НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕЇНІКИ

ЛАБОРАТОРНА РОБОТА 4

з дисципліни:

«Сучасні методології і технології розробки програмного забезпечення»

на тему:

«Багатошарова архітектура програмних додатків. Використання системи контролю версій. Основи UML.»

Студента 3 курсу групи ІТ-81
Венделовського Івана Сергійовича

Кількість балів:_____ Оцінка_____

Викладач:_____к.т.н. Штифурак Юрій Михайлович
(оцінка)

Київ – 2021

ЗМІСТ

1 ПОСТАНОКА ЗАДАЧІ

1) Покрити розроблене ПЗ автономними модульними тестами.

1 ПОКРИТТЯ ЗМІСТОВНОЇ ЛОГІКИ МОДУЛЬНИМИ ТЕСТАМИ

1.1 Реалізація тестів для класу «BillSevice»

```
public class BillServiceTest
{
    private BillService billService;
    private Mock<IBillRepository> billRepository;
    private Mock<IUserRepository> userRepository;
    private Mock<IDeliveryRepository> deliveryRepository;
    private Mock<IWayRepository> wayRepository;

    [SetUp]
    public void SetupBeforeEachTest()
    {
        billRepository = new Mock<IBillRepository>();
        userRepository=new Mock<IUserRepository>();
        deliveryRepository=new Mock<IDeliveryRepository>();
        wayRepository=new Mock<IWayRepository>();

        billRepository.Setup( expression: a => a.FindByIdAndIsDeliveryPaidFalse(ServicesTestConstant.getBillId()))
            .Returns(ServicesTestConstant.getBill());
        billRepository.Setup( expression: a => a.FindAllByUserIdAndIsDeliveryPaidFalse(It.IsAny<string>()))
            .Returns(ServicesTestConstant.getBills());

        userRepository.Setup( expression: a => a.FindByIdAndUserMoneyInCentsGreaterThanEqual
            ( userName: ServicesTestConstant.getUserId(),ServicesTestConstant.getBill().CostInCents)
            ).Returns(ServicesTestConstant.getAddreser());

        userRepository.Setup( expression: a => a.FindByEmail(It.IsAny<string>())
        ).Returns(ServicesTestConstant.getAdversee());
        userRepository.Setup( expression: a => a.FindByName(It.IsAny<string>())
        ).Returns(ServicesTestConstant.getAdversee());

        wayRepository.Setup( expression: a => a.FindByLocalitySand_IdAndLocalityGet_Id
            ( localitySandID: It.IsAny<long>(), localityGetID: It.IsAny<long>())
        ).Returns(ServicesTestConstant.getWay());
        billService = new BillService
            (billRepository.Object, userRepository.Object,deliveryRepository.Object,wayRepository.Object);
    }
```

Рисунок 1.1 – Тестовий клас, необхідні фікстури та їх ініціалізація.

1.1.1 Реалізація тестів методу «InitializeBill»

```
[Test]
public void initializeBillCorrect(){
    Bill bill = ServicesTestConstant.getBill();
    bill.CostInCents=_defaultCostInCents;
    bill.BillId=_defaultBillId;
    bill.IsDeliveryPaid = false;
    Delivery delivery = ServicesTestConstant.getDelivery();
    delivery.DeliveryId  = _defaultDeliveryId;
    _wayRepository.Setup( expression: a => a.FindByLocalitySand_IdAndLocalityGet_Id
        ( localitySandID: It.IsAny<long>(), localityGetID: It.IsAny<long>())
    ).Returns(ServicesTestConstant.getWay());

    Bill billResult = _billService.InitializeBill(ServicesTestConstant.getDeliveryOrderCreateDto(),
        initiatorName: ServicesTestConstant.getUserId());

    Assert.AreEqual( expected: bill, actual: billResult);
}
```

