**Лабораторная работа №2**

**Задание для выполнения**

Разработать программу, реализующую решение одной из следующих задач согласно варианту с визуализацией результатов работы. В программе должно быть предусмотрено сравнение выполнения задания с синхронизацией потоков и без них. Имитация снятия денег со счета несколькими клиентами для наличия положительного баланса.

**Объяснение:**

Основные компоненты программы:

1. Класс BankAccount

* takeMoneySync() - синхронизированное снятие денег с использованием мьютекса
* takeMoneyNoSync() - несинхронизированное снятие денег
* checkBalance() - проверка текущего баланса
* Ведение статистики операций и логирование

2. Класс Client

* Представляет клиента банка
* Каждый клиент пытается снять максимально возможную сумму
* Отслеживание статуса (ожидание/работа/завершено)
* Визуальное отображение прогресса каждого клиента

3. Класс BankingSystemMutex

* Точка входа в программу
* Управление симуляцией (запуск/остановка/сброс)
* Координация работы всех клиентов
* Обновление интерфейса в реальном времени

Ключевые особенности:

С синхронизацией:

* Использует мьютекс для предотвращения одновременного доступа
* Гарантирует, что баланс не уйдет в минус
* Более медленная обработка, но корректные результаты

Без синхронизации:

* Прямой доступ к балансу без блокировок
* Может привести к отрицательному балансу из-за гонки потоков
* Быстрая обработка, но некорректные результаты

**Решение:**

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Симуляция банковского счета</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

        }

        body {

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

            min-height: 100vh;

            padding: 20px;

            color: #333;

        }

        .container {

            max-width: 1200px;

            margin: 0 auto;

            background: rgba(255, 255, 255, 0.95);

            border-radius: 20px;

            padding: 30px;

            box-shadow: 0 20px 40px rgba(0, 0, 0, 0.1);

        }

        h1 {

            text-align: center;

            color: #2c3e50;

            margin-bottom: 30px;

            font-size: 2.5rem;

            background: linear-gradient(135deg, #667eea, #764ba2);

            -webkit-background-clip: text;

            -webkit-text-fill-color: transparent;

            background-clip: text;

        }

        .controls {

            display: flex;

            gap: 20px;

            margin-bottom: 30px;

            flex-wrap: wrap;

            align-items: center;

            justify-content: center;

        }

        .input-group {

            display: flex;

            flex-direction: column;

            gap: 5px;

        }

        label {

            font-weight: 600;

            color: #555;

            font-size: 0.9rem;

        }

        input[type="number"] {

            padding: 12px;

            border: 2px solid #e0e0e0;

            border-radius: 10px;

            font-size: 16px;

            transition: all 0.3s ease;

            width: 150px;

        }

        input[type="number"]:focus {

            outline: none;

            border-color: #667eea;

            box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);

        }

        .buttons {

            display: flex;

            gap: 15px;

            flex-wrap: wrap;

        }

        button {

            padding: 12px 25px;

            border: none;

            border-radius: 10px;

            font-size: 16px;

            font-weight: 600;

            cursor: pointer;

            transition: all 0.3s ease;

            text-transform: uppercase;

            letter-spacing: 0.5px;

        }

        .btn-primary {

            background: linear-gradient(135deg, #667eea, #764ba2);

            color: white;

        }

        .btn-primary:hover {

            transform: translateY(-2px);

            box-shadow: 0 10px 20px rgba(102, 126, 234, 0.3);

        }

        .btn-secondary {

            background: linear-gradient(135deg, #f093fb, #f5576c);

            color: white;

        }

        .btn-secondary:hover {

            transform: translateY(-2px);

            box-shadow: 0 10px 20px rgba(245, 87, 108, 0.3);

        }

        .btn-danger {

            background: linear-gradient(135deg, #ff6b6b, #ee5a52);

            color: white;

        }

        .btn-danger:hover {

            transform: translateY(-2px);

            box-shadow: 0 10px 20px rgba(238, 90, 82, 0.3);

        }

        button:disabled {

            opacity: 0.6;

            cursor: not-allowed;

            transform: none !important;

