# CR5 – Introducing Multi-Product Coupons

---

## 1. Funny Introduction

“The combinable apple coupons were a huge success!
Seeing this, the GoodPrice store’s marketing team decided to introduce even more coupons!
It’s time for combinable banana coupons – and even multi-product coupons, so-called X coupons!”

---

## 2. New Coupons

- **B5-MAX10**: 5% discount on bananas, combinable with B* coupons, MAX=10%.
- **B5-MAX15**: 5% discount on bananas, combinable with B* coupons, MAX=15%.
- **X5**: 5% discount on all products, not combinable with any other coupon. If it applies even partially → cannot be returned.
- **X10**: 10% discount on all products, not combinable with any other coupon. If it applies even partially → cannot be returned.
- **X5-MAX10**: 5% discount on all products, MAX=10% per product, combinable with any coupon. If it applies even partially → cannot be returned.

---

## 3. Rules

- All previous rules also apply to X coupons.
- **New rule:** if an X coupon applies even partially, it cannot be returned.
- It is not possible to return an A + B coupon in place of an X coupon (e.g., X5 cannot be replaced by A5+B5), because then both would have to be returned separately.
- **Special exception:** X5 can be combined with A5-MAX10, since X5 essentially contains an A5 coupon.

---

## 4. Examples (from unit tests)

1. **B5-MAX10 alone** → 5% discount on bananas.
2. **B5 + B5-MAX10** → 10% discount on bananas (MAX=10%).
3. **B5 + B5-MAX15 + B5-MAX10** → strictest MAX=10%, return the bigger MAX (15) coupon.
4. **X5** → 5% discount on all products.
5. **X10 + other coupons** → only X10 applies, all others are returned.
6. **X5 partially applies** (e.g., only apples in the cart) → X5 remains, other coupons are returned.

7. **X5 + A5-MAX10** → exception: can be combined (apple 10%, banana 5%).
8. **X5-MAX10 alone** → 5% discount on all products, MAX=10%.
9. **X5-MAX10 + A5 + B5** → both products discounted up to 10%, no returns.
10. **X5-MAX10 + A5 + A5** → apple side would exceed MAX=10%, one A5 is returned.
11. **X5-MAX10 + B5-MAX10 + B5** → banana discounted 10%, no returns.
12. **X5 partial application** (only banana in cart + A5 coupon) → banana gets 5% from X5, A5 is returned, X5 remains.

---

## 5. Development Task

"Your task is to modify the existing solution to comply with CR5!
The next subsection is CR6. Only start CR6 if all previous CRs are fully implemented!
We recommend using the unit tests at the end of the subsection, as well as vibe coding techniques to complete the task!"