Рисунок 1.2 – Перевірка коректності створення рахунку при коректних вхідних даних.

```
[Test]
public void initializeBillCorrectInCorrectAddressee(){
    Bill bill = ServicesTestConstant.getBill();
    bill.CostInCents=_defaultCostInCents;
    bill.BillId=_defaultBillId;
    bill.IsDeliveryPaid = false;
    Delivery delivery = ServicesTestConstant.getDelivery();
    delivery.DeliveryId  = _defaultDeliveryId;
    _userRepository.Setup( expression: a => a.FindByEmail(It.IsAny<string>())
    ).Returns(null as User);

    var actualResult =
        Assert.Throws<NoSuchUserException>( code: () => _billService.InitializeBill
            (ServicesTestConstant.getDeliveryOrderCreateDto(), initiatorName: ServicesTestConstant.getUserId()));

    Assert.AreEqual( expected: typeof(NoSuchUserException), actualResult.GetType());
}
```

Рисунок 1.3 – Перевірка виникнення «NoSuchUserException» виключення у випадку якщо не існує користувача отриманого із вхідних даних.

```
[Test]
public void initializeBillIncorrectInWay(){
    Bill bill = ServicesTestConstant.getBill();
    bill.CostInCents=_defaultCostInCents;
    bill.BillId=_defaultBillId;
    bill.IsDeliveryPaid = false;
    Delivery delivery = ServicesTestConstant.getDelivery();
    delivery.DeliveryId  = _defaultDeliveryId;

    _wayRepository.Setup( expression: a => a.FindByLocalitySand_IdAndLocalityGet_Id
        ( localitySandID: It.IsAny<long>(), localityGetID: It.IsAny<long>())
    ).Returns((Way) null);

    var actualResult =
        Assert.Throws<NoSuchWayException>( code: ()=>_billService.InitializeBill
            (ServicesTestConstant.getDeliveryOrderCreateDto(), initiatorName: ServicesTestConstant.getUserId()));

    Assert.AreEqual( expected: typeof(NoSuchWayException), actualResult.GetType());
}
```

Рисунок 1.4 – Перевірка виникнення «NoSuchWayException» виключення у випадку якщо не існує маршруту отриманого із вхідних даних.

1.1.2 Реалізація тестів методу «GetBillsToPayByUserName»

```
[Test]
public void getBillsToPayByUserId()
{
    BillInfoToPayModel billInfoToPayDto = ServicesTestConstant.getBillInfoToPayDto();
    Bill bill = ServicesTestConstant.getBill();
    billInfoToPayDto.LocalityGetName = bill.Delivery.Way.LocalityGet.NameEn;
    billInfoToPayDto.LocalitySandName = bill.Delivery.Way.LocalitySand.NameEn;

    List<BillInfoToPayModel> result = _billService.GetBillsToPayByUserName(ServicesTestConstant.getUserId());

    _billRepository.Verify( expression: place => place.FindAllByUserIdAndIsDeliveryPaidFalse
        (ServicesTestConstant.getUserId()), Times.Once());
    Assert.AreEqual( expected: ServicesTestConstant.getBills().Count, actual: result.Count);
    Assert.AreEqual( expected: billInfoToPayDto, actual: result[0]);
}
```

Рисунок 1.5 – Перевірка коректності отримання платежів за ім’ям користувача при коректних вхідних даних.

```
[Test]
public void getBillsToPayByUserIdUserIsNotExist()
{
    _billRepository.Setup( expression: a => a.FindAllByUserIdAndIsDeliveryPaidFalse(It.IsAny<string>())).Returns(new List<Bill>());

    List<BillInfoToPayModel> billInfoToPayDtos = _billService.GetBillsToPayByUserName(ServicesTestConstant.getUserId());

    _billRepository.Verify( expression: place => place.FindAllByUserIdAndIsDeliveryPaidFalse(It.IsAny<string>()), Times.Once());
    Assert.AreEqual( expected: 0, actual: billInfoToPayDtos.Count);
}
```