        }

        .simulation-area {

            display: grid;

            grid-template-columns: 1fr 1fr;

            gap: 30px;

            margin-bottom: 30px;

        }

        .simulation-panel {

            background: white;

            border-radius: 15px;

            padding: 25px;

            box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);

            border: 2px solid #f0f0f0;

        }

        .simulation-panel h3 {

            margin-bottom: 20px;

            color: #2c3e50;

            font-size: 1.4rem;

            text-align: center;

        }

        .sync-panel {

            border-left: 4px solid #27ae60;

        }

        .no-sync-panel {

            border-left: 4px solid #e74c3c;

        }

        .balance-display {

            text-align: center;

            margin-bottom: 20px;

        }

        .balance {

            font-size: 2rem;

            font-weight: bold;

            margin: 10px 0;

            padding: 15px;

            border-radius: 10px;

            background: linear-gradient(135deg, #f8f9fa, #e9ecef);

        }

        .positive {

            color: #27ae60;

            background: linear-gradient(135deg, #d4edda, #c3e6cb);

        }

        .negative {

            color: #e74c3c;

            background: linear-gradient(135deg, #f8d7da, #f5c6cb);

        }

        .clients-container {

            max-height: 300px;

            overflow-y: auto;

            border: 1px solid #e0e0e0;

            border-radius: 10px;

            padding: 15px;

            background: #fafafa;

        }

        .client {

            display: flex;

            justify-content: space-between;

            align-items: center;

            padding: 10px;

            margin: 5px 0;

            background: white;

            border-radius: 8px;

            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

            transition: all 0.3s ease;

        }

        .client:hover {

            transform: translateX(5px);

            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.15);

        }

        .client-info {

            display: flex;

            align-items: center;

            gap: 10px;

        }

        .client-avatar {

            width: 40px;

            height: 40px;

            border-radius: 50%;

            background: linear-gradient(135deg, #667eea, #764ba2);

            display: flex;

            align-items: center;

            justify-content: center;

            color: white;

            font-weight: bold;

        }

        .client-status {

            font-size: 0.9rem;

            padding: 5px 10px;

            border-radius: 15px;

            font-weight: bold;

        }

        .working {

            background: #fff3cd;

            color: #856404;

        }

        .waiting {

            background: #d1ecf1;

            color: #0c5460;

        }

        .finished {

            background: #d4edda;

            color: #155724;

        }

        .stats {

            background: white;

            border-radius: 15px;

            padding: 25px;

            box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);

            margin-top: 20px;

        }

        .stats h3 {

            color: #2c3e50;

            margin-bottom: 20px;

            text-align: center;

            font-size: 1.4rem;

        }

        .stats-grid {

            display: grid;

            grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

            gap: 20px;

        }

        .stat-item {

            text-align: center;

            padding: 20px;

            background: linear-gradient(135deg, #f8f9fa, #e9ecef);

            border-radius: 10px;

            transition: all 0.3s ease;

        }

        .stat-item:hover {

            transform: translateY(-5px);

            box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);

        }

        .stat-label {

            font-weight: 600;

            color: #666;

            margin-bottom: 5px;

        }

        .stat-value {

            font-size: 1.5rem;

            font-weight: bold;

            color: #2c3e50;

        }

        .log-container {

            background: white;

            border-radius: 15px;

            padding: 25px;

            box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);

            margin-top: 20px;

        }

        .log-container h3 {

            color: #2c3e50;

            margin-bottom: 20px;

            text-align: center;

        }

        .log {

            max-height: 200px;

            overflow-y: auto;

            border: 1px solid #e0e0e0;

            border-radius: 10px;

            padding: 15px;

            background: #fafafa;

            font-family: 'Courier New', monospace;

            font-size: 0.9rem;

        }

        .log-entry {

            margin: 5px 0;

            padding: 5px 10px;

            border-radius: 5px;

            background: white;

            border-left: 3px solid #667eea;

        }

        .progress-bar {

            width: 100%;

            height: 8px;

            background: #e0e0e0;

            border-radius: 4px;

            overflow: hidden;

            margin: 10px 0;

        }

        .progress-fill {

            height: 100%;

            background: linear-gradient(90deg, #667eea, #764ba2);

            transition: width 0.3s ease;

            border-radius: 4px;

