Алгоритм

Для взаємодії по протоколу TCP/IP було використано інтерфейс сокетів. Термін «сокет» (socket) позначає одночасно бібліотеку мережевих інтерфейсів і термінал каналу зв’язку (точку зв’язку), через який процес може передавати або отримувати дані. TCP/IP-сокети дозволяють реалізувати надійні двонаправлені, орієнтовані на роботу з потоками з’єднання точка-точка між віддаленими вузлами. В цілому алгоритм роботи системи клієнт-сервер виглядає наступним чином:

1. Сервер підключається до порту на хості і чекає з’єднання з клієнтом
2. Клієнт створює сокет і намагається з’єднатися через нього з віддаленим сервером
3. У разі встановлення з’єднання, сервер отримує сокет для взаємодії з клієнтом і переходить в режим очікування команд від клієнта
4. Клієнт формує команду і передає її через сокет сервера, після чого переходить в режим очікування відповіді
5. Сервер приймає команду через сокет, виконує її і пересилає відповідь клієнту, після чого знову переходить в режим очікування запитів від клієнта

Отже, на стороні сервера необхідно почати прослуховувати певний порт TCP / IP, для того щоб віддалений клієнт міг виконувати підключення. Це було зроблено за допомогою класу java.net.ServerSocket. Спочатку, було створено екземпляр класу ServerSocket і вказано в конструкторі порт, який буде прослуховуватися. Якщо заданий порт зайнятий, генерувався виняток IOException. Далі, сервер переводився в режим очікування підключень. Робилося це за допомогою методу accept() класу ServerSocket. Метод accept очікує з’єднання з боку клієнта і повертає об’єкт класу java.net.Socket в разі успішної роботи. У разі помилки, метод генерує виняток IOException. Отриманий об’єкт java.net.Socket далі використовувався для прийому і передачі даних клієнту. На стороні клієнта здійснювалось підключення до сервера. Для цього використовувався клас java.net.Socket. На клієнті явно створювався об’єкт класу Socket і в конструктор передавалися установки сервера: IP-адреса сервера і порт. У разі помилки, конструктор генерував виняток IOException. Створений об’єкт java.net.Socket далі використовувався для прийому і передачі даних сервера.

Обмін даними між клієнтом і сервером здійснювався через об’єкти java.net.Socket. Після того, як з’єднання між клієнтом і сервером встановлювалось і обидві програми отримували свої об’єкти Socket, вони починали спілкуватися. Щоб передати інформацію на віддалений комп’ютер програмою, здійснювався запис цієї інформації в потік виводу сокета. Отримували потік виведення за допомогою методу getOutputStream об’єкта класу Socket. Щоб отримати інформацію, передану з боку віддаленої програми, виконувалось читання з потоку введення сокета. Потік введення повертався за допомогою методу getInputStream об’єкта класу Socket.

Для прийому/передачі текстових даних конструювались потоки BufferedReader і PrintWriter на основі потоків введення-виведення сокета (getInputStream () і getOutputStream ()). Для прийому/передачі двійкових даних необхідно сконструювати потоки DataInputStream і DataOutputStream на основі потоків введення-виведення сокета (getInputStream () і getOutputStream ()). Відповідно до першого варінту було обрано строку з роздільником в якості формату повідомлень. При використанні даного формату, всі поля інформаційного повідомлення записувалися в текстовому вигляді в один рядок, але розділені деяким службовим символом. Наприклад, в наступному рядку як роздільник використовується символ відсотка: «Ukraine#Poland#9#5».

При взаємодії за схемою «клієнт-сервер» ініціатором взаємодії виступає клієнт. Таким чином, сервер завжди повинен знаходитися в режимі очікування, так як сервер не знає, в який момент часу від клієнта надійде черговий запит. Клієнт відправляє серверу запит на виконання певної операції, і переходить в режим очікування відповіді. Сервер, отримує запит, виконує необхідну операцію і відправляє клієнту відповідь. Клієнт обробляє відповідь і продовжує своє нормально функціонування. Сервер ж, знову переходить в режим очікування запитів.

В даній програмі сервер перевіряв, чи вірна кількість параметрів прийшла йому на вхід, чи вірний тип вхідних параметрів, і чи правильно встановлено код операції. В якості роздільника використовувся символ «#».