Рисунок 1.6 – Перевірка відсутності платежів у випадку якщо не існує зданого користувача.

```
[Test]
public void initializeBillIncorrectInWay(){
    Bill bill = ServicesTestConstant.getBill();
    bill.CostInCents=_defaultCostInCents;
    bill.BillId=_defaultBillId;
    bill.IsDeliveryPaid = false;
    Delivery delivery = ServicesTestConstant.getDelivery();
    delivery.DeliveryId  = _defaultDeliveryId;

    _wayRepository.Setup( expression: a => a.FindByLocalitySand_IdAndLocalityGet_Id
        ( localitySandID: It.IsAny<long>(), localityGetID: It.IsAny<long>())
    ).Returns((Way) null);

    var actualResult =
        Assert.Throws<NoSuchWayException>( code: ()=>_billService.InitializeBill
            (ServicesTestConstant.getDeliveryOrderCreateDto(), initiatorName: ServicesTestConstant.getUserId()));

    Assert.AreEqual( expected: typeof(NoSuchWayException), actualResult.GetType());
}
```

Рисунок 1.7 – Перевірка виникнення «NoSuchWayException» виключення у випадку якщо не існує маршруту отриманого із вхідних даних.

1.1.3 Реалізація тестів методу «PayForDelivery»

```
[Test]
public void payForDeliveryWhenAllCorrect(){
    Bill bill = ServicesTestConstant.getBill();
    bill.IsDeliveryPaid = false;
    _billRepository.Setup( expression: a => a.FindByIdAndIsDeliveryPaidFalse(ServicesTestConstant.getBillId())).Returns(bill);

    bool payResult = _billService.PayForDelivery( ServicesTestConstant.getUserId(),  ServicesTestConstant.getBillId());

    _billRepository.Verify
        ( expression: place => place.FindByIdAndIsDeliveryPaidFalse(ServicesTestConstant.getBillId()), Times.Once());
    _userRepository.Verify( expression: place => place.FindByIdAndUserMoneyInCentsGreaterThanEqual
        ( userName: ServicesTestConstant.getUserId(), bill.CostInCents), Times.Once());

    Assert.IsTrue(payResult);
    Assert.IsTrue(bill.IsDeliveryPaid);
}
```

Рисунок 1.8 – Перевірка коректності оплати при коректних вхідних даних.

```
[Test]
public void payForDeliveryNotEnoughMoney(){
    Bill bill = ServicesTestConstant.getBill();
    bill.IsDeliveryPaid=false;
    User adverser = ServicesTestConstant.getAddreser();
    adverser.UserMoneyInCents = 0L;
    _userRepository.Setup( expression: a => a.FindByIdAndUserMoneyInCentsGreaterThanEqual
        ( userName: ServicesTestConstant.getUserId(),ServicesTestConstant.getBill().CostInCents)
    ).Returns((User) null);

    var actualResult =
        Assert.Throws<NotEnoughMoneyException>
            ( code: ()=>_billService.PayForDelivery(ServicesTestConstant.getUserId(), ServicesTestConstant.getBillId()));

    Assert.AreEqual( expected: typeof(NotEnoughMoneyException), actualResult.GetType());
}
```

Рисунок 1.9 – Перевірка виникнення «NotEnoughMoneyException» виключення у випадку якщо у користувача не достатньо коштів для виконання оплати.

```
[Test]
public void payForDeliveryDeliveryAlreadyPaid() {
    _billRepository.Setup( expression: a => a.FindByIdAndIsDeliveryPaidFalse(ServicesTestConstant.getBillId()))
        .Returns((Bill) null);

    var actualResult =
        Assert.Throws<DeliveryAlreadyPaidException>( code: ()=>_billService.PayForDelivery
            (ServicesTestConstant.getUserId(), ServicesTestConstant.getBillId()));

    Assert.AreEqual( expected: typeof(DeliveryAlreadyPaidException), actualResult.GetType());
}
```

Рисунок 1.10 – Перевірка виникнення «DeliveryAlreadyPaidException» виключення у випадку якщо рахунок вже оплачено.

1.2 Реалізація тестів для класу «DeliveryService»

```
public class DeliveryServiceTest
{
    private DeliveryService _deliveryService;
    private Mock<IWayRepository> _wayRepository;
    private Mock<IDeliveryRepository> _deliveryRepository;

    [SetUp]
    public void SetupBeforeEachTest()
    {
        _wayRepository = new Mock<IWayRepository>();
        _deliveryRepository = new Mock<IDeliveryRepository>();
        _deliveryService = new DeliveryService(_wayRepository.Object, _deliveryRepository.Object);
    }
```

Рисунок 1.11 – Тестовий клас, необхідні фікстури та їх ініціалізація.