        }

        @media (max-width: 768px) {

            .simulation-area {

                grid-template-columns: 1fr;

            }

            .controls {

                flex-direction: column;

                align-items: stretch;

            }

            .buttons {

                justify-content: center;

            }

            h1 {

                font-size: 2rem;

            }

        }

    </style>

</head>

<body>

    <div class="container">

        <h1>💰 Симуляция банковского счета</h1>

        <div class="controls">

            <div class="input-group">

                <label for="initialBalance">Начальный баланс:</label>

                <input type="number" id="initialBalance" value="10000" min="1000" max="100000">

            </div>

            <div class="input-group">

                <label for="clientCount">Количество клиентов:</label>

                <input type="number" id="clientCount" value="5" min="2" max="20">

            </div>

            <div class="input-group">

                <label for="maxWithdraw">Макс. снятие:</label>

                <input type="number" id="maxWithdraw" value="500" min="100" max="2000">

            </div>

            <div class="buttons">

                <button class="btn-primary" onclick="startSimulation()">🚀 Запустить симуляцию</button>

                <button class="btn-secondary" onclick="resetSimulation()">🔄 Сбросить</button>

                <button class="btn-danger" onclick="stopSimulation()">⏹️ Остановить</button>

            </div>

        </div>

        <div class="simulation-area">

            <div class="simulation-panel sync-panel">

                <h3>🔒 С синхронизацией</h3>

                <div class="balance-display">

                    <div>Баланс счета:</div>

                    <div class="balance positive" id="syncBalance">0 ₽</div>

                    <div class="progress-bar">

                        <div class="progress-fill" id="syncProgress"></div>

                    </div>

                </div>

                <div class="clients-container" id="syncClients"></div>

            </div>

            <div class="simulation-panel no-sync-panel">

                <h3>🔓 Без синхронизации</h3>

                <div class="balance-display">

                    <div>Баланс счета:</div>

                    <div class="balance positive" id="noSyncBalance">0 ₽</div>

                    <div class="progress-bar">

                        <div class="progress-fill" id="noSyncProgress"></div>

                    </div>

                </div>

                <div class="clients-container" id="noSyncClients"></div>

            </div>

        </div>

        <div class="stats">

            <h3>📊 Статистика сравнения</h3>

            <div class="stats-grid">

                <div class="stat-item">

                    <div class="stat-label">Итоговый баланс (синхр.)</div>

                    <div class="stat-value" id="finalSyncBalance">0 ₽</div>

                </div>

                <div class="stat-item">

                    <div class="stat-label">Итоговый баланс (без синхр.)</div>

                    <div class="stat-value" id="finalNoSyncBalance">0 ₽</div>

                </div>

                <div class="stat-item">

                    <div class="stat-label">Всего операций (синхр.)</div>

                    <div class="stat-value" id="syncOperations">0</div>

                </div>

                <div class="stat-item">

                    <div class="stat-label">Всего операций (без синхр.)</div>

                    <div class="stat-value" id="noSyncOperations">0</div>

                </div>

            </div>

        </div>

        <div class="log-container">

            <h3>📝 Лог операций</h3>

            <div class="log" id="operationLog"></div>

        </div>

    </div>

    <script>

        // Классы для симуляции банковской системы

        class BankAccount {

            constructor(initialBalance) {

                this.balance = initialBalance;

                this.initialBalance = initialBalance;

                this.operationCount = 0;

                this.mutex = false; // Простая реализация мьютекса

            }

            // Синхронизированное снятие денег

            async takeMoneySync(amount, clientId) {

                // Ожидание освобождения мьютекса

                while (this.mutex) {

                    await this.sleep(10);

                }

                this.mutex = true; // Захват мьютекса

                try {

                    await this.sleep(50 + Math.random() \* 100); // Имитация времени обработки

                    if (this.balance >= amount) {

                        this.balance -= amount;

                        this.operationCount++;

                        this.logOperation(`Клиент ${clientId} снял ${amount} ₽ (синхр.). Баланс: ${this.balance} ₽`, 'sync');

                        return true;

