КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

«Формальні методи розробки програмних систем»

Формальна специфікація та верифікація системи «Паркомат»

Виконала:
студентка 1 курсу магістратури
групи ШІ
Кіндякова Діана Валеріївна

Формальна специфікація системи «Паркомат»

Був використаний **моделе-орієнтовний метод** специфікації, тобто описано аспекти специфікуємої системи за допомогою аксіом, що властиві специфікаціям, орієнтованим на властивостям.

Тип специфікації був обраний **поведінковий**, тобто було описане обмеження на поведінку об’єкта специфікації, у моєму випадку – Паркомат, а саме на функціональні можливості, безпеку та виконання. Також даний тип специфікації можна частково віднести до **структурного**, оскільки у цій системі був описаний внутрішній склад об’єкту специфікації, а саме водій, пасажири і так далі.

Перейдемо до опису нашої системи «Паркомат».

Серед самих практичних функцій, які можна описати для нашої системи, було обрано:

- **Функція «Прийом оплати»**
- **Функція «Видача квитанції»**
- **Функція «Перевірка оплати»**

Також визначимо певні умови, якими будуть керуватись дані функції:

- Не можна приймати оплату картою, якщо термінал не працює.
- Не можна приймати оплату, якщо не обрано тривалість паркування.
- Тривалість паркування має бути більша за 0
- Не можна приймати оплату, якщо не обрано номер транспортного засобу.
- Не можна видати квитанцію, якщо не була здійснена оплата.
- Не можна оплатити менше вартості паркування.
- Решта при оплаті готівкою не видається.
- Внесена сума не може бути від’ємним числом.
- Для кожного транспортного засобу може одночасно існувати лише одна діюча квитанція.
- Якщо оплачується паркування, для транспорту, що вже має діючу квитанцію, нова квитанція видається від часу завершення останньої існуючої квитанції.

- Квитанція вважається діючою, якщо час початку паркування разом з тривалістю менший за поточний час.
- Оплата для транспортного засобу підтверджена, лише якщо для нього існує квитанція.

Зібравши всі необхідні умови, можемо приступити до формального опису функції за допомогою псевдо-коду:

---

1. **Функція: Прийняти оплату**

Вхідні дані (Чи працює термінал, чи вибрано тривалість паркування, тривалість паркування, чи введено номер транспортного засобу, номер транспортного засобу, вид оплати (готівка/карта), внесена сума, тариф, квитанції, поточний час)

Логіка виконання:
Якщо термінал не працює або не вибрано тривалість паркування або не введено номер транспортного засобу, або тривалість паркування не більша за 0

- Прийняти оплату неможливо. Повернути гроші.

Якщо сума менша за вартість паркування (тариф * тривалість)

- Прийняти оплату неможливо. Повернути гроші.

Якщо оплата карткою і термінал не працює

- Прийняти оплату неможливо.

Якщо транспортний засіб уже має діючу квитанцію

- Прийняти оплату. Додаємо внесену суму до загальної оплати
- Видаємо квитанцію (Так, номер транспортного засобу, тривалість паркування, час завершення останньої існуючої квитанції)

Інакше

- Прийняти оплату. Додаємо внесену суму до загальної оплати
- Видаємо нову квитанцію(Так, номер транспортного засобу, тривалість паркування, поточний час)

Функція: Видати квитанцію

Вхідні дані: (Чи була здійснена оплата, номер транспортного засобу, тривалість паркування, час початку)

Логіка виконання:
Якщо оплата не здійснена

- Видати квитанцію неможливо.

Генеруємо квитанцію:

- Вказуємо номер транспортного засобу
- Вказуємо тривалість паркування
- Вказуємо час початку
- Зберігаємо квитанцію в системі

Функція: Перевірити оплату

Вхідні дані: (Номер транспортного засобу, поточний час, квитанції)

Логіка виконання:
Якщо для даного транспортного засобу немає діючої квитанції

- Оплата не підтверджена

Інакше

- Оплата підтверджена

Після того, як ми описали методи, можна перейти до їхньої реалізації.

Короткий опис проекту:

- **Мова програмування:** Dafny
- **Середовище розробки:** VS Code

Реалізація проекту:

У основу всього проекту ліг створений загальний клас, що має назву

ParkingMeter, у ньому зберігається інформація про минулі, дійсні та майбутні квитанції для кожного транспортного засобу, а також загальна сума грошей, що заплатили за паркування.