1.2.1 Реалізація тестів методу «GetDeliveryInfoToGet»

```
[Test]
public void GetDeliveryInfoToGet()
{
    Delivery delivery = ServicesTestConstant.getDelivery();
    delivery.Bill = ServicesTestConstant.getBill();
    delivery.Bill.User = ServicesTestConstant.getAddreser();
    DeliveryInfoToGetDto deliveryInfoToGetDto = ServicesTestConstant.getDeliveryInfoToGetDto();
    deliveryInfoToGetDto.LocalityGetName = delivery.Way.LocalityGet.NameEn;
    deliveryInfoToGetDto.LocalitySandName = delivery.Way.LocalitySand.NameEn;

    _deliveryRepository.Setup( expression: s => s.FindAllByAddressee_IdAndIsPackageReceivedFalseAndBill_IsDeliveryPaidTrue(
        ServicesTestConstant.getUserId())).Returns(new List<Delivery>{delivery});

    List<DeliveryInfoToGetDto> result = _deliveryService.GetDeliveryInfoToGet(ServicesTestConstant.getUserId());

    _deliveryRepository.Verify(
        expression: place =>
            place.FindAllByAddressee_IdAndIsPackageReceivedFalseAndBill_IsDeliveryPaidTrue(It.IsAny<string>()),
        Times.Once());

    Assert.AreEqual( expected: deliveryInfoToGetDto,  actual: result[0]);
    Assert.AreEqual( expected: ServicesTestConstant.getDeliveres().Count,  actual: result.Count);
}
```

Рисунок 1.12 – Перевірка коректності списку доставок на отримання при коректних вхідних даних.

1.2.2 Реалізація тестів методу «ConfirmGettingDelivery»

```
[Test]
public void confirmGettingDeliveryAllCorrect()
{
    Delivery delivery = ServicesTestConstant.getDelivery();
    _deliveryRepository.Setup( expression: s => s.FindByIdAndAddressee_IdAndIsPackageReceivedFalse(
            ServicesTestConstant.getUserId(), ServicesTestConstant.getDeliveryId()))
        .Returns(delivery);

    bool result = _deliveryService.ConfirmGettingDelivery(ServicesTestConstant.getUserId(),
        ServicesTestConstant.getDeliveryId());

    _deliveryRepository.Verify(
        expression: place =>
            place.FindByIdAndAddressee_IdAndIsPackageReceivedFalse
                ( userName: It.IsAny<string>(),  deliveryId: It.IsAny<long>()), Times.Once());
    _deliveryRepository.Verify(
        expression: place =>
            place.Save(),
        Times.Once());

    Assert.IsTrue(result);
    Assert.IsTrue(delivery.IsPackageReceived);
}
```

Рисунок 1.13 – Перевірка коректності підтвердження отримання доставки при коректних вхідних даних.

```
[Test]
public void confirmGettingDeliveryIsNoExistDelivery()
{
    _deliveryRepository.Setup( expression: s => s.FindByIdAndAddressee_IdAndIsPackageReceivedFalse(
            ServicesTestConstant.getUserId(), ServicesTestConstant.getDeliveryId()))
        .Returns((Delivery) null);
    var actualResult =
        Assert.Throws<AskedDataIsNotExist>( code: () => _deliveryService.ConfirmGettingDelivery(
            ServicesTestConstant.getUserId(),
            ServicesTestConstant.getDeliveryId()));
    Assert.AreEqual( expected: typeof(AskedDataIsNotExist), actualResult.GetType());
}
```

Рисунок 1.14 – Перевірка виникнення виключення «AskedDataIsNotExist» у випадку якщо заданої доставки не існує.