                    } else {

                        this.logOperation(`Клиент ${clientId} не смог снять ${amount} ₽. Недостаточно средств (синхр.)`, 'sync');

                        return false;

                    }

                } finally {

                    this.mutex = false; // Освобождение мьютекса

                }

            }

            // Несинхронизированное снятие денег

            async takeMoneyNoSync(amount, clientId) {

                await this.sleep(50 + Math.random() \* 100); // Имитация времени обработки

                if (this.balance >= amount) {

                    const oldBalance = this.balance;

                    this.balance -= amount;

                    this.operationCount++;

                    this.logOperation(`Клиент ${clientId} снял ${amount} ₽ (без синхр.). Баланс: ${this.balance} ₽`, 'nosync');

                    return true;

                } else {

                    this.logOperation(`Клиент ${clientId} не смог снять ${amount} ₽. Недостаточно средств (без синхр.)`, 'nosync');

                    return false;

                }

            }

            checkBalance() {

                return this.balance;

            }

            logOperation(message, type) {

                const log = document.getElementById('operationLog');

                const entry = document.createElement('div');

                entry.className = 'log-entry';

                entry.style.borderLeftColor = type === 'sync' ? '#27ae60' : '#e74c3c';

                entry.textContent = `${new Date().toLocaleTimeString()}: ${message}`;

                log.appendChild(entry);

                log.scrollTop = log.scrollHeight;

            }

            sleep(ms) {

                return new Promise(resolve => setTimeout(resolve, ms));

            }

        }

        class Client {

            constructor(id, maxWithdraw) {

                this.id = id;

                this.maxWithdraw = maxWithdraw;

                this.totalWithdrawn = 0;

                this.operationCount = 0;

                this.status = 'waiting';

                this.isActive = false;

            }

            async startWithdrawing(account, isSync = true) {

                this.isActive = true;

                this.status = 'working';

                this.updateClientDisplay(isSync);

                while (this.isActive && account.checkBalance() > 0) {

                    const withdrawAmount = Math.floor(Math.random() \* this.maxWithdraw) + 1;

                    let success;

                    if (isSync) {

                        success = await account.takeMoneySync(withdrawAmount, this.id);

                    } else {

                        success = await account.takeMoneyNoSync(withdrawAmount, this.id);

                    }

                    if (success) {

                        this.totalWithdrawn += withdrawAmount;

                        this.operationCount++;

                    }

                    this.updateClientDisplay(isSync);

                    await account.sleep(100 + Math.random() \* 200);

                }

                this.status = 'finished';

                this.isActive = false;

                this.updateClientDisplay(isSync);

            }

            stop() {

                this.isActive = false;

                this.status = 'finished';

            }

            updateClientDisplay(isSync) {

                const containerId = isSync ? 'syncClients' : 'noSyncClients';

                const container = document.getElementById(containerId);

                let clientElement = document.getElementById(`${containerId}-client-${this.id}`);

                if (!clientElement) {

                    clientElement = document.createElement('div');

                    clientElement.className = 'client';

                    clientElement.id = `${containerId}-client-${this.id}`;

                    container.appendChild(clientElement);

                }

                clientElement.innerHTML = `

                    <div class="client-info">

                        <div class="client-avatar">${this.id}</div>

                        <div>

                            <div><strong>Клиент ${this.id}</strong></div>

                            <div style="font-size: 0.8rem; color: #666;">

                                Снято: ${this.totalWithdrawn} ₽ (${this.operationCount} операций)

                            </div>

                        </div>

                    </div>

                    <div class="client-status ${this.status}">${this.getStatusText()}</div>

                `;

            }

            getStatusText() {

                switch (this.status) {

                    case 'waiting': return 'Ожидание';

                    case 'working': return 'Работает';

                    case 'finished': return 'Завершено';

                    default: return 'Неизвестно';

                }

            }

        }

        class BankingSystemMutex {

            constructor() {

                this.syncAccount = null;

                this.noSyncAccount = null;

                this.syncClients = [];

                this.noSyncClients = [];

                this.isRunning = false;

                this.simulationInterval = null;

