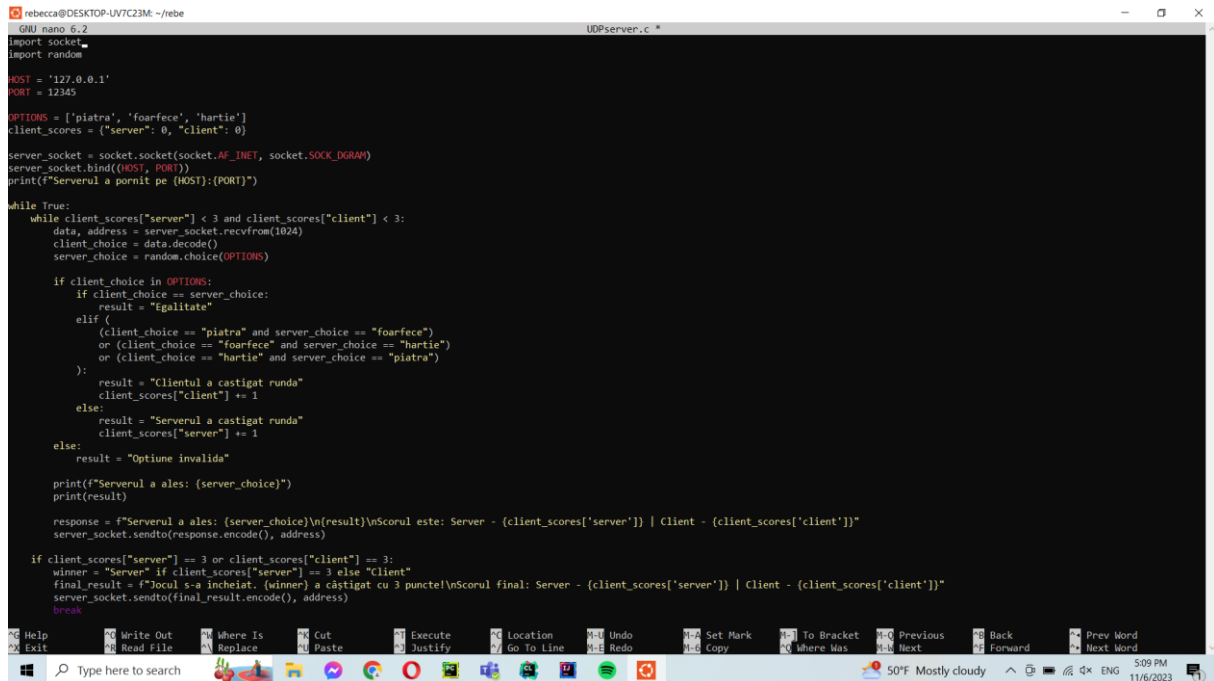


# PIATRA, FOARFECE, HARTIE

## UDP



```
rebecca@DESKTOP-UV7C23M: ~/rebe
GNU nano 6.2
import socket
import random

HOST = '127.0.0.1'
PORT = 12345

OPTIONS = ['piatra', 'foarfece', 'hartie']
client_scores = {"server": 0, "client": 0}

server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind((HOST, PORT))
print(f"Serverul a pornit pe (HOST):(PORT)")

while True:
    while client_scores["server"] < 3 and client_scores["client"] < 3:
        data, address = server_socket.recvfrom(1024)
        client_choice = data.decode()
        server_choice = random.choice(OPTIONS)

        if client_choice in OPTIONS:
            if client_choice == server_choice:
                result = "Egalitate"
            elif (
                (client_choice == "piatra" and server_choice == "foarfece")
                or (client_choice == "foarfece" and server_choice == "hartie")
                or (client_choice == "hartie" and server_choice == "piatra")
            ):
                result = "Clientul a castigat runda"
                client_scores["client"] += 1
            else:
                result = "Serverul a castigat runda"
                client_scores["server"] += 1
        else:
            result = "Optiune invalida"

        print(f"Serverul a ales: {server_choice}")
        print(result)

        response = f"Serverul a ales: {server_choice}\n(result)\nScorul este: Server - {client_scores['server']} | Client - {client_scores['client']}"
        server_socket.sendto(response.encode(), address)

    if client_scores["server"] == 3 or client_scores["client"] == 3:
        winner = "Server" if client_scores["server"] == 3 else "Client"
        final_result = f"Jocul s-a incheiat. (winner) a castigat cu 3 puncte!\nScorul final: Server - {client_scores['server']} | Client - {client_scores['client']}"
        server_socket.sendto(final_result.encode(), address)
        break
```

Acesta este un server UDP scris in limbajul Python pentru un joc de "PIATRA, FOARFECE, HARTIE".

Serverul asteapta conexiuni pe adresa IP si portul specificate si permite unui client sa joace impotriva sa.

Serverul incepe prin importarea bibliotecilor necesare, cum ar fi socket pentru gestionarea comunicarii prin protocolul UDP si random pentru a genera aleatoriu optiunile serverului: "piatra", "foarfece" sau "hartie".

Adresa IP si portul de care serverul asculta sunt specificate in variabilele HOST si PORT. In acest exemplu, serverul este configurat sa asculte de adresa locala (localhost) la portul 12345.

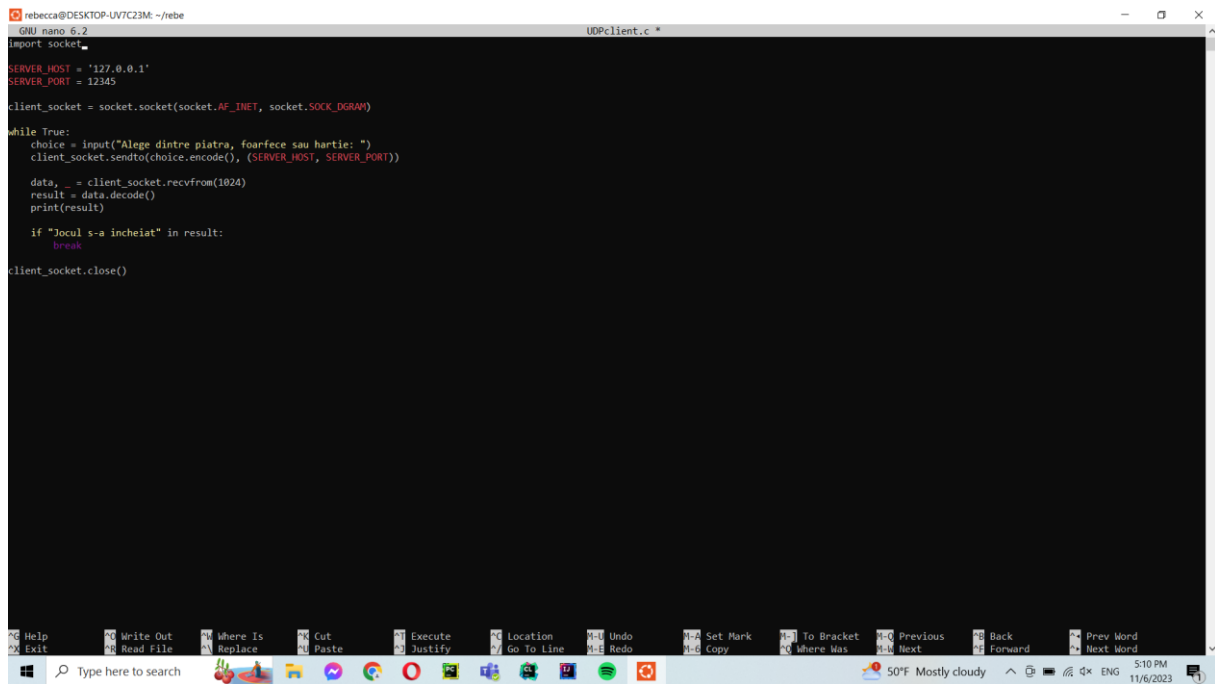
Optiunile disponibile in joc sunt definite in lista OPTIONS, si serverul pastreaza intr-un dictionar pentru a urmari scorul atat al sau, cat si al clientului.

Serverul creeaza un socket UDP si se leaga la adresa si portul specificate.

Apoi, intra intr-o bucla infinita in care asteapta alegerile clientului.

Cand clientul trimite o alegere, serverul compara alegerea clientului cu a sa, determina rezultatul rundeii si actualizeaza scorul. Apoi, trimite un raspuns catre client cu alegerea serverului, rezultatul rundeii si scorul actual.

Jocul continua pana cand unul dintre jucatori acumuleaza 3 puncte, cand serverul trimite un mesaj final care declara incheierea jocului si afiseaza scorul final.

A screenshot of a terminal window titled 'rebecca@DESKTOP-UV7C23M: ~/rebe'. The terminal shows a nano editor editing a file named 'UDPclient.c'. The code is a Python script for a UDP client. It imports the 'socket' module, sets 'SERVER\_HOST' to '127.0.0.1' and 'SERVER\_PORT' to 12345. It creates a 'client\_socket' using 'socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)'. A 'while True:' loop prompts the user to 'Alege dintre piatra, foarfece sau hartie: ', sends the choice to the server, receives a response, and prints it. A break condition is included for 'Jocul s-a incheiat'. The loop ends with 'client\_socket.close()'. The terminal window has a standard Linux desktop environment with a taskbar at the bottom showing various application icons and system status information like '50°F Mostly cloudy' and '5:10 PM 11/6/2023'.

Acesta este un client UDP scris in limbajul Python pentru un joc de "PIATRA, FOARFECI, HARTIE".

Initializarea clientului implica specificarea adresei IP si portul la care se afla serverul. Aceste informatii sunt stocate in variabilele SERVER\_HOST si SERVER\_PORT.

Clientul creeaza un socket UDP cu ajutorul functiei socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM), astfel stabilind modalitatea de comunicare cu serverul.

Apoi, clientul intra intr-o bucla infinita (while True) care asteapta introducerea unei alegeri din partea utilizatorului. Aceasta alegere poate fi "piatra," "foarfece" sau "hartie," si este primita de la utilizator prin intermediul functiei input.

Dupa ce utilizatorul alege, clientul trimite aceasta alegere catre server utilizand client\_socket.sendto(choice.encode(), (SERVER\_HOST, SERVER\_PORT)), unde choice reprezinta alegerea utilizatorului.

Clientul asteapta sa primeasca un raspuns de la server prin utilizarea client\_socket.recvfrom(1024) și decodeaza acest raspuns pentru a-l afisa utilizatorului. Raspunsul indica rezultatul runde (castig, pierdere sau egalitate) si scorul actual.

Jocul continua pana cand serverul anunta ca s-a incheiat, moment in care bucla se incheie si clientul inchide socket-ul UDP cu client\_socket.close().

Aici este un exemplu de cum functioneaza programul:

```
rebecca@DESKTOP-UV7C23M: ~/rebe
rebecca@DESKTOP-UV7C23M:~$ cd rebe
rebecca@DESKTOP-UV7C23M:~/rebe$ python3 UDPServer.c
Serverul a pornit pe 127.0.0.1:12345
Serverul a ales: piatra
Egalitate
Serverul a ales: hartie
Clientul a castigat runda
Serverul a ales: piatra
Clientul a castigat runda
Serverul a ales: piatra
Clientul a castigat runda
rebecca@DESKTOP-UV7C23M:~/rebe$ python3 UDPServer.c
Serverul a pornit pe 127.0.0.1:12345
Serverul a ales: foarfece
Serverul a castigat runda
Serverul a ales: foarfece
Serverul a castigat runda
Clientul a castigat runda
Serverul a ales: piatra
Serverul a castigat runda
rebecca@DESKTOP-UV7C23M:~/rebe$

rebecca@DESKTOP-UV7C23M:~/rebe$ cd rebe
rebecca@DESKTOP-UV7C23M:~/rebe$ python3 UDPClient.c
Alege dintre piatra, foarfece sau hartie: piatra
Serverul a ales: piatra
Egalitate
Scorul este: Server - 0 | Client - 0
Alege dintre piatra, foarfece sau hartie: foarfece
Serverul a ales: hartie
Clientul a castigat runda
Scorul este: Server - 0 | Client - 1
Alege dintre piatra, foarfece sau hartie: hartie
Serverul a ales: piatra
Clientul a castigat runda
Scorul este: Server - 0 | Client - 2
Alege dintre piatra, foarfece sau hartie: hartie
Serverul a ales: piatra
Clientul a castigat runda
Scorul este: Server - 0 | Client - 3
Alege dintre piatra, foarfece sau hartie: piatra
Jocul s-a incheiat. Client a castigat cu 3 puncte!
Scorul final: Server - 0 | Client - 3
rebecca@DESKTOP-UV7C23M:~/rebe$ python3 UDPClient.c
Alege dintre piatra, foarfece sau hartie: hartie
Serverul a ales: foarfece
Serverul a castigat runda
Scorul este: Server - 1 | Client - 0
Alege dintre piatra, foarfece sau hartie: hartie
Serverul a ales: foarfece
Serverul a castigat runda
Scorul este: Server - 2 | Client - 0
Alege dintre piatra, foarfece sau hartie: piatra
Serverul a ales: piatra
Clientul a castigat runda
Scorul este: Server - 2 | Client - 1
Alege dintre piatra, foarfece sau hartie: foarfece
Serverul a ales: piatra
Scorul este: Server - 3 | Client - 1
Alege dintre piatra, foarfece sau hartie: foarfece
Jocul s-a incheiat. Server a castigat cu 3 puncte!
Scorul final: Server - 3 | Client - 1
rebecca@DESKTOP-UV7C23M:~/rebe$
```

## TCP

```
Select rebecca@DESKTOP-UV7C23M: ~/rebe
GNU nano 6.2
TCPServer.java
import java.io.*;
import java.net.*;

public class TCPServer {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(12345);
            System.out.println("Serverul este pornit. Asteptand jucatori...");

            Socket player1Socket = serverSocket.accept();
            System.out.println("Jucatorul 1 s-a conectat.");

            Socket player2Socket = serverSocket.accept();
            System.out.println("Jucatorul 2 s-a conectat.");

            BufferedReader player1In = new BufferedReader(new InputStreamReader(player1Socket.getInputStream()));
            PrintWriter player1Out = new PrintWriter(player1Socket.getOutputStream(), true);
            BufferedReader player2In = new BufferedReader(new InputStreamReader(player2Socket.getInputStream()));
            PrintWriter player2Out = new PrintWriter(player2Socket.getOutputStream(), true);

            int player1Score = 0;
            int player2Score = 0;

            while (true) {
                player1Out.println("Bine ai venit, esti jucatorul 1.");
                player2Out.println("Bine ai venit, esti jucatorul 2.");

                while (player1Score < 3 && player2Score < 3) {
                    player1Out.println("Score: Jucator 1: " + player1Score + " | Jucator 2: " + player2Score);
                    player2Out.println("Score: Jucator 1: " + player1Score + " | Jucator 2: " + player2Score);

                    player1Out.println("Alege: piatra, hartie sau foarfece.");
                    player2Out.println("Alege: piatra, hartie sau foarfece.");

                    String player1Choice = player1In.readLine();
                    String player2Choice = player2In.readLine();

                    if (player1Choice.equals(player2Choice)) {
                        player1Out.println("Egalitate!");
                        player2Out.println("Egalitate!");
                    } else {
                        (player1Choice.equals("piatra") && player2Choice.equals("foarfece")) ||
                        (player1Choice.equals("foarfece") && player2Choice.equals("hartie")) ||
                        (player1Choice.equals("hartie") && player2Choice.equals("piatra"))
                    } {

```

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled 'rebecca@DESKTOP-UV7C23M: ~/rebe' with a dark background. It displays Java code for a TCP server. The code includes logic for two players, choice validation, score tracking, and game termination conditions. A second window, titled 'TCPserver.java \*', shows the same code in a nano editor. The Windows taskbar at the bottom includes the Start button, a search bar, and various application icons. The system tray shows the date and time as 5:14 PM on 11/6/2023.

```
player1Out.println("Score: Jucator 1: " + player1Score + " | Jucator 2: " + player2Score);
player2Out.println("Score: Jucator 1: " + player1Score + " | Jucator 2: " + player2Score);

player1Out.println("Alege: piatra, hartie sau foarfece.");
player2Out.println("Alege: piatra, hartie sau foarfece.");

String player1Choice = player1In.readLine();
String player2Choice = player2In.readLine();

if (player1Choice.equals(player2Choice)) {
    player1Out.println("egalitate");
    player2Out.println("egalitate");
} else if (
    (player1Choice.equals("piatra") && player2Choice.equals("foarfece")) ||
    (player1Choice.equals("foarfece") && player2Choice.equals("hartie")) ||
    (player1Choice.equals("hartie") && player2Choice.equals("piatra"))
) {
    player1Out.println("Jucatorul 1 a castigat runda!");
    player2Out.println("Jucatorul 1 a castigat runda!");
    player1Score++;
} else {
    player1Out.println("Jucatorul 2 a castigat runda!");
    player2Out.println("Jucatorul 2 a castigat runda!");
    player2Score++;
}

if (player1Score == 3) {
    player1Out.println("Jucatorul 1 a castigat jocul cu un scor de 3 puncte!");
    player2Out.println("Jucatorul 1 a castigat jocul cu un scor de 3 puncte!");
} else {
    player1Out.println("Jucatorul 2 a castigat jocul cu un scor de 3 puncte!");
    player2Out.println("Jucatorul 2 a castigat jocul cu un scor de 3 puncte!");
}

player1Socket.close();
player2Socket.close();
break;
} catch (IOException e) {
    e.printStackTrace();
}
}
```

Acesta este un server TCP scris in limbajul Java pentru un joc de "PIATRA, FOARFECE, HARTIE".

In prima etapa, serverul creeaza un obiect `ServerSocket` care asculta pe portul 12345. Aceasta linie este responsabila pentru pornirea serverului si asteptarea conexiunilor de la clienti. Cand se deschide portul specificat, serverul afiseaza un mesaj in consola pentru a indica ca este pornit si asteapta jucatori.

Dupa ce serverul primeste o conexiune de la un jucator, creeaza doua socketuri separate pentru a comunica cu fiecare jucator. Aceste socketuri sunt `player1Socket` si `player2Socket`, fiecare corespunzand unui jucator.

Pentru fiecare jucator, serverul creeaza un flux de intrare (`BufferedReader`) pentru a citi datele trimise de jucator si un flux de iesire (`PrintWriter`) pentru a trimite raspunsuri catre jucator.

Jocul incepe cu initializarea scorurilor jucatorilor (`player1Score` si `player2Score`) la zero.

Intr-o bucla infinita (`while true`), serverul ii intampina pe jucatori si le afiseaza scorurile curente.

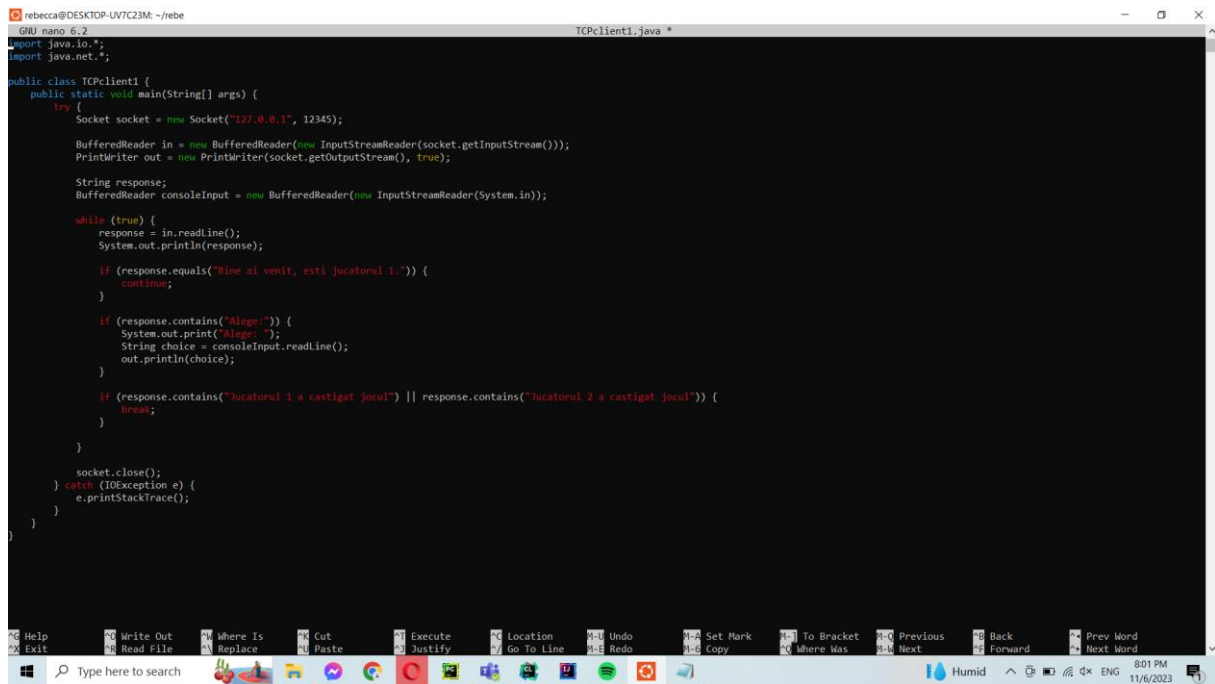
Apoi, fiecare jucator face o alegere dintre "piatra," "hartie" si "foarfece." Serverul primeste alegerile jucatorilor si le compara pentru a determina rezultatul fiecarei runde (egalitate, victorie jucator 1 sau victorie jucator 2).

Serverul afiseaza scorurile actualizate pentru amandoi jucatorii dupa fiecare runda.

Jocul continua pana cand unul dintre jucatori acumuleaza 3 puncte, cand serverul anunta incheierea jocului, castigatorul si scorul final al ambilor jucatori.

Dupa incheierea jocului, serverul inchide socketurile pentru ambii jucatori si se termina bucla infinita.

Serverul are si gestionarea erorilor pentru tratarea situatiilor neasteptate. Daca apare o exceptie, cum ar fi o eroare de citire sau o problema de conectare la socketuri, aceasta este afisata in consola, iar serverul continua sa ruleze.



```
rebecca@DESKTOP-UV7C23M: ~/rebe
GNU nano 6.2
import java.io.*;
import java.net.*;
import java.util.*;

public class TCPClient1 {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("127.0.0.1", 12345);

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            String response;
            BufferedReader consoleInput = new BufferedReader(new InputStreamReader(System.in));

            while (true) {
                response = in.readLine();
                System.out.println(response);

                if (response.equals("Bine ai venit, esti jucatorul 1.")) {
                    continue;
                }

                if (response.contains("Alege:")) {
                    System.out.print("Alege: ");
                    String choice = consoleInput.readLine();
                    out.println(choice);
                }

                if (response.contains("Jucatorul 1 a castigat jocul") || response.contains("Jucatorul 2 a castigat jocul")) {
                    break;
                }
            }

            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Acesta este un client TCP scris in limbajul Java pentru un joc de "PIATRA, FOARFECE, HARTIE".

La inceput, clientul creeaza un socket si se conecteaza la adresa IP "127.0.0.1" (localhost) si portul 12345. Aceasta linie stabileste conexiunea cu serverul care ruleaza, pe portul specificat.

Dupa stabilirea conexiunii, clientul creeaza doua fluxuri de date. Unul, numit in, este un BufferedReader si este utilizat pentru a citi mesajele primite de la server. Celalalt, numit out, este un PrintWriter si este folosit pentru a trimite raspunsuri la server.

In interiorul unei bucle infinite (while true), clientul primeste mesaje de la server si le afiseaza in consola folosind System.out.println(response).

Daca primul mesaj primit este "Bine ai venit, esti jucatorul 1," clientul trece peste acest mesaj, deoarece este doar un mesaj de bun venit si nu necesita o actiune suplimentara.

Cand clientul primeste un mesaj care contine "Alege:", acesta stie ca trebuie sa faca o alegere si solicita utilizatorului sa introduca "piatra," "hartie" sau "foarfece" de la tastatura. Alegerea utilizatorului este citita cu ajutorul consoleInput.readLine() si apoi trimisa la server utilizand out.println(choice).

Clientul asteapta mesaje noi de la server, iar bucla continua pana cand primeste un mesaj care indica incheierea jocului, cum ar fi "Jucatorul 1 a castigat jocul" sau "Jucatorul 2 a castigat jocul." La acest punct, bucla se incheie.

Dupa incheierea jocului, clientul inchide socketul pentru a inchide conexiunea catre server si pentru a elibera resursele folosite.

Clientul are si gestionarea erorilor pentru a trata eventualele probleme de comunicare. In cazul in care apare o exceptie, precum o eroare de retea sau o eroare la nivelul socketului, aceasta este afisata in consola.

Acelasi lucru face si TCPclient2:

The screenshot displays a Windows desktop environment. The primary application is a Java IDE, showing a file named `TCPClient2.java`. The code is as follows:

```

import java.io.*;
import java.net.*;

public class TCPClient2 {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("127.0.0.1", 12345);

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            String response;
            BufferedReader consoleInput = new BufferedReader(new InputStreamReader(System.in));

            while (true) {
                response = in.readLine();
                System.out.println(response);

                if (response.equals("Bine ai venit, esti jucatorul 2. ")) {
                    continue;
                }

                if (response.contains("Alege: ")) {
                    System.out.print("Alege: ");
                    String choice = consoleInput.readLine();
                    out.println(choice);
                }

                if (response.contains("Jucatorul 2 a castigat jocul") || response.contains("Jucatorul 1 a castigat jocul")) {
                    break;
                }
            }

            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

The IDE's status bar at the bottom indicates the current line is 40, with 40 lines in total. The system tray shows the temperature as 48°F, weather as Cloudy, and the time as 8:05 PM on 11/6/2021.

Aici este un exemplu de cum functioneaza programul:

```
rebecca@DESKTOP-UV7C23M: ~/rebe
rebecca@DESKTOP-UV7C23M:~$ cd rebe
rebecca@DESKTOP-UV7C23M:~/rebe$ javac TCPserver.java
rebecca@DESKTOP-UV7C23M:~/rebe$ java TCPserver
Serverul este pornit. Asteptand jucatori...
Jucatorul 1 s-a conectat.
rebecca@DESKTOP-UV7C23M:~/rebe$

rebecca@DESKTOP-UV7C23M:~$ cd rebe
rebecca@DESKTOP-UV7C23M:~/rebe$ javac TCPclient1.java
rebecca@DESKTOP-UV7C23M:~/rebe$ java TCPclient1
Bine ai venit, esti jucatorul 1.
Scor: Jucator 1: 0 | Jucator 2: 0
Alege: piatra, hartie sau foarfece.
Alege: piatra
Egalitate!
Scor: Jucator 1: 0 | Jucator 2: 0
Alege: piatra, hartie sau foarfece.
Alege: hartie
Jucatorul 1 a castigat runda!
Scor: Jucator 1: 1 | Jucator 2: 0
Alege: piatra, hartie sau foarfece.
Alege: foarfece
Jucatorul 1 a castigat runda!
Scor: Jucator 1: 2 | Jucator 2: 0
Alege: piatra, hartie sau foarfece.
Alege: foarfece
Jucatorul 2 a castigat runda!
Scor: Jucator 1: 2 | Jucator 2: 1
Alege: piatra, hartie sau foarfece.
Alege: hartie
Jucatorul 2 a castigat runda!
Scor: Jucator 1: 2 | Jucator 2: 2
Alege: piatra, hartie sau foarfece.
Alege: hartie
Egalitate!
Scor: Jucator 1: 2 | Jucator 2: 2
Alege: piatra, hartie sau foarfece.
Alege: piatra
Jucatorul 2 a castigat runda!
Jucatorul 2 a castigat jocul cu un scor de 3 puncte!
rebecca@DESKTOP-UV7C23M:~/rebe$

rebecca@DESKTOP-UV7C23M:~$ cd rebe
rebecca@DESKTOP-UV7C23M:~/rebe$ javac TCPclient2.java
rebecca@DESKTOP-UV7C23M:~/rebe$ java TCPclient2
Bine ai venit, esti jucatorul 2.
Scor: Jucator 1: 0 | Jucator 2: 0
Alege: piatra, hartie sau foarfece.
Alege: piatra
Egalitate!
Scor: Jucator 1: 0 | Jucator 2: 0
Alege: piatra, hartie sau foarfece.
Alege: piatra
Jucatorul 1 a castigat runda!
Scor: Jucator 1: 1 | Jucator 2: 0
Alege: piatra, hartie sau foarