

```
package org.junit.samples.money;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;
import junit.framework.JUnit4TestAdapter;
import junit.samples.money.IMoney;
import junit.samples.money.Money;
import junit.samples.money.MoneyBag;
import org.junit.Before;
import org.junit.Test;

public class MoneyTest {
    private Money f12CHF;
    private Money f14CHF;
    private Money f7USD;
    private Money f21USD;

    private IMoney fMB1;
    private IMoney fMB2;

    public static junit.framework.Test suite() {
        return new JUnit4TestAdapter(MoneyTest.class);
    }

    @Before public void setUp() {
        f12CHF= new Money(12, "CHF");
        f14CHF= new Money(14, "CHF");
        f7USD= new Money( 7, "USD");
        f21USD= new Money(21, "USD");

        fMB1= MoneyBag.create(f12CHF, f7USD);
        fMB2= MoneyBag.create(f14CHF, f21USD);
    }

    @Test public void testBagMultiply() {
        // {[12 CHF][7 USD]} *2 == {[24 CHF][14 USD]}
        IMoney expected= MoneyBag.create(new Money(24, "CHF"), new Money(14, "USD"));
;
        assertEquals(expected, fMB1.multiply(2));
        assertEquals(fMB1, fMB1.multiply(1));
        assertTrue(fMB1.multiply(0).isZero());
    }

    @Test public void testBagNegate() {
        // {[12 CHF][7 USD]} negate == {[-12 CHF][-7 USD]}
        IMoney expected= MoneyBag.create(new Money(-12, "CHF"), new Money(-7, "USD"));
);

        assertEquals(expected, fMB1.negate());
    }

    @Test public void testBagSimpleAdd() {
        // {[12 CHF][7 USD]} + [14 CHF] == {[26 CHF][7 USD]}
        IMoney expected= MoneyBag.create(new Money(26, "CHF"), new Money(7, "USD"));
        assertEquals(expected, fMB1.add(f14CHF));
    }

    @Test public void testBagSubtract() {
        // {[12 CHF][7 USD]} - {[14 CHF][21 USD]} == {[-2 CHF][-14 USD]}
        IMoney expected= MoneyBag.create(new Money(-2, "CHF"), new Money(-14, "USD"));
);

        assertEquals(expected, fMB1.subtract(fMB2));
    }

    @Test public void testBagSumAdd() {
        // {[12 CHF][7 USD]} + {[14 CHF][21 USD]} == {[26 CHF][28 USD]}
        IMoney expected= MoneyBag.create(new Money(26, "CHF"), new Money(28, "USD"));
;
    }
}
```

## MoneyTest.java

```
        assertEquals(expected, fMB1.add(fMB2));
    }
    @Test public void testIsZero() {
        assertTrue(fMB1.subtract(fMB1).isZero());
        assertTrue(MoneyBag.create(new Money(0, "CHF"), new Money(0, "USD")).isZero());
    }
    @Test public void testMixedSimpleAdd() {
        // [12 CHF] + [7 USD] == {[12 CHF][7 USD]}
        IMoney expected= MoneyBag.create(f12CHF, f7USD);
        assertEquals(expected, f12CHF.add(f7USD));
    }
    @Test public void testBagNotEquals() {
        IMoney bag= MoneyBag.create(f12CHF, f7USD);
        assertFalse(bag.equals(new Money(12, "DEM").add(f7USD)));
    }
    @Test public void testMoneyBagEquals() {
        assertTrue(!fMB1.equals(null));

        assertEquals(fMB1, fMB1);
        IMoney equal= MoneyBag.create(new Money(12, "CHF"), new Money(7, "USD"));
        assertTrue(fMB1.equals(equal));
        assertTrue(!fMB1.equals(f12CHF));
        assertTrue(!f12CHF.equals(fMB1));
        assertTrue(!fMB1.equals(fMB2));
    }
    @Test public void testMoneyBagHash() {
        IMoney equal= MoneyBag.create(new Money(12, "CHF"), new Money(7, "USD"));
        assertEquals(fMB1.hashCode(), equal.hashCode());
    }
    @Test public void testMoneyEquals() {
        assertTrue(!f12CHF.equals(null));
        Money equalMoney= new Money(12, "CHF");
        assertEquals(f12CHF, f12CHF);
        assertEquals(f12CHF, equalMoney);
        assertEquals(f12CHF.hashCode(), equalMoney.hashCode());
        assertTrue(!f12CHF.equals(f14CHF));
    }
    @Test public void zeroMoniesAreEqualRegardlessOfCurrency() {
        Money zeroDollars= new Money(0, "USD");
        Money zeroFrancs= new Money(0, "CHF");

        assertEquals(zeroDollars, zeroFrancs);
        assertEquals(zeroDollars.hashCode(), zeroFrancs.hashCode());
    }
    @Test public void testMoneyHash() {
        assertTrue(!f12CHF.equals(null));
        Money equal= new Money(12, "CHF");
        assertEquals(f12CHF.hashCode(), equal.hashCode());
    }
    @Test public void testSimplify() {
        IMoney money= MoneyBag.create(new Money(26, "CHF"), new Money(28, "CHF"));
        assertEquals(new Money(54, "CHF"), money);
    }
    @Test public void testNormalize2() {
        // {[12 CHF][7 USD]} - [12 CHF] == [7 USD]
        Money expected= new Money(7, "USD");
        assertEquals(expected, fMB1.subtract(f12CHF));
    }
    @Test public void testNormalize3() {
        // {[12 CHF][7 USD]} - {[12 CHF][3 USD]} == [4 USD]
        IMoney ms1= MoneyBag.create(new Money(12, "CHF"), new Money(3, "USD"));
        Money expected= new Money(4, "USD");
        assertEquals(expected, fMB1.subtract(ms1));
    }
}
```

```
}
@Test public void testNormalize4() { // [12 CHF] - {[12 CHF][3 USD]} == [-3 USD]
    IMoney msl= MoneyBag.create(new Money(12, "CHF"), new Money(3, "USD"));
    Money expected= new Money(-3, "USD");
    assertEquals(expected, f12CHF.subtract(msl));
}
@Test public void testPrint() {
    assertEquals("[12 CHF]", f12CHF.toString());
}
@Test public void testSimpleAdd() {
    // [12 CHF] + [14 CHF] == [26 CHF]
    Money expected= new Money(26, "CHF");
    assertEquals(expected, f12CHF.add(f14CHF));
}
@Test public void testSimpleBagAdd() {
    // [14 CHF] + {[12 CHF][7 USD]} == {[26 CHF][7 USD]}
    IMoney expected= MoneyBag.create(new Money(26, "CHF"), new Money(7, "USD"));
    assertEquals(expected, f14CHF.add(fMB1));
}
@Test public void testSimpleMultiply() {
    // [14 CHF] *2 == [28 CHF]
    Money expected= new Money(28, "CHF");
    assertEquals(expected, f14CHF.multiply(2));
}
@Test public void testSimpleNegate() {
    // [14 CHF] negate == [-14 CHF]
    Money expected= new Money(-14, "CHF");
    assertEquals(expected, f14CHF.negate());
}
@Test public void testSimpleSubtract() {
    // [14 CHF] - [12 CHF] == [2 CHF]
    Money expected= new Money(2, "CHF");
    assertEquals(expected, f14CHF.subtract(f12CHF));
}
}
```