            }

            async startSimulation() {

                if (this.isRunning) return;

                const initialBalance = parseInt(document.getElementById('initialBalance').value);

                const clientCount = parseInt(document.getElementById('clientCount').value);

                const maxWithdraw = parseInt(document.getElementById('maxWithdraw').value);

                // Инициализация счетов

                this.syncAccount = new BankAccount(initialBalance);

                this.noSyncAccount = new BankAccount(initialBalance);

                // Создание клиентов

                this.syncClients = [];

                this.noSyncClients = [];

                for (let i = 1; i <= clientCount; i++) {

                    this.syncClients.push(new Client(i, maxWithdraw));

                    this.noSyncClients.push(new Client(i, maxWithdraw));

                }

                this.isRunning = true;

                this.clearLogs();

                this.clearClientsDisplay();

                // Обновление интерфейса

                this.updateButtons();

                this.updateBalanceDisplay();

                // Запуск клиентов

                this.syncClients.forEach(client => {

                    client.startWithdrawing(this.syncAccount, true);

                });

                this.noSyncClients.forEach(client => {

                    client.startWithdrawing(this.noSyncAccount, false);

                });

                // Обновление дисплея каждые 100мс

                this.simulationInterval = setInterval(() => {

                    this.updateBalanceDisplay();

                    this.updateProgress();

                    this.checkSimulationEnd();

                }, 100);

            }

            stopSimulation() {

                if (!this.isRunning) return;

                this.isRunning = false;

                // Остановка всех клиентов

                this.syncClients.forEach(client => client.stop());

                this.noSyncClients.forEach(client => client.stop());

                if (this.simulationInterval) {

                    clearInterval(this.simulationInterval);

                    this.simulationInterval = null;

                }

                this.updateButtons();

                this.updateStats();

            }

            resetSimulation() {

                this.stopSimulation();

                this.clearLogs();

                this.clearClientsDisplay();

                this.resetStats();

                this.resetBalanceDisplay();

            }

            updateBalanceDisplay() {

                if (!this.syncAccount || !this.noSyncAccount) return;

                const syncBalance = document.getElementById('syncBalance');

                const noSyncBalance = document.getElementById('noSyncBalance');

                const syncValue = this.syncAccount.checkBalance();

                const noSyncValue = this.noSyncAccount.checkBalance();

                syncBalance.textContent = `${syncValue.toLocaleString()} ₽`;

                noSyncBalance.textContent = `${noSyncValue.toLocaleString()} ₽`;

                // Обновление цветов

                syncBalance.className = `balance ${syncValue >= 0 ? 'positive' : 'negative'}`;

                noSyncBalance.className = `balance ${noSyncValue >= 0 ? 'positive' : 'negative'}`;

            }

            updateProgress() {

                if (!this.syncAccount || !this.noSyncAccount) return;

                const syncProgress = document.getElementById('syncProgress');

                const noSyncProgress = document.getElementById('noSyncProgress');

                const syncPercent = Math.max(0, (this.syncAccount.checkBalance() / this.syncAccount.initialBalance) \* 100);

                const noSyncPercent = Math.max(0, (this.noSyncAccount.checkBalance() / this.noSyncAccount.initialBalance) \* 100);

                syncProgress.style.width = `${syncPercent}%`;

                noSyncProgress.style.width = `${noSyncPercent}%`;

            }

            checkSimulationEnd() {

                if (!this.isRunning) return;

                const syncFinished = this.syncClients.every(client => !client.isActive);

                const noSyncFinished = this.noSyncClients.every(client => !client.isActive);

                if (syncFinished && noSyncFinished) {

                    this.stopSimulation();

                }

            }

            updateButtons() {

                const startBtn = document.querySelector('.btn-primary');

                const stopBtn = document.querySelector('.btn-danger');

                const resetBtn = document.querySelector('.btn-secondary');

                startBtn.disabled = this.isRunning;

                stopBtn.disabled = !this.isRunning;

                resetBtn.disabled = this.isRunning;

            }

            updateStats() {

                if (!this.syncAccount || !this.noSyncAccount) return;

                document.getElementById('finalSyncBalance').textContent =

                    `${this.syncAccount.checkBalance().toLocaleString()} ₽`;

                document.getElementById('finalNoSyncBalance').textContent =

                    `${this.noSyncAccount.checkBalance().toLocaleString()} ₽`;

                document.getElementById('syncOperations').textContent =

                    this.syncAccount.operationCount;

                document.getElementById('noSyncOperations').textContent =

                    this.noSyncAccount.operationCount;

