**Ders:** İleri Programlama (Fatih Aydın)

**Proje:** Server – Client haberleşmesi

**Katılımcılar:**

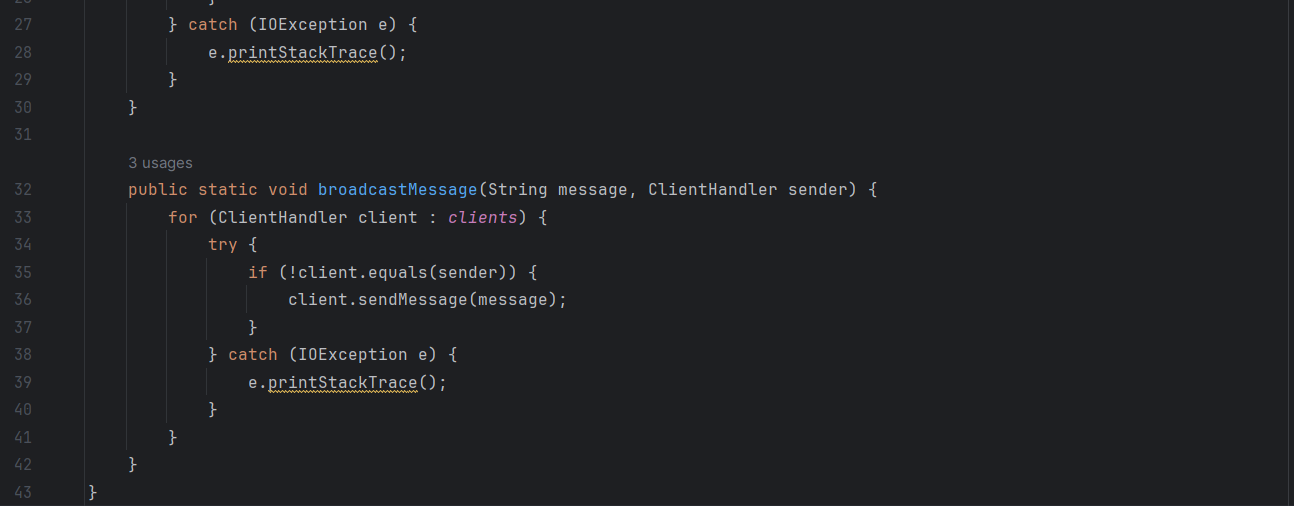
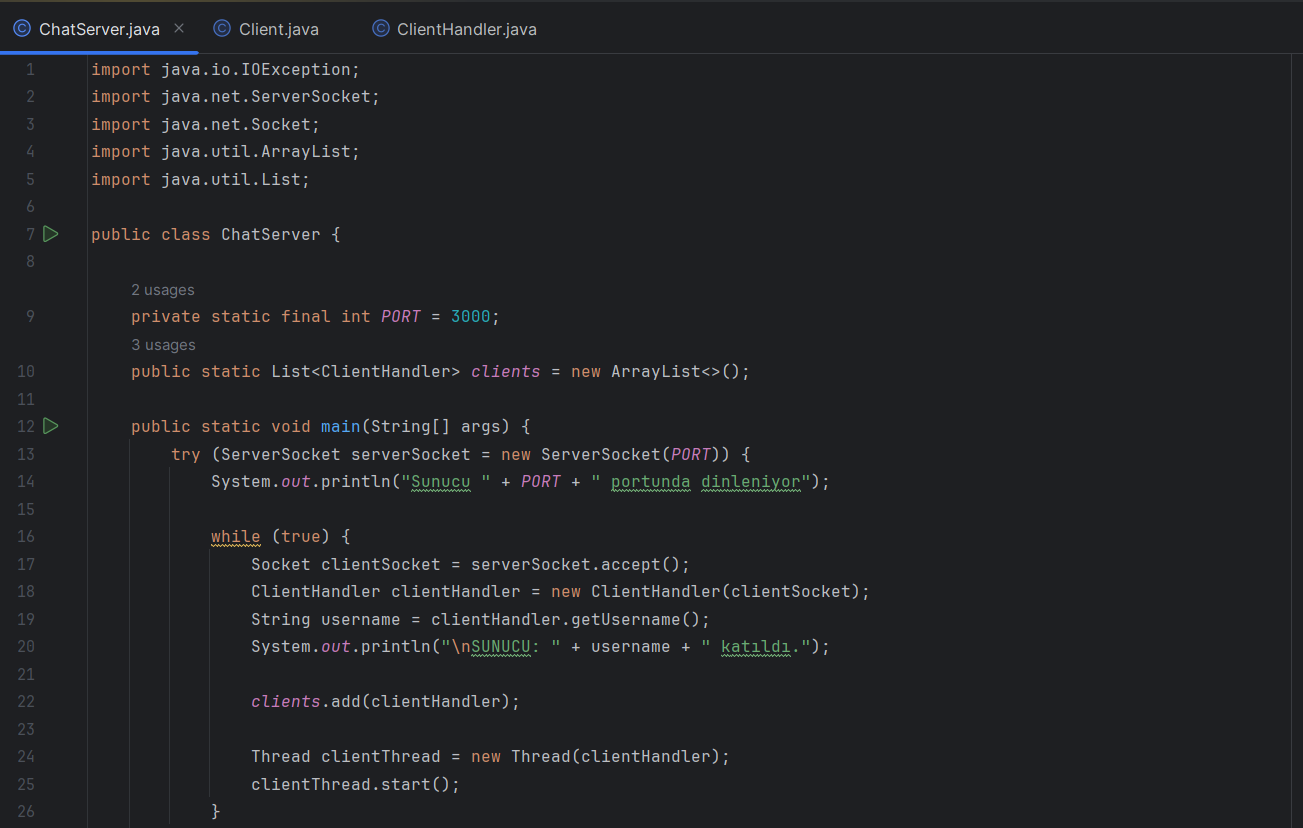
Cemre Nur Cander - 202013709030

