Ще обясня как работи този клиент-сървър код последователно, като разделя изпълнението на две части: клиентската и сървърната страна.

**Как работи клиентът:**

1. **Вход от потребителя (getInput)**:
   * Клиентът изисква от потребителя да въведе:
     + Положително число.
     + Специални команди: "admin" за изключване на сървъра или "exit" за изход само на клиента.
   * Входът се валидира:
     + Ако числото е невалидно или не е положително, потребителят трябва да въведе отново.
     + Ако се въведе "admin", методът връща -2.
     + Ако се въведе "exit", методът връща -1.
2. **Изпращане на съобщения до сървъра**:
   * В зависимост от въведеното:
     + Ако числото е валидно, то се изпраща до сървъра.
     + Ако е "exit", клиентът изпраща "exit" съобщение до сървъра и затваря връзката.
     + Ако е "admin", клиентът изпраща "admin" съобщение, което указва на сървъра да се изключи.
3. **Получаване на отговори от сървъра**:
   * Клиентът чете отговорите от сървъра, докато не получи "EndOfSending".
   * Ако сървърът изпрати "admin" или "exit", клиентът затваря всички ресурси (сокет, стриймове) и прекратява изпълнението си.

**Как работи сървърът:**

1. **Очакване на връзка**:
   * Сървърът стартира и чака клиент да се свърже чрез метод accept().
2. **Получаване на съобщения от клиента**:
   * Сървърът чете входните данни от клиента чрез BufferedReader.
   * Обработва съобщението в зависимост от съдържанието му:
     + Ако е "exit", сървърът затваря връзката с клиента, но продължава да работи за други клиенти.
     + Ако е "admin", сървърът изпраща обратно "admin", затваря текущата връзка и спира изпълнението си.
     + Ако е число, извиква метода scraping, за да генерира резултат.
3. **Скрапинг (scraping метод)**:
   * Сървърът използва библиотеката **Jsoup**, за да изтегли данни от зададен уебсайт.
   * Извлича определен брой редове данни в зависимост от числото, което клиентът е изпратил.
4. **Изпращане на резултат към клиента**:
   * Сървърът изпраща резултата от скрапинга на клиента ред по ред.
   * Накрая изпраща "EndOfSending", за да сигнализира края на съобщението.
5. **Повтаряне на цикъла**:
   * Докато не се получи команда "admin" или "exit", сървърът продължава да приема съобщения от свързания клиент.

**Последователност на комуникация**

1. **Клиентът стартира и се свързва със сървъра**.
2. **Сървърът приема връзката и започва да слуша съобщенията от клиента**.
3. **Клиентът въвежда вход, изпраща го на сървъра, и сървърът обработва заявката**:
   * При положително число сървърът извършва скрапинг и връща резултати.
   * При "exit" сървърът прекратява само връзката с този клиент.
   * При "admin" сървърът спира напълно.
4. **Клиентът показва резултата и цикълът продължава, докато не се въведе "exit" или "admin"**.

**Какво е последователно в кода?**

* **Сървърът** обслужва един клиент за един момент, което означава, че докато една връзка не бъде затворена, други клиенти не могат да се свържат.
* **Клиентът** изчаква резултата от сървъра, преди да поиска нов вход.
* Липсата на многонишковост прави сървъра последователен (сериализиран), което може да доведе до забавяния при работа с множество клиенти.

**Примерна комуникация**

1. **Клиент**: Въвежда 3.
   * **Сървър**: Извършва скрапинг на 3 реда и връща резултата.
2. **Клиент**: Въвежда exit.
   * **Сървър**: Затваря връзката с клиента, но остава активен за други клиенти.
3. **Клиент 2**: Въвежда admin.
   * **Сървър**: Спира и уведомява клиента.