            }

            resetStats() {

                document.getElementById('finalSyncBalance').textContent = '0 ₽';

                document.getElementById('finalNoSyncBalance').textContent = '0 ₽';

                document.getElementById('syncOperations').textContent = '0';

                document.getElementById('noSyncOperations').textContent = '0';

            }

            resetBalanceDisplay() {

                document.getElementById('syncBalance').textContent = '0 ₽';

                document.getElementById('noSyncBalance').textContent = '0 ₽';

                document.getElementById('syncProgress').style.width = '100%';

                document.getElementById('noSyncProgress').style.width = '100%';

            }

            clearLogs() {

                document.getElementById('operationLog').innerHTML = '';

            }

            clearClientsDisplay() {

                document.getElementById('syncClients').innerHTML = '';

                document.getElementById('noSyncClients').innerHTML = '';

            }

        }

        // Глобальный объект системы

        const bankingSystem = new BankingSystemMutex();

        // Функции для кнопок

        function startSimulation() {

            bankingSystem.startSimulation();

        }

        function stopSimulation() {

            bankingSystem.stopSimulation();

        }

        function resetSimulation() {

            bankingSystem.resetSimulation();

        }

        // Инициализация при загрузке страницы

        document.addEventListener('DOMContentLoaded', function() {

            bankingSystem.updateButtons();

            bankingSystem.resetBalanceDisplay();

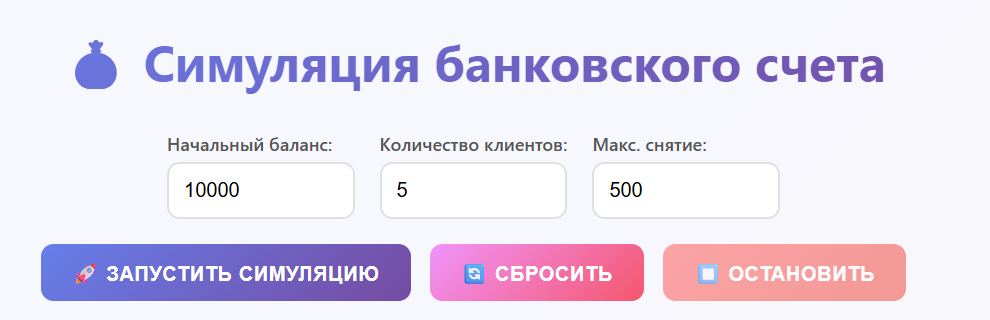
        });