Melike Seray Özdemir - 202013709021

Yusuf Emre Çoban - 202013709005

**1. Proje İçeriği ve Çalışması:**

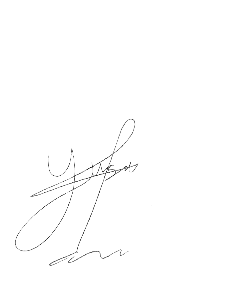
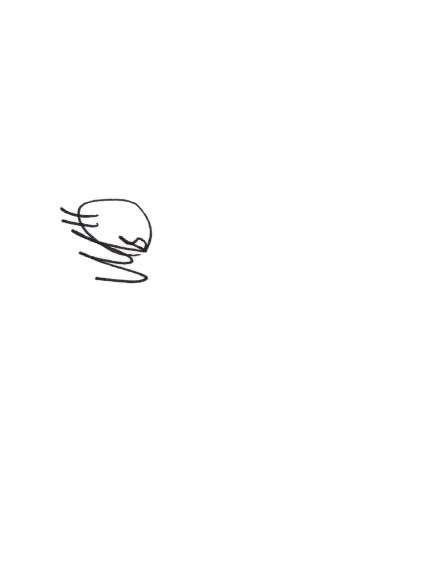
**1.1. ChatServer Sınıfı**

****

**1.1.1. Sunucu Başlatma:**

main metodu, sunucuyu başlatır. ServerSocket kullanarak belirtilen portu dinlemeye başlar.

ServerSocket'in accept metodunu kullanarak gelen istemcileri kabul eder ve onlar için bir ClientHandler oluşturur.



**1.1.2. ClientHandler:**

ClientHandler sınıfı, her bir istemci bağlantısı için oluşturulur.

Yeni bir istemci bağlandığında, ClientHandler ona bir kullanıcı adı almasını sağlar.

