedvezmeny2Normal;
        double expected = kerekites5re(egysegAr * mennyiseg * (1 -
kedvezmeny));
        Kosar kosar = new Kosar(List.of(new Tetel(termek, mennyiseg)));
        double actual = target.getKosarAr(kosar, idoszak);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr1_normal_banan_hatar1() {
        Idoszak idoszak = normal;
        Termek termek = Termek.BANAN;
        double egysegAr = bananEgysegArNormal;
        double mennyiseg = bananHatar1Normal;
        double kedvezmeny = bananKedvezmeny1Normal;
        double expected = kerekites5re(egysegAr * mennyiseg * (1 -
kedvezmeny));
        Kosar kosar = new Kosar(List.of(new Tetel(termek, mennyiseg)));
        double actual = target.getKosarAr(kosar, idoszak);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr1_tavaszi_alma_hatar1() {
        Idoszak idoszak = tavaszi;
        Termek termek = Termek.ALMA;
        double egysegAr = almaEgysegArTavaszi;
        double mennyiseg = almaHatar1Tavaszi;
        double kedvezmeny = almaKedvezmeny1Tavaszi;
        double expected = kerekites5re(egysegAr * mennyiseg * (1 -
kedvezmeny));
        Kosar kosar = new Kosar(List.of(new Tetel(termek, mennyiseg)));
        double actual = target.getKosarAr(kosar, idoszak);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr1_tavaszi_alma_hatar2() {
        Idoszak idoszak = tavaszi;
        Termek termek = Termek.ALMA;
        double egysegAr = almaEgysegArTavaszi;
        double mennyiseg = almaHatar2Tavaszi;
        double kedvezmeny = almaKedvezmeny2Tavaszi;
        double expected = kerekites5re(egysegAr * mennyiseg * (1 -
kedvezmeny));
        Kosar kosar = new Kosar(List.of(new Tetel(termek, mennyiseg)));
        double actual = target.getKosarAr(kosar, idoszak);
```

```
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr1_tavaszi_banan_hatar1() {
        Idoszak idoszak = tavaszi;
        Termek termek = Termek.BANAN;
        double egysegAr = bananEgysegArTavaszi;
        double mennyiseg = bananHatar1Tavaszi;
        double kedvezmeny = bananKedvezmeny1Tavaszi;
        double expected = kerekites5re(egysegAr * mennyiseg * (1 -
kedvezmeny));
        Kosar kosar = new Kosar(List.of(new Tetel(termek, mennyiseg)));
        double actual = target.getKosarAr(kosar, idoszak);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr1_tavaszi_banan_hatar2() {
        Idoszak idoszak = tavaszi;
        Termek termek = Termek.BANAN;
        double egysegAr = bananEgysegArTavaszi;
        double mennyiseg = bananHatar2Tavaszi;
        double kedvezmeny = bananKedvezmeny2Tavaszi;
        double expected = kerekites5re(egysegAr * mennyiseg * (1 -
kedvezmeny));
        Kosar kosar = new Kosar(List.of(new Tetel(termek, mennyiseg)));
        double actual = target.getKosarAr(kosar, idoszak);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr1_teli_alma_hatar1() {
        Idoszak idoszak = teli;
        Termek termek = Termek.ALMA;
        double egysegAr = almaEgysegArTeli;
        double mennyiseg = almaHatar1Teli;
        double kedvezmeny = almaKedvezmeny1Teli;
        double expected = kerekites5re(egysegAr * mennyiseg * (1 -
kedvezmeny));
        Kosar kosar = new Kosar(List.of(new Tetel(termek, mennyiseg)));
        double actual = target.getKosarAr(kosar, idoszak);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr1_teli_alma_hatar2() {
        Idoszak idoszak = teli;
        Termek termek = Termek.ALMA;
        double egysegAr = almaEgysegArTeli;
        double mennyiseg = almaHatar2Teli;
        double kedvezmeny = almaKedvezmeny2Teli;
        double expected = kerekites5re(egysegAr * mennyiseg * (1 -
kedvezmeny));
        Kosar kosar = new Kosar(List.of(new Tetel(termek, mennyiseg)));
        double actual = target.getKosarAr(kosar, idoszak);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr1_teli_banan_hatar1() {
        Idoszak idoszak = teli;
        Termek termek = Termek.BANAN;
        double egysegAr = bananEgysegArTeli;
        double mennyiseg = bananHatar1Teli;
        double kedvezmeny = bananKedvezmeny1Teli;
```

```
        double expected = kerekites5re(egysegAr * mennyiseg * (1 -
kedvezmeny));
        Kosar kosar = new Kosar(List.of(new Tetel(termek, mennyiseg)));
        double actual = target.getKosarAr(kosar, idoszak);
        assertEquals(expected, actual, 0.001);
    }
    private double kerekites5re(double osszeg) {
        double maradek = osszeg % 10.0;
        if (maradek < 2.5) {
            return osszeg - maradek;
        } else if (maradek < 5.0) {
            return osszeg - maradek + 5.0;
        } else if (maradek < 7.5) {
            return osszeg - maradek + 5.0;
        } else {
            return osszeg - maradek + 10.0;
        }
    }
}
```