Та допоміжний клас, для збереження інформації про квитанції (часу початку, та тривалість паркування)

```
class ParkingMeter {

    var receipts: map<string, seq<Receipt>> // Зберігає квитанції: номер транспорту -> список (час початку, тривалість)
    var totalEarnings: int // Загальна сума заробітку
    var ratePerHour: int // Тариф за годину
```

```
class Receipt{
    var startTime : int
    var duration : int

    constructor(startTime: int, duration: int)
        ensures this.startTime == startTime
        ensures this.duration == duration
    {
        this.startTime := startTime;
        this.duration := duration;
    }
}
```

Конструктор, в який ми передаємо початкові значення для цих полів:

```
constructor(rate: int)
    ensures ratePerHour == rate
    ensures totalEarnings == 0
    ensures receipts == map[]
{
    ratePerHour := rate;
    totalEarnings := 0;
    receipts := map[];
    print"Паркометр ініціалізовано з тарифом за годину: ", rate, "\n";
}
```

1. Метод: «Видати квитанцію»

```
method IssueReceipt(terminalWorks : bool, vehicleNumber: string, duration: int,
                    paymentType:string, amountPaid : int, currentTime : int)  returns (success: bool, ghost paymentSuccessful: bool)
    requires duration > 0
    requires amountPaid >= 0
    modifies this
    ensures paymentSuccessful ==> success
    ensures success ==> paymentSuccessful == true
    ensures success ==> vehicleNumber in this.receipts && exists t:
            Receipt :: t in this.receipts[vehicleNumber] && t.startTime + t.duration > currentTime
{
    print "Видача квитанції на паркування для транспорту: ", vehicleNumber, "\n";
    var paymentMade := AcceptPayment(terminalWorks, vehicleNumber, duration, paymentType, amountPaid, currentTime);
    assert (terminalWorks || paymentType !=  "card") && (amountPaid >= this.ratePerHour * duration) ==> paymentMade;
    paymentSuccessful := paymentMade;
    if (!paymentMade) {
        print "Оплата не прийнята для транспорту: ", vehicleNumber, "\n";
        var success := false;
        return success, paymentSuccessful;
    }
    assert paymentSuccessful == true;
    var newStartTime: int := currentTime;
    if vehicleNumber in this.receipts && |this.receipts[vehicleNumber]| > 0 {
        var lastReceipt := this.receipts[vehicleNumber][|this.receipts[vehicleNumber]| - 1];
        if lastReceipt.startTime + lastReceipt.duration > currentTime {
            newStartTime := lastReceipt.startTime + lastReceipt.duration;
            print "Корегуємо час початку для транспорту: ", vehicleNumber, " на ", newStartTime, "\n";
        }
    }
    var newReceipt := new Receipt(newStartTime, duration);
    print "Створена нова квитанція: \n";
    if (vehicleNumber in this.receipts) {
        this.receipts := this.receipts[vehicleNumber := this.receipts[vehicleNumber] + [newReceipt]];
        print "Додано квитанцію для транспорту: ", vehicleNumber, "\n";
    } else {
        this.receipts :=  this.receipts[vehicleNumber := [newReceipt]];
        print "Створено квитанцію для транспорту: ", vehicleNumber, "\n";
    }
    return true, paymentSuccessful;
}
```

2. Метод: «Перевірити оплату»

```
// Метод для прийняття оплати
method AcceptPayment(terminalWorks: bool, vehicleNumber: string, duration: int,
                     paymentType: string, amountPaid: int, currentTime: int) returns (success: bool)
    requires duration > 0
    requires amountPaid >= 0
    modifies this
    ensures (terminalWorks || paymentType !=  "card") && (amountPaid >= this.ratePerHour * duration) ==> success
    ensures success ==> totalEarnings >= old(totalEarnings) + this.ratePerHour * duration
{
    print "Прийняття оплати для транспорту: ", vehicleNumber, "\n";

    if (!terminalWorks && paymentType == "card") {
        print "Термінал не працює для оплати карткою.\n";
        return false;
    }

    var cost := this.ratePerHour * duration;
    if (amountPaid < cost) {
        print "Недостатня оплата для транспорту: ", vehicleNumber, "\n";
        return false;
    }

    this.totalEarnings := this.totalEarnings + amountPaid;
    print "Оплата прийнята. Загальні заробітки оновлені: ", this.totalEarnings, "\n";
    return true;
}
```

3. Метод: «Перевірити оплату»

```
// Метод для перевірки, чи було сплачено
method VerifyPayment(vehicleNumber: string, currentTime: int) returns (isPaid: bool)
    ensures isPaid ==> vehicleNumber in this.receipts && exists t:
            Receipt :: t in this.receipts[vehicleNumber] && t.startTime + t.duration > currentTime
{
    print "Перевірка оплати для транспорту: ", vehicleNumber, "\n";

    if vehicleNumber in this.receipts {
        var receiptsList := this.receipts[vehicleNumber];
        var n := |receiptsList|;
        var i := 0;

        while i < n {
            var receipt := receiptsList[i];
            if receipt.startTime + receipt.duration > currentTime {
                print "Знайдена дійсна квитанція для транспорту: ", vehicleNumber, "\n";
                return true;
            }
            i := i + 1;
        }
    }

    print "Дійсна оплата не знайдена для транспорту: ", vehicleNumber, "\n";
    return false;
}
```