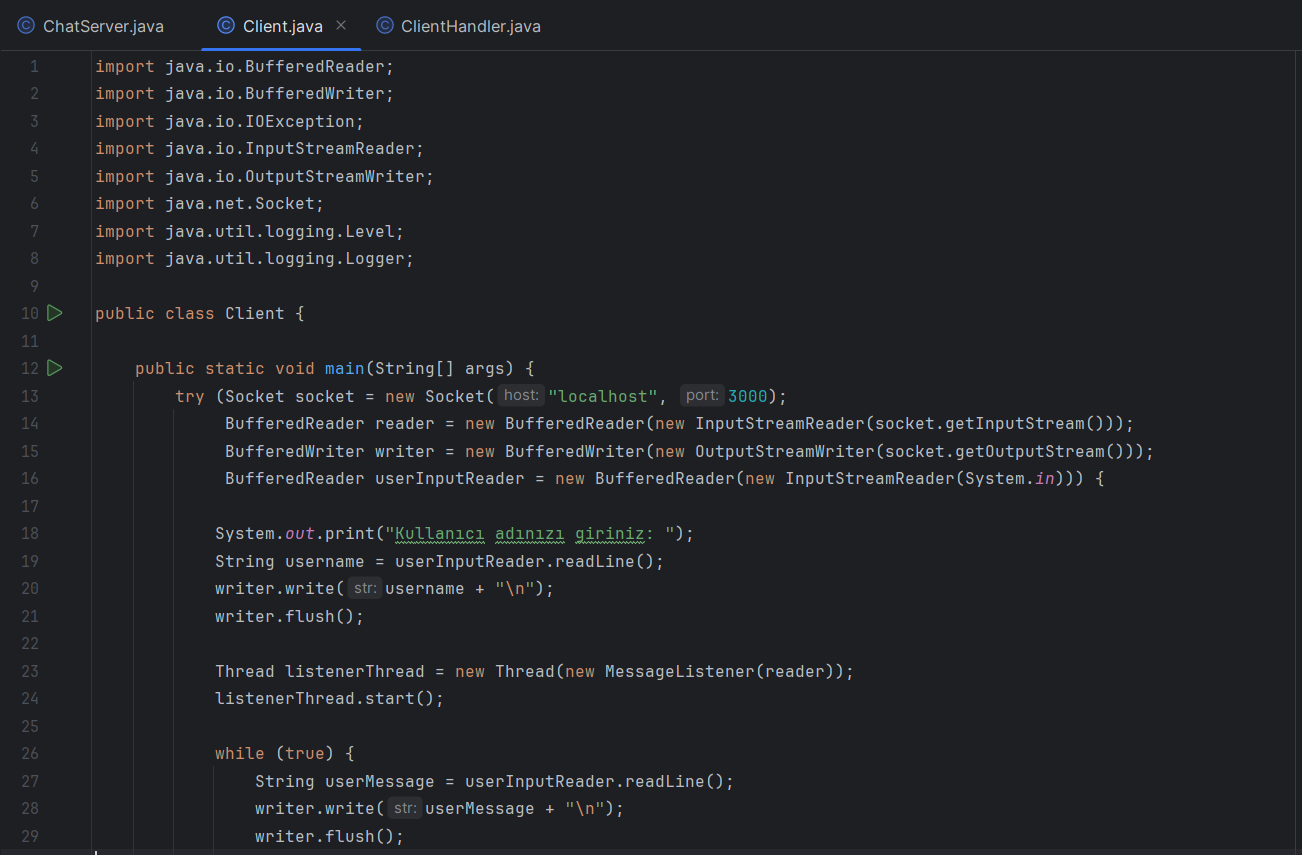
ClientHandler içindeki bir iş parçacığı, istemciyle iletişim kurar. İstemciden gelen mesajları okur ve diğer istemcilere iletmek için broadcastMessage metodunu kullanır.

