const { expect } = require("chai");

const { ethers } = require("hardhat");

describe("AITU\_DAMIR\_SE2327", function () {

  let AITU\_DAMIR\_SE2327;

  let token;

  let owner;

  let addr1;

  let addr2;

  beforeEach(async function () {

    // Получаем аккаунты

    [owner, addr1, addr2] = await ethers.getSigners();

    // Деплоим контракт

    AITU\_DAMIR\_SE2327 = await ethers.getContractFactory("AITU\_DAMIR\_SE2327");

    token = await AITU\_DAMIR\_SE2327.deploy();

    await token.deployed();

  });

  describe("Deployment", function () {

    it("Should set the correct name and symbol", async function () {

      expect(await token.name()).to.equal("AITU\_DAMIR\_SE2327");

      expect(await token.symbol()).to.equal("MTK");

    });

    it("Should mint initial supply to the owner", async function () {

      const ownerBalance = await token.balanceOf(owner.address);

      expect(ownerBalance).to.equal(2000 \* 10 \*\* 18); // 2000 токенов с учетом decimals

    });

  });

  describe("transferWithDetails", function () {

    it("Should transfer tokens and emit TransactionDetails event", async function () {

      const amount = 100;

      const tx = await token.transferWithDetails(addr1.address, amount);

      // Проверяем, что событие было вызвано

      await expect(tx)

        .to.emit(token, "TransactionDetails")

        .withArgs(owner.address, addr1.address, amount, await token.getBlockTimestamp());

      // Проверяем балансы

      const ownerBalance = await token.balanceOf(owner.address);

      const addr1Balance = await token.balanceOf(addr1.address);

      expect(ownerBalance).to.equal(2000 \* 10 \*\* 18 - amount);

      expect(addr1Balance).to.equal(amount);

    });

    it("Should revert if sender has insufficient balance", async function () {

      const amount = 3000 \* 10 \*\* 18; // Больше, чем у владельца

      await expect(token.transferWithDetails(addr1.address, amount)).to.be.revertedWith("ERC20: transfer amount exceeds balance");

    });

  });

  describe("getBlockTimestamp", function () {

    it("Should return the current block timestamp", async function () {

      const blockNumber = await ethers.provider.getBlockNumber();

      const block = await ethers.provider.getBlock(blockNumber);

      expect(await token.getBlockTimestamp()).to.equal(block.timestamp);

    });

  });

  describe("getSenderAndReceiver", function () {

    it("Should return the correct sender and receiver addresses", async function () {

      const [sender, receiver] = await token.getSenderAndReceiver(addr1.address);

      expect(sender).to.equal(owner.address); // Отправитель — владелец контракта

      expect(receiver).to.equal(addr1.address); // Получатель — addr1

    });

  });

});