1.2.3 Реалізація тестів методу «GetDeliveryCostAndTimeDto»

```
[Test]
public void getDeliveryCostAndTimeDtoAllCorrect()
{
    DeliveryInfoRequestModel deliveryInfoRequestDto = getDeliveryInfoRequestDto( weightRangeReal: 1);
    PriceAndTimeOnDeliveryModel priceAndTimeOnDeliveryDto = getPriceAndTimeOnDeliveryDto();
    Delivery delivery = ServicesTestConstant.getDelivery();
    Way way = delivery.Way;
    _wayRepository.Setup( expression: s => s.FindByLocalitySand_IdAndLocalityGet_Id
        ( localitySandID: It.IsAny<long>(), localityGetID: It.IsAny<long>())).Returns(way);

    PriceAndTimeOnDeliveryModel result = _deliveryService.GetDeliveryCostAndTimeDto(deliveryInfoRequestDto);
    _wayRepository.Verify(
        expression: s => s.FindByLocalitySand_IdAndLocalityGet_Id
            ( localitySandID: It.IsAny<long>(), localityGetID: It.IsAny<long>()), Times.Once());
    Assert.AreEqual( expected: priceAndTimeOnDeliveryDto, actual: result);
}
```

Рисунок 1.15 – Перевірка коректності отримання даних про час та вартість доставки при коректних вхідних даних.

```
[Test]
public void getDeliveryCostAndTimeIncorrectWay()
{
    DeliveryInfoRequestModel deliveryInfoRequestDto = getDeliveryInfoRequestDto( weightRangeReal: 1);
    PriceAndTimeOnDeliveryModel priceAndTimeOnDeliveryDto = getPriceAndTimeOnDeliveryDto();
    _wayRepository.Setup( expression: s => s.FindByLocalitySand_IdAndLocalityGet_Id
        ( localitySandID: It.IsAny<long>(), localityGetID: It.IsAny<long>())).Returns((Way) null);

    var actualResult =
        Assert.Throws<NoSuchWayException>( code: () =>
            _deliveryService.GetDeliveryCostAndTimeDto(deliveryInfoRequestDto));
    Assert.AreEqual( expected: typeof(NoSuchWayException), actualResult.GetType());
}
```

Рисунок 1.16 – Перевірка виникнення «NoSuchWayException» виключення у випадку якщо заданого маршруту не існує.

```
[Test]
public void getDeliveryCostAndTimeIncorrectWeightFactorBiggerOnOne()
{
    int weightRangeMax = 2;
    int weightRangeReal = 2;
    DeliveryInfoRequestModel deliveryInfoRequestDto = getDeliveryInfoRequestDto(weightRangeReal);
    Delivery delivery = ServicesTestConstant.getDelivery();
    Way way = delivery.Way;
    way.WayToTariffWeightFactors[0].TariffWeightFactor.MaxWeightRange = weightRangeMax;
    _wayRepository.Setup( expression: s => s.FindByLocalitySand_IdAndLocalityGet_Id
        ( localitySandID: It.IsAny<long>(), localityGetID: It.IsAny<long>())).Returns(way);

    var actualResult =
        Assert.Throws<UnsupportableWeightFactorException>( code: () =>
            _deliveryService.GetDeliveryCostAndTimeDto(deliveryInfoRequestDto));
    Assert.AreEqual( expected: typeof(UnsupportableWeightFactorException), actualResult.GetType());
}
```

Рисунок 1.17 – Перевірка виникнення «UnsupportableWeightFactorException» виключення у випадку вантаж занадто важки для заданого маршруту.

1.3 Реалізація тестів для класу «UserService»

```
public class DeliveryServiceTest
{
    private DeliveryService _deliveryService;
    private Mock<IWayRepository> _wayRepository;
    private Mock<IDeliveryRepository> _deliveryRepository;

    [SetUp]
    public void SetupBeforeEachTest()
    {
        _wayRepository = new Mock<IWayRepository>();
        _deliveryRepository = new Mock<IDeliveryRepository>();
        _deliveryService = new DeliveryService(_wayRepository.Object, _deliveryRepository.Object);
    }
```

Рисунок 1.18 – Тестовий клас, необхідні фікстури та їх ініціалізація.