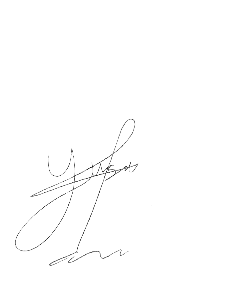
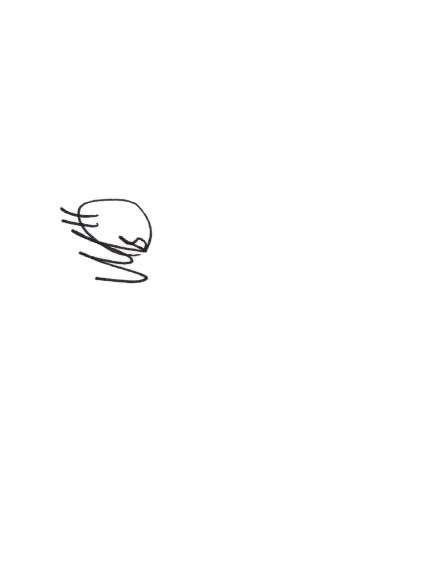
**1.1.3. broadcastMessage Metodu:**

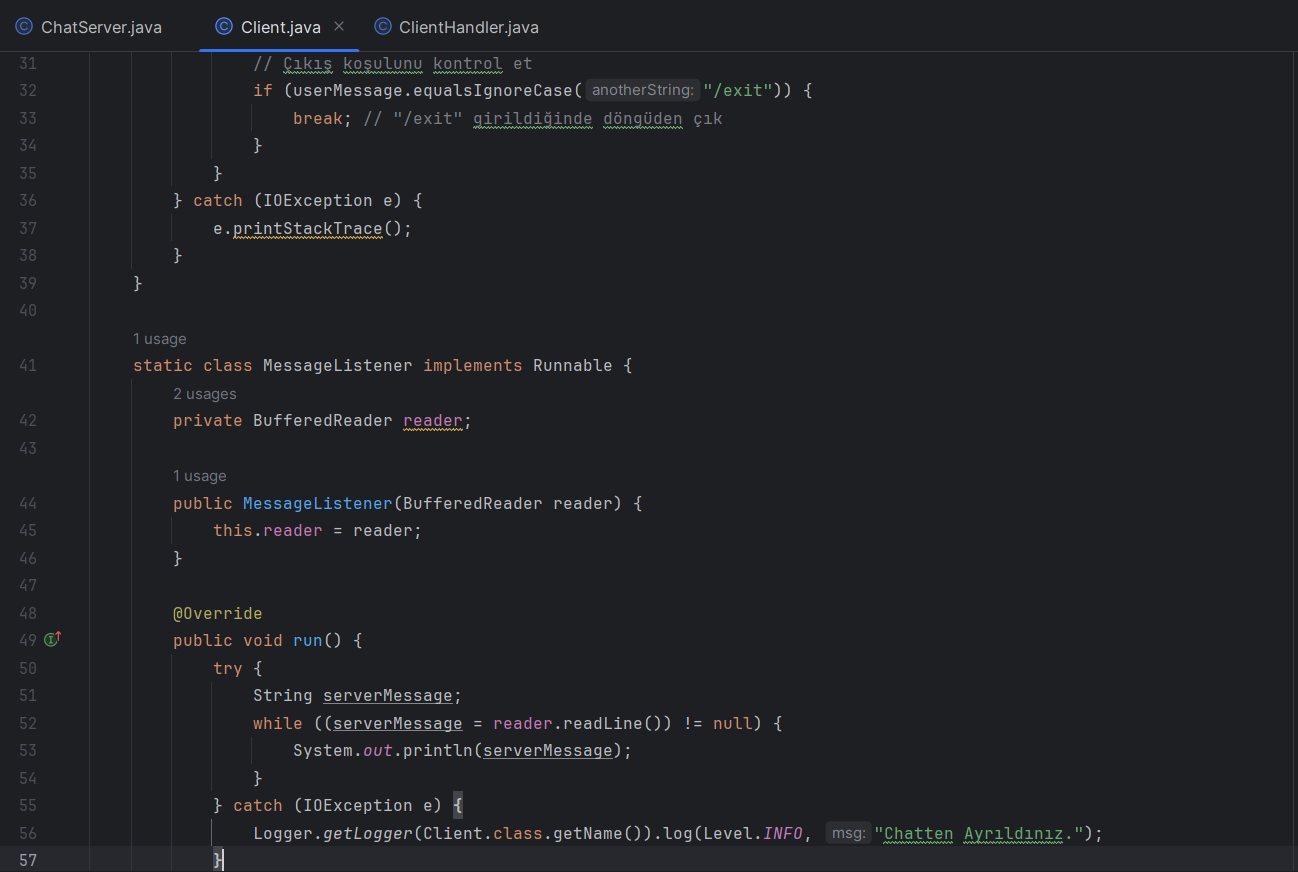
Bu metod, belirtilen mesajı tüm istemcilere gönderir. Ancak, gönderen istemci hariç tutulur (if (!client.equals(sender)) kontrolü ile).

