## Alapfeladat – Kassza program a JóÁr áruházban

**Az Ön feladata az lesz, hogy elkészítse a JóÁr áruház kasszarendszerének alapverzióját.** A kasszarendszer központi eleme a következő metódus:

```
double getKosárÁr(Kosár kosár)
```

Ez a metódus egy vásárlás végösszegét számítja ki forintban. A visszatérési érték tehát egy **szám**, amely a vásárlás során fizetendő végső összeget jelenti **készpénzes fizetés esetén**.

A feladat során az alábbi szabályokat kell figyelembe venni:

---

### Kosár

A kosár kétféle terméket tartalmazhat:

- **alma**, kilogrammban megadva (pl. 1.5 kg),
- **banán**, kilogrammban megadva (pl. 3 kg),
- az egyes termék többször is benne lehetnek a kosárban, ilyenkor össze kell adni az alma mennyiséget, illetve a banán mennyiséget.

---

### Alapárak

A termékek **alapárai** a következők:

- **1 kg alma → 500 Ft**
- **1 kg banán → 450 Ft**

---

### Mennyiségi kedvezmények

A JóÁr áruház normál időszakában a következő kedvezmények érvényesek:

**Alma:**

- Legalább **5 kg** vásárlása esetén → **10%** kedvezmény az alma összárára.
- Legalább **20 kg** vásárlása esetén → **15%** kedvezmény az alma összárára.

**Banán:**

- Legalább **2 kg** vásárlása esetén → **10%** kedvezmény a banán összárára.

---

## Fizetés

Jelenleg a rendszer **csak készpénzes fizetést** támogat.

---

## Kerekítés szabályai (készpénzes fizetés esetén)

A végösszeget **5 forintra kell kerekíteni**, a következő szabályok szerint:

- Ha az összeg 0,01–2,49 Ft-ra végződik → **lefelé**, a legközelebbi 0-ra.
- Ha az összeg 2,50–4,99 Ft-ra végződik → **felfelé**, a legközelebbi 5-re.
- Ha az összeg 5,01–7,49 Ft-ra végződik → **lefelé**, a legközelebbi 5-re.
- Ha az összeg 7,50–9,99 Ft-ra végződik → **felfelé**, a legközelebbi 0-ra.

Például:

- 2782 Ft → 2780 Ft
- 2783 Ft → 2785 Ft
- 2787 Ft → 2785 Ft
- 2789 Ft → 2790 Ft

Íme néhány további példa kosártartalomra, a dokumentumban leírt szabályok alapján (alapárak, kedvezmények, kerekítés):

---

## Példák kosárra és az árakra

### 1. Kosár: 1 kg alma

- Ár: 1 × 500 Ft = 500 Ft
- Kedvezmény: nincs
- Kerekítés: nincs szükséges
- **Végösszeg: 500 Ft**

---

### 2. Kosár: 2 kg alma

- Ár: 2 × 500 Ft = 1000 Ft
- Kedvezmény: nincs
- Kerekítés: nincs szükséges
- **Végösszeg: 1000 Ft**

---

### 3. Kosár: 5 kg alma

- Ár: 5 × 500 Ft = 2500 Ft

- Kedvezmény: 10% → 250 Ft
- Kedvezményes ár: 2250 Ft
- Kerekítés: 2250 Ft → **2250 Ft**
- **Végösszeg: 2250 Ft**

---

**4. Kosár: 20 kg alma**

- Ár: 20 × 500 Ft = 10000 Ft
- Kedvezmény: 15% → 1500 Ft
- Kedvezményes ár: 8500 Ft
- Kerekítés: nincs szükséges
- **Végösszeg: 8500 Ft**

---

**5. Kosár: 1 kg alma, 1 kg banán**

- Alma: 500 Ft (nincs kedvezmény)
- Banán: 450 Ft (nincs kedvezmény)
- Összesen: 950 Ft
- Kerekítés: 950 Ft → **950 Ft**
- **Végösszeg: 950 Ft**

---

**6. Kosár: 2 kg banán**

- Ár: 2 × 450 Ft = 900 Ft
- Kedvezmény: 10% → 90 Ft
- Kedvezményes ár: 810 Ft
- Kerekítés: 810 Ft → **810 Ft**
- **Végösszeg: 810 Ft**

---

**7. Kosár: 1 kg alma, 2 kg alma, 2 kg alma**

- Összesen: 1 + 2 + 2 = 5 kg alma
- Ár: 5 × 500 Ft = 2500 Ft
- Kedvezmény: 10% → 250 Ft
- Kedvezményes ár: 2250 Ft
- Kerekítés: 2250 Ft → **2250 Ft**
- **Végösszeg: 2250 Ft**