1.3.1 Реалізація тестів методу «FindByName»

```
[Test]
public void findByEmailAllCorrect()
{
    User user = ServicesTestConstant.getAddreser();
    userRepository.Setup( expression: s => s.FindByName(It.IsAny<string>())).Returns(user);

    User result = _userService.FindByName(user.UserName);

    Assert.AreEqual( expected: user,  actual: result);
    userRepository.Verify(
        expression: place =>
            place.FindByName(It.IsAny<string>()),
        Times.Once());
}
```

Рисунок 1.19 – Перевірка коректності пошуку користувача при коректних вхідних даних.

```
[Test]
public void findByEmailIncorrectEmail()
{
    User user = ServicesTestConstant.getAddreser();
    userRepository.Setup( expression: s => s.FindByEmail(It.IsAny<string>())).Returns((User) null);

    var actualResult =
        Assert.Throws<UsernameNotFoundException>( code: () => _userService.FindByName(user.UserName));

    Assert.AreEqual( expected: typeof(UsernameNotFoundException), actualResult.GetType());
}
```

Рисунок 1.20 – Перевірка виникнення «UsernameNotFoundException» виключення у випадку якщо користувача з заданою поштою не існує.

1.3.2 Реалізація тестів методу «ReplenishAccountBalance»

```
[Test]
public void replenishAccountBalanceAllCorrect()
{
    User expected = ServicesTestConstant.getAddreser();
    User setIn = ServicesTestConstant.getAddreser();
    setIn.UserMoneyInCents = 0L;
    expected.UserMoneyInCents = 10L;
    long paymentSum = 10L;
    userRepository.Setup( expression: s => s.FindByName(It.IsAny<string>()))
        .Returns(setIn);

    User result = _userService.ReplenishAccountBalance(expected.UserName, paymentSum);

    userRepository.Verify(
        expression: place =>
            place.Save(),
        Times.Once());

    Assert.AreEqual(expected,  actual: result);
    Assert.AreEqual( expected: 10L,  actual: setIn.UserMoneyInCents);
}
```

Рисунок 1.21 – Перевірка коректності поповнення рахунку при коректних вхідних даних.

```
[Test]
public void replenishAccountBalanceNoSuchUser()
{
    userRepository.Setup( expression: s => s.FindByName(It.IsAny<string>()))
        .Returns((User) null);

    var actualResult =
        Assert.Throws<NoSuchUserException>( code: () =>
            _userService.ReplenishAccountBalance(ServicesTestConstant.getUserId(),  amountMoney: 10));

    Assert.AreEqual( expected: typeof(NoSuchUserException), actualResult.GetType());
}
```

Рисунок 1.22 – Перевірка виникнення виключення «NoSuchUserException» у випадку якщо заданого користувача не існує.

```
[Test]
public void replenishAccountBalanceToMuchMoneyException()
{
    User user = new User( userName: null,  password: null,  userMoneyInCents: long.MaxValue);
    userRepository.Setup( expression: s => s.FindByName(It.IsAny<string>()))
        .Returns(user);

    var actualResult =
        Assert.Throws<ToMuchMoneyException>( code: () =>
            _userService.ReplenishAccountBalance(ServicesTestConstant.getUserId(),  amountMoney: 10));
    Assert.AreEqual( expected: typeof(ToMuchMoneyException), actualResult.GetType());
}
```

Рисунок 1.23 – Перевірка виникнення «ToMuchMoneyException» виключення у випадку поповнення на занадто велику суму.

ВИСНОВКИ

В дані роботі було повністю створено протестовано всю значиму логіку додатку. Додаток було спроектовано коректно, з повним дотриманням принципів солід та модульності це дозволило уникнути необхідності в рефакторингу коду перед та у процесі тестування.

В звіті було наведено тести ключових методів реалізують основний функціонал додатку. Всі тести можна знайти у репозиторії проекту за адресою «https://github.com/VINIPOOH/delivery_dot_net».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Ноубл, Дж., Андерсон, Т., Брэйтуэйт, Г., Казарио, М., Третола, Р. Flex 4. Рецепты программирования. — БХВ-Петербург, 2011. — С. 548. — 720 с

2) Самоучитель UML 2. – СПб.: БХВ-Петербург, 2007. – 567 с.: ил. ISBN 978-5-94157-878-8

3) Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. П75 Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2001. — 368 с.: ил. (Серия «Библиотека программиста») ISBN 5-272-00355-1

4) Мартин Фаулер., Чистий код: создание, анализ и рефакторинг. – СПб.: Питер, 2019. – 464 с.: ил.