Tüm bağlı istemciler (clients listesindeki ClientHandler nesneleri) üzerinde döngü kullanarak mesaj gönderme işlemini gerçekleştirir.

**1.2. Client Sınıfı**

****



****

**1.2.1. Bağlantı Oluşturma:**

main metodu, localhost üzerinde belirtilen port numarasıyla sunucuya bir soket bağlantısı oluşturur (Socket socket = new Socket("localhost", 3000)).

**1.2.2. Kullanıcı Adı Alma:**

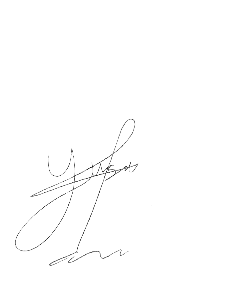
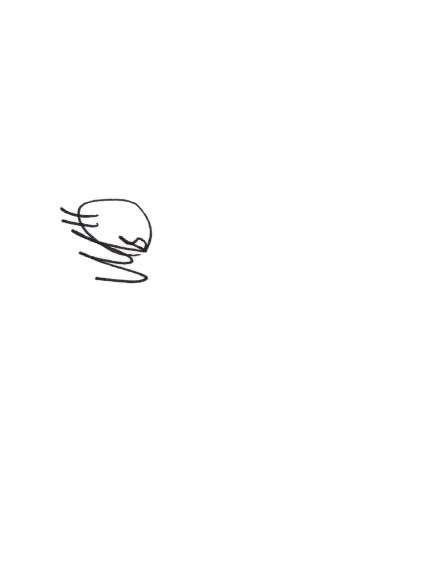
Kullanıcıdan bir kullanıcı adı girmesi istenir (System.out.print("Kullanıcı adınızı giriniz: ");).

Girilen kullanıcı adı, sunucuya gönderilir (writer.write(username + "\n"); writer.flush();).

**1.2.3. Mesaj Dinleyici:**

Sunucudan gelen mesajları dinlemek için ayrı bir iş parçacığı (MessageListener) oluşturulur ve başlatılır (Thread listenerThread = new Thread(new MessageListener(reader)); listenerThread.start();).

Bu iş parçacığı, sunucudan gelen mesajları sürekli olarak dinler (while ((serverMessage = reader.readLine()) != null)).



**1.2.4. Kullanıcı Girişi ve Mesaj Gönderme Döngüsü:**

Kullanıcıdan sürekli olarak mesajlar alınır (String userMessage = userInputReader.readLine();).

Girilen mesaj, sunucuya gönderilir (writer.write(userMessage + "\n"); writer.flush();).

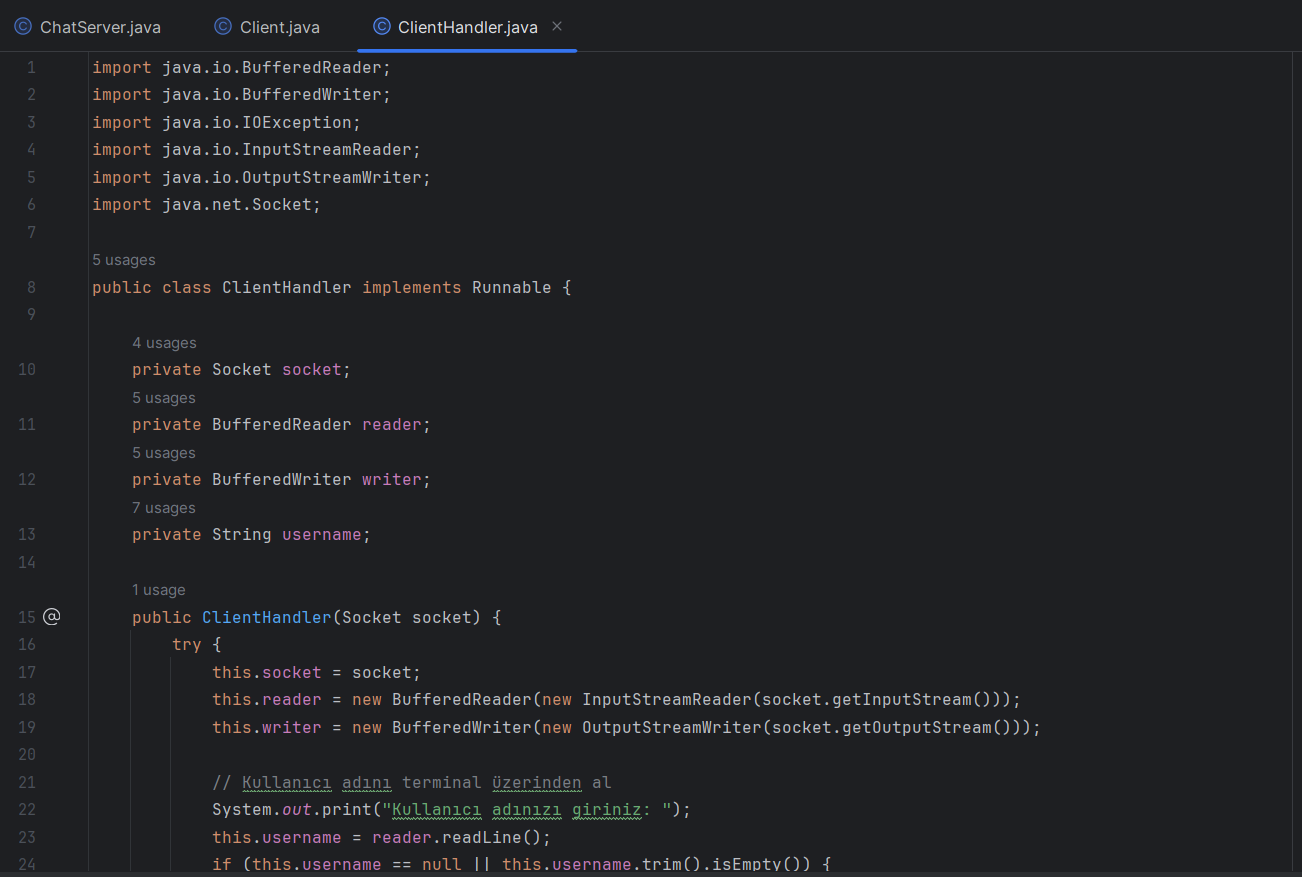
Kullanıcı "/exit" komutunu girdiğinde, döngüden çıkılır ve istemci programı kapatılır.

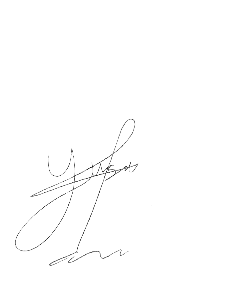
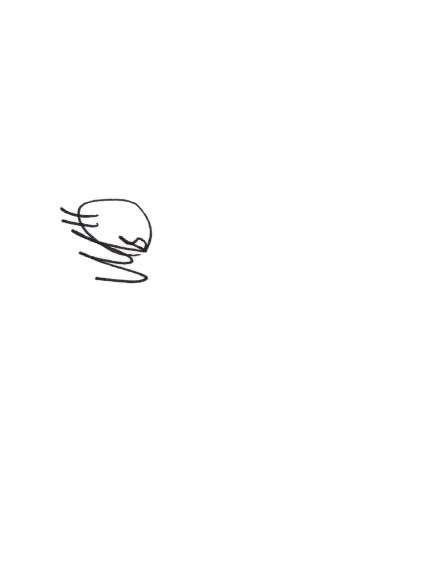
**1.2.5. Mesaj Dinleyici (MessageListener) Sınıfı:**

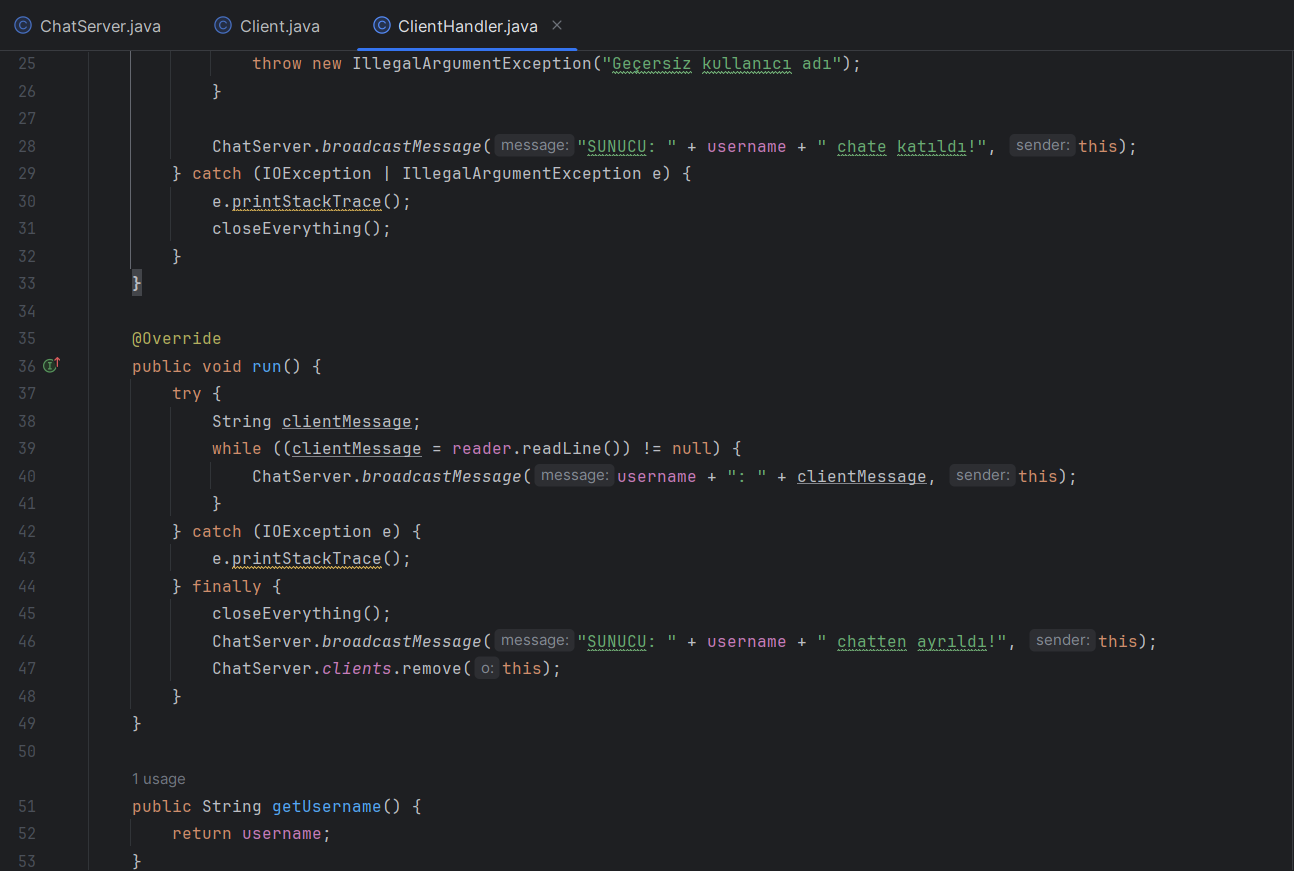
Bu sınıf, sunucudan gelen mesajları dinlemek için kullanılır.

BufferedReader üzerinden sunucudan gelen mesajları okur (reader.readLine()).

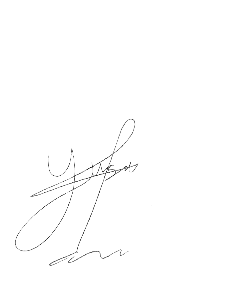
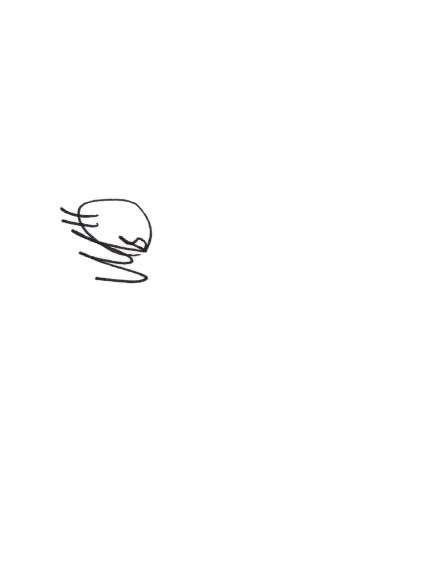
**1.3. ClientHandler Sınıfı**

****









**1.3.1. ClientHandler Sınıfı:**

Runnable arayüzünü uygulayarak bir iş parçacığı olarak çalışabilir.

Socket, BufferedReader ve BufferedWriter gibi giriş/çıkış nesnelerini kullanarak istemci bağlantısını yönetir.

**1.3.2. Kurucu Metot:**

Yeni bir ClientHandler oluşturulduğunda, istemciye ait soket ve giriş/çıkış akışlarını başlatır.

Kullanıcı adını istemciden alır (this.username = reader.readLine();). Eğer kullanıcı adı geçersizse veya boşsa, bir istisna fırlatır ve istemci bağlantısını kapatır.

**1.3.3. run Metodu:**

İstemciden gelen mesajları sürekli olarak dinler ve bu mesajları ChatServer üzerinden diğer istemcilere iletmek için broadcastMessage metodunu kullanır.

Bir istemci bağlantısı kesildiğinde veya istemci tarafından "/exit" komutu girildiğinde döngüden çıkar. Bu durumda istemci bağlantısını kapatır ve sunucuya ayrıldığını bildiren bir mesaj iletmek üzere ChatServer'a haber verir.

**1.3.4. Diğer Yardımcı Metotlar:**

**1.3.4.1.** getUsername: İstemcinin kullanıcı adını döndürür.

**1.3.4.2.** sendMessage: İstemciye mesaj göndermek için kullanılır.

**1.3.4.3.** closeEverything: Giriş/çıkış akışlarını ve soketi kapatır.