---

## 6. Unit Tests

```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 org.store.*;

class StoreCR5Tests {
    static Store target;
    static Period normal;

    @BeforeAll
    public static void initStore() {
        target = new Store();
        normal = new Period("Normal");
        // Settings known from CR0/CR1
        normal.setUnitPrice(Product.APPLE, 500.0);
        normal.setUnitPrice(Product.BANANA, 450.0);
        normal.setDiscount(Product.APPLE, 5.0, 0.1);
        normal.setDiscount(Product.APPLE, 20.0, 0.15);
        normal.setDiscount(Product.BANANA, 2.0, 0.1);
        target.addPeriod(normal);
    }

    // --- Helper: rounding to 5 HUF (CR0 rule)
    private double roundTo5(double amount) {
        double remainder = amount % 10.0;
        double base = Math.floor(amount / 10.0) * 10.0;
        if (remainder < 2.5) return base;
        if (remainder < 5.0) return base + 5.0;
        if (remainder < 7.5) return base + 5.0;
        return base + 10.0;
    }

    @Test
    void test_cr5_example1_b5max10_alone() {
        Cart cart = new Cart(List.of(new Item(Product.BANANA, 1.0)));
        PriceInfo price = target.getCartPrice(cart, normal, List.of("B5-
MAX10"));
        assertEquals(roundTo5(450.0 * 0.95), price.getAmount(), 0.001);
        assertEquals(List.of(), price.getUnusedCoupons());
    }

    @Test
    void test_cr5_example2_b5_and_b5max10_max10() {
        Cart cart = new Cart(List.of(new Item(Product.BANANA, 1.0)));
        PriceInfo price = target.getCartPrice(cart, normal, List.of("B5",
"B5-MAX10"));
        assertEquals(roundTo5(450.0 * 0.90), price.getAmount(), 0.001);
        assertEquals(List.of(), price.getUnusedCoupons());
    }

    @Test
    void test_cr5_example3_twoMAX_strictest10_return15() {
        Cart cart = new Cart(List.of(new Item(Product.BANANA, 1.0)));
        PriceInfo price = target.getCartPrice(cart, normal, List.of("B5",
"B5-MAX15", "B5-MAX10"));
        assertEquals(roundTo5(450.0 * 0.90), price.getAmount(), 0.001);
        assertEquals(List.of("B5-MAX15"), price.getUnusedCoupons());
    }

    @Test
    void test_cr5_example4_x5_alone_twoProducts() {
        Cart cart = new Cart(List.of(
                new Item(Product.APPLE, 1.0),
                new Item(Product.BANANA, 1.0)
        ));
        PriceInfo price = target.getCartPrice(cart, normal, List.of("X5"));
        assertEquals(roundTo5(950.0 * 0.95), price.getAmount(), 0.001);
        assertEquals(List.of(), price.getUnusedCoupons());
    }

    @Test
    void test_cr5_example5_x10_notCombinable() {
        Cart cart = new Cart(List.of(
                new Item(Product.APPLE, 2.0),
                new Item(Product.BANANA, 1.0)
        ));
        PriceInfo price = target.getCartPrice(cart, normal, List.of("X10",
"A5", "B5", "A5-MAX10", "B5-MAX10"));
        assertEquals(roundTo5(1450.0 * 0.90), price.getAmount(), 0.001);
        assertEquals(List.of("A5", "B5", "A5-MAX10", "B5-MAX10"),
price.getUnusedCoupons());
    }

    @Test
    void test_cr5_example6_x5_partial_notReturned() {
        Cart cart = new Cart(List.of(new Item(Product.APPLE, 1.0)));
        PriceInfo price = target.getCartPrice(cart, normal, List.of("X5",
"B5"));
        assertEquals(roundTo5(500.0 * 0.95), price.getAmount(), 0.001);
        assertEquals(List.of("B5"), price.getUnusedCoupons());
    }

    @Test
    void test_cr5_example7_x5_and_a5max10_exception() {
        Cart cart = new Cart(List.of(
                new Item(Product.APPLE, 1.0),
                new Item(Product.BANANA, 1.0)
        ));
        PriceInfo price = target.getCartPrice(cart, normal, List.of("X5",
"A5-MAX10"));
        double expected = roundTo5(500.0 * 0.90 + 450.0 * 0.95);
        assertEquals(expected, price.getAmount(), 0.001);
        assertEquals(List.of(), price.getUnusedCoupons());
    }

    @Test
    void test_cr5_example8_x5max10_alone_twoProducts() {
        Cart cart = new Cart(List.of(
                new Item(Product.APPLE, 1.0),
                new Item(Product.BANANA, 1.0)
        ));
        PriceInfo price = target.getCartPrice(cart, normal, List.of("X5-
MAX10"));
        assertEquals(roundTo5(500.0 * 0.95 + 450.0 * 0.95),
price.getAmount(), 0.001);
        assertEquals(List.of(), price.getUnusedCoupons());
    }

    @Test
    void test_cr5_example9_x5max10_a5_b5() {
        Cart cart = new Cart(List.of(
                new Item(Product.APPLE, 1.0),
                new Item(Product.BANANA, 1.0)
        ));
        PriceInfo price = target.getCartPrice(cart, normal, List.of("X5-
MAX10", "A5", "B5"));
        assertEquals(roundTo5(500.0 * 0.90 + 450.0 * 0.90),
price.getAmount(), 0.001);
        assertEquals(List.of(), price.getUnusedCoupons());
    }

    @Test
    void test_cr5_example10_x5max10_twoA5_oneReturned() {
        Cart cart = new Cart(List.of(new Item(Product.APPLE, 1.0)));
        PriceInfo price = target.getCartPrice(cart, normal, List.of("X5-
MAX10", "A5", "A5"));
        assertEquals(roundTo5(500.0 * 0.90), price.getAmount(), 0.001);
        assertEquals(List.of("A5"), price.getUnusedCoupons());
    }

    @Test
    void test_cr5_example11_x5max10_bSide() {
        Cart cart = new Cart(List.of(new Item(Product.BANANA, 1.0)));
        PriceInfo price = target.getCartPrice(cart, normal, List.of("X5-
MAX10", "B5-MAX10", "B5"));
        assertEquals(roundTo5(450.0 * 0.90), price.getAmount(), 0.001);
        assertEquals(List.of(), price.getUnusedCoupons());
    }

    @Test
    void test_cr5_example12_x5_partialWithA5() {
        Cart cart = new Cart(List.of(new Item(Product.BANANA, 1.0)));
        PriceInfo price = target.getCartPrice(cart, normal, List.of("X5",
"A5"));
        assertEquals(roundTo5(450.0 * 0.95), price.getAmount(), 0.001);
```

```java
        assertEquals(List.of("A5"), price.getUnusedCoupons());
    }
}
```