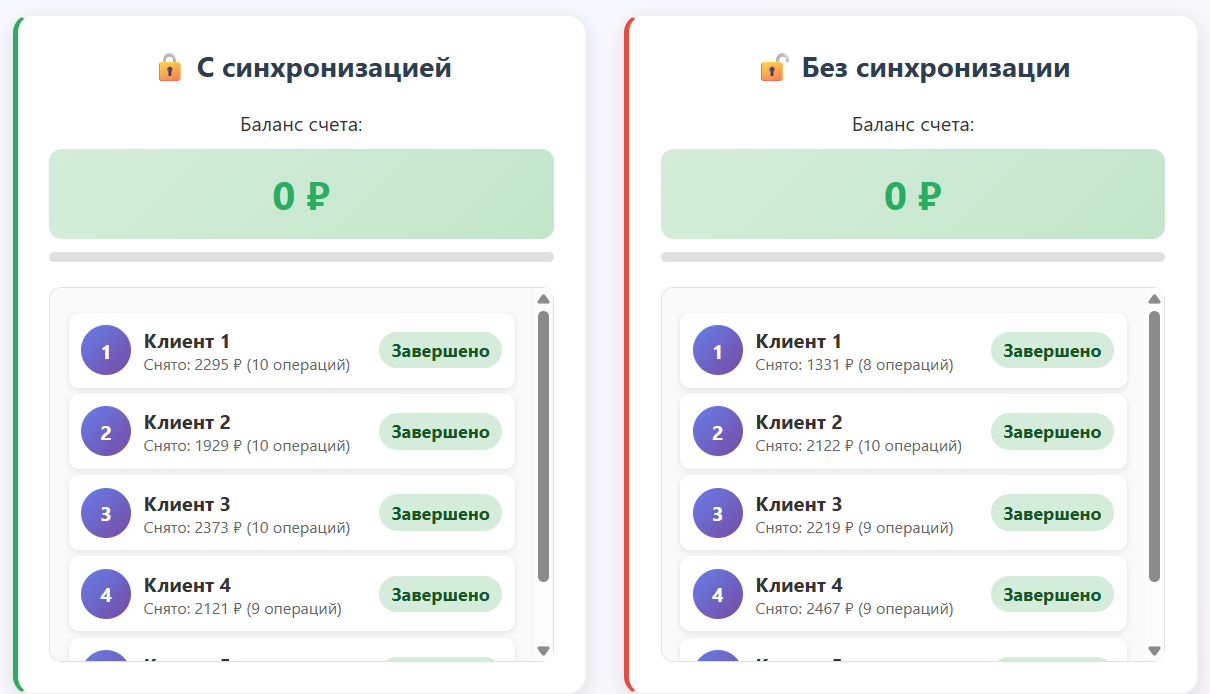
    </script>

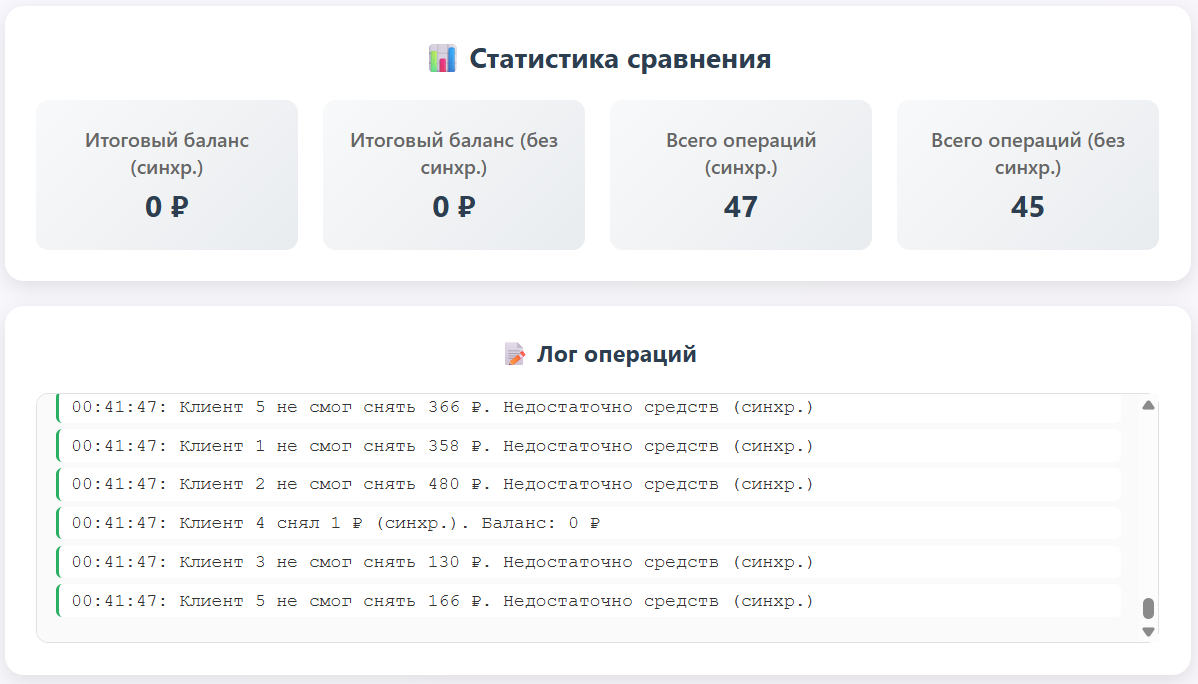
</body>

</html>

**Результат:**

****

****

****