---

**8. Kosár: 2 kg alma, 2 kg banán**

- Alma: 2 × 500 = 1000 Ft (nincs kedvezmény)
- Banán: 2 × 450 = 900 Ft → 10% kedvezmény = 810 Ft
- Összesen: 1000 + 810 = 1810 Ft
- Kerekítés: 1810 Ft → **1810 Ft**
- **Végösszeg: 1810 Ft**

---

## 9. Kosár: 1,01 kg alma

- Ár: 1,01 × 500 = **505,00 Ft**
- Kedvezmény: nincs
- **Kerekítés: 505,00 → 505 Ft** (nincs változás)
- **Végösszeg: 505 Ft**

---

## 10. Kosár: 1,99 kg banán

- Ár: 1,99 × 450 = **895,50 Ft**
- Kedvezmény: nincs
- **Kerekítés: 895,50 → 895 Ft** (Ha az összeg 5,01–7,49 Ft-ra végződik → **lefelé**, a legközelebbi 5-re.)
- **Végösszeg: 895 Ft**

---

## 11. Kosár: 1,789 kg alma

- Ár: 1,789 × 500 = **894,50 Ft**
- Kedvezmény: nincs
- **Kerekítés: 894,50 → 895 Ft (Ha az összeg 2,50-4,99 Ft-ra végződik → felfelé, a legközelebbi 5-re.)**
- **Végösszeg: 895 Ft**

---

## 12. Kosár: 2,01 kg banán

- Ár: 2,01 × 450 = **904,50 Ft**
- Kedvezmény: 10% → 90,45 Ft
- Kedvezményes ár: 904,50 − 90,45 = **814,05 Ft**
- **Kerekítés: 814,05 → 815 Ft (Ha az összeg 2,50-4,99 Ft-ra végződik → felfelé, a legközelebbi 5-re.)**
- **Végösszeg: 815 Ft**

---

## 13. Kosár: 3,333 kg alma + 1,777 kg banán

- Alma: 3,333 × 500 = **1666,50 Ft** (nincs kedvezmény)
- Banán: 1,777 × 450 = **799,65 Ft** (nincs kedvezmény)
- Összesen: 1666,50 + 799,65 = **2466,15 Ft**
- **Kerekítés: 2466,15 → 2465 Ft** (Ha az összeg 5,01–7,49 Ft-ra végződik → **lefelé**, a legközelebbi 5-re.)
- **Végösszeg: 2465 Ft**

---

## 14. Kosár: 5,001 kg alma

- Ár: 5,001 × 500 = **2500,50 Ft**
- Kedvezmény: 10% → 250,05 Ft
- Kedvezményes ár: **2250,45 Ft**
- **Kerekítés: 2250,45 → 2250 Ft** (Ha az összeg 0,01–2,49 Ft-ra végződik → **lefelé**, a legközelebbi 0-ra.)
- **Végösszeg: 2250 Ft**

---

### 15. Kosár: 3,789 kg alma és 2,777 kg banán

- Alma: 3,789 × 500 = **1894,50 Ft**
- Banán: 2,777 × 450 = **1249,65 Ft**
- Kedvezmény: banánra 10% (2,777 > 2 kg)
- Kedvezményes banán ár: **1124,685 Ft**
- Összesen: 1894,50 + 1124,68 = **3019,185 Ft**
- **Kerekítés: 3019,185 → 3020 Ft** (Ha az összeg 7,50–9,99 Ft-ra végződik → **felfelé**, a legközelebbi 0-ra.)
- **Végösszeg: 3020 Ft**

---

A későbbiekben **változhatnak a rendszer követelményei**, például új fizetési módok, kuponok, vagy időszakok bevezetésével. Ezeket külön dokumentáljuk majd, **Change Request** (röviden: **CR**) formájában.

Hogy még konkrétabb legyen a feladat megadjuk a példáknak megfelelő unit teszteket JUnit5 szintaxist használva.

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import java.util.List;
import org.aruhaz.Aruhaz;
import org.aruhaz.Kosar;
import org.aruhaz.Tetel;
import org.aruhaz.Termek;
import org.aruhaz.TermekAr;

class AruhazTesztek {
    @Test
    void teszt_cr0_pelda1_1kgAlma() {
```

```
        double egysegAr = 500.0;
        double mennyiseg = 1.0;
        Aruhaz target = new Aruhaz(Termek.ALMA, egysegAr);
        double expected = egysegAr * mennyiseg;
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA,
mennyiseg)));
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr0_pelda2_2kgAlma() {
        double egysegAr = 500.0;
        double mennyiseg = 2.0;
        Aruhaz target = new Aruhaz(Termek.ALMA, egysegAr);
        double expected = egysegAr * mennyiseg;
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA,
mennyiseg)));
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr0_pelda3_5kgAlma_kedvezmennyel() {
        double egysegAr = 500.0;
        double hatar1 = 5.0;
        double kedvezmeny1 = 0.1;
        double mennyiseg = hatar1;
        Aruhaz target = new Aruhaz(Termek.ALMA, egysegAr);
        target.setKedvezmeny(Termek.ALMA, hatar1, kedvezmeny1);
        double expected = egysegAr * mennyiseg * (1.0-kedvezmeny1);
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA,
mennyiseg)));
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr0_pelda4_20kgAlma_kedvezmennyel() {
        double egysegAr = 500.0;
        double hatar1 = 5.0;
        double kedvezmeny1 = 0.1;
        double hatar2 = 20.0;
        double kedvezmeny2 = 0.15;
        double mennyiseg = hatar2;
        Aruhaz target = new Aruhaz(Termek.ALMA, egysegAr);
        target.setKedvezmeny(Termek.ALMA, hatar1, kedvezmeny1);
        target.setKedvezmeny(Termek.ALMA, hatar2, kedvezmeny2);
        double expected = egysegAr * mennyiseg * (1.0-kedvezmeny2);
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA,
mennyiseg)));
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr0_pelda5_1kgAlma_1kgBanan() {
        double egysegArAlma = 500.0;
        double mennyisegAlma = 1.0;
        double egysegArBanan = 450.0;
        double mennyisegBanan = 1.0;
        Aruhaz target = new Aruhaz(List.of(
                new TermekAr(Termek.ALMA, egysegArAlma),
                new TermekAr(Termek.BANAN, egysegArBanan)
        ));
```

```
        Kosar kosar = new Kosar(List.of(
                new Tetel(Termek.ALMA, mennyisegAlma),
                new Tetel(Termek.BANAN, mennyisegBanan)
        ));
        double expected = egysegArAlma*mennyisegAlma +
                egysegArBanan*mennyisegBanan;
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr0_pelda6_2kgBanan_kedvezmennyel() {
        double egysegAr = 450.0;
        double hatar1 = 2.0;
        double kedvezmeny1 = 0.1;
        double mennyiseg = hatar1;
        Aruhaz target = new Aruhaz(Termek.BANAN, egysegAr);
        target.setKedvezmeny(Termek.BANAN, hatar1, kedvezmeny1);
        double expected = egysegAr * mennyiseg * (1.0-kedvezmeny1);
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.BANAN,
mennyiseg)));
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr0_pelda7_tobbAlma_osszesen4kg() {
        double egysegAr = 500.0;
        double mennyiseg1 = 1.0;
        double mennyiseg2 = 2.0;
        double mennyiseg3 = 2.0;
        double osszesMennyiseg = mennyiseg1 + mennyiseg2 + mennyiseg3;
        Aruhaz target = new Aruhaz(Termek.ALMA, egysegAr);
        double hatar1 = 5.0;
        double kedvezmeny1 = 0.1;
        target.setKedvezmeny(Termek.ALMA, hatar1, kedvezmeny1);
        Kosar kosar = new Kosar(List.of(
                new Tetel(Termek.ALMA, mennyiseg1),
                new Tetel(Termek.ALMA, mennyiseg2),
                new Tetel(Termek.ALMA, mennyiseg3)
        ));
        double expected = egysegAr * osszesMennyiseg * (1.0-kedvezmeny1);
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr0_pelda8_2kgAlma_2kgBanan() {
        double egysegArAlma = 500.0;
        double egysegArBanan = 450.0;
        double kedvezmenyHatarBanan = 2.0;
        double kedvezmenyBanan = 0.1;
        double mennyisegAlma = 2.0;
        double mennyisegBanan = 2.0;
        Aruhaz target = new Aruhaz(List.of(
                new TermekAr(Termek.ALMA, egysegArAlma),
                new TermekAr(Termek.BANAN, egysegArBanan)
        ));
        target.setKedvezmeny(Termek.BANAN, kedvezmenyHatarBanan,
kedvezmenyBanan);
        double bruttoAlma = egysegArAlma * mennyisegAlma;
        double bruttoBanan = egysegArBanan * mennyisegBanan;
        double nettoBanan = bruttoBanan * (1.0 - kedvezmenyBanan);
        double expected = bruttoAlma + nettoBanan;
```

```
        Kosar kosar = new Kosar(List.of(
                new Tetel(Termek.ALMA, mennyisegAlma),
                new Tetel(Termek.BANAN, mennyisegBanan)
        ));
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr0_pelda9_1_01kgAlma() {
        double egysegAr = 500.0;
        double mennyiseg = 1.01;
        Aruhaz target = new Aruhaz(Termek.ALMA, egysegAr);
        double expected = egysegAr * mennyiseg;
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA,
mennyiseg)));
        double actual = target.getKosarAr(kosar);
        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;
        }
    }
    @Test
    void teszt_cr0_pelda10_1_99kgBanan() {
        double egysegAr = 450.0;
        double mennyiseg = 1.99;
        Aruhaz target = new Aruhaz(Termek.BANAN, egysegAr);
        double brutto = egysegAr * mennyiseg;
        double expected = kerekites5re(brutto);
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.BANAN,
mennyiseg)));
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr0_pelda11_1_789kgAlma() {
        double egysegAr = 500.0;
        double mennyiseg = 1.789;
        Aruhaz target = new Aruhaz(Termek.ALMA, egysegAr);
        double brutto = egysegAr * mennyiseg;
        double expected = kerekites5re(brutto);
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA,
mennyiseg)));
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr0_pelda12_2_01kgBanan_kedvezmennyel() {
        double egysegAr = 450.0;
        double mennyiseg = 2.01;
        double kedvezmenyHatar = 2.0;
        double kedvezmeny = 0.1;
        Aruhaz target = new Aruhaz(Termek.BANAN, egysegAr);
```

```
        target.setKedvezmeny(Termek.BANAN, kedvezmenyHatar, kedvezmeny);
        double brutto = egysegAr * mennyiseg;
        double netto = brutto * (1.0 - kedvezmeny);
        System.out.println(netto);
        double expected = kerekites5re(netto);
        System.out.println(expected);
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.BANAN,
mennyiseg)));
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr0_pelda13_3_333kgAlma_1_777kgBanan() {
        double egysegArAlma = 500.0;
        double egysegArBanan = 450.0;
        double mennyisegAlma = 3.333;
        double mennyisegBanan = 1.777;
        Aruhaz target = new Aruhaz(List.of(
                new TermekAr(Termek.ALMA, egysegArAlma),
                new TermekAr(Termek.BANAN, egysegArBanan)
        ));
        double bruttoAlma = egysegArAlma * mennyisegAlma;
        double bruttoBanan = egysegArBanan * mennyisegBanan;
        double brutto = bruttoAlma + bruttoBanan;
        double expected = kerekites5re(brutto);
        Kosar kosar = new Kosar(List.of(
                new Tetel(Termek.ALMA, mennyisegAlma),
                new Tetel(Termek.BANAN, mennyisegBanan)
        ));
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr0_pelda14_5_001kgAlma_kedvezmennyel() {
        double egysegAr = 500.0;
        double mennyiseg = 5.001;
        double kedvezmenyHatar = 5.0;
        double kedvezmeny = 0.1;
        Aruhaz target = new Aruhaz(Termek.ALMA, egysegAr);
        target.setKedvezmeny(Termek.ALMA, kedvezmenyHatar, kedvezmeny);
        double brutto = egysegAr * mennyiseg;
        double netto = brutto * (1.0 - kedvezmeny);
        double expected = kerekites5re(netto);
        Kosar kosar = new Kosar(List.of(new Tetel(Termek.ALMA,
mennyiseg)));
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
    @Test
    void teszt_cr0_pelda15_3_789kgAlma_2_777kgBanan() {
        double egysegArAlma = 500.0;
        double egysegArBanan = 450.0;
        double mennyisegAlma = 3.789;
        double mennyisegBanan = 2.777;
        double kedvezmenyHatarBanan = 2.0;
        double kedvezmenyBanan = 0.1;
        Aruhaz target = new Aruhaz(List.of(
                new TermekAr(Termek.ALMA, egysegArAlma),
                new TermekAr(Termek.BANAN, egysegArBanan)
        ));
        target.setKedvezmeny(Termek.BANAN, kedvezmenyHatarBanan,
```

```
kedvezmenyBanan);
        double bruttoAlma = egysegArAlma * mennyisegAlma;
        double bruttoBanan = egysegArBanan * mennyisegBanan;
        double nettoBanan = bruttoBanan * (1.0 - kedvezmenyBanan);
        double osszesen = bruttoAlma + nettoBanan;
        double expected = kerekites5re(osszesen);
        Kosar kosar = new Kosar(List.of(
                new Tetel(Termek.ALMA, mennyisegAlma),
                new Tetel(Termek.BANAN, mennyisegBanan)
        ));
        double actual = target.getKosarAr(kosar);
        assertEquals(expected, actual, 0.001);
    }
}
```