Для симуляції поведінки цієї системи напишемо окремо функцію Main, що буде вхідною точкою для нашого проекту

```
method Main(){
    var rate := 10; // Тариф за годину
    var parkingMeter := new ParkingMeter(rate);

    // Видаємо квитанцію для транспорту
    var vehicleNumber := "AB1234CD";
    var duration := 2; // тривалість паркування в годинах
    var paymentType := "card"; // тип оплати
    var amountPaid := 20; // сума оплати
    var currentTime := 10; // поточний час (час початку паркування)

    // Демонстрація видачі квитанції
    print "Демо 1: Видача квитанції\n";
    var success := parkingMeter.IssueReceipt(true, vehicleNumber, duration, paymentType, amountPaid, currentTime);
    print("Квитанція видана?: ", success);

    // Перевіряємо оплату для транспорту
    print "\nДемо 2: Перевірка оплати\n";
    var isPaid := parkingMeter.VerifyPayment(vehicleNumber, currentTime + 1); // Перевірка через 1 годину після початку
    print "Оплата підтверджена: ", isPaid, "\n";

    // Додаємо ще одну квитанцію для того ж транспорту
    print "\nДемо 3: Додавання нової квитанції\n";
    var newDuration := 3; // нова тривалість
    var newAmountPaid := 200; // нова сума оплати
    var newCurrentTime := currentTime + 1;
    var newPaymentType := "cash";
    var newSuccess := parkingMeter.IssueReceipt(true, vehicleNumber, newDuration, newPaymentType, newAmountPaid, newCurrentTime);
    print "Нова квитанція видана?: ", newSuccess, "\n";
```

```
    // Перевіряємо оплату після додавання нової квитанції
    print "\nДемо 4: Перевірка оплати після додавання нової квитанції\n";
    var newIsPaid := parkingMeter.VerifyPayment(vehicleNumber, currentTime + 4); // Перевірка через 4 години після початку
    print "Оплата підтверджена для нової квитанції: ", newIsPaid, "\n";

    // Перевіряємо загальні заробіток
    print "\nДемо 5: Загальні заробітое паркометра\n";
    print "Загальні заробітки: ", parkingMeter.totalEarnings, "\n";
}
```

Результати:

```
 Dafny program verifier finished with 9 verified, 0 errors
```

```
Паркометр ініціалізовано з тарифом за годину: 10
Демо 1: Видача квитанції
Видача квитанції на паркування для транспорту: AB1234CD
Прийняття оплати для транспорту: AB1234CD
Оплата прийнята. Загальні заробітки оновлені: 20
Створена нова квитанція:
Створено квитанцію для транспорту: AB1234CD
Квитанц?я видана?: true

Демо 2: Перевірка оплати
Перевірка оплати для транспорту: AB1234CD
Знайдена дійсна квитанція для транспорту: AB1234CD
Оплата п?дтверджена: true

Демо 3: Додавання нової квитанції
Видача квитанції на паркування для транспорту: AB1234CD
Прийняття оплати для транспорту: AB1234CD
Оплата прийнята. Загальні заробітки оновлені: 50
Коригуємо час початку для транспорту: AB1234CD на 12
Створена нова квитанція:
Додано квитанцію для транспорту: AB1234CD
Нова квитанція видана?: true

Демо 4: Перевірка оплати після додавання нової квитанції
Перевірка оплати для транспорту: AB1234CD
Знайдена дійсна квитанція для транспорту: AB1234CD
Оплата підтверджена для нової квитанції: true

Демо 5: Загальні заробітое паркометра
Загальні заробітки: 50
```

Висновок:

Завдяки використанню формальних специфікацій та визначення строгих умов для функціональних можливостей паркомата, таких як прийом оплати, видача квитанцій та перевірка оплати, вдалося розробити систему, що гарантує коректну роботу в різних сценаріях. Важливим етапом у розробці стало також

забезпечення безпеки системи — всі умови для успішної операції чітко визначені та перевіряються на кожному етапі.

Реалізація в мові Dafny дозволяє не тільки реалізувати саму систему, але й формально довести її правильність, що є ключовим аспектом для програмного забезпечення, яке працює в умовах високої надійності, таких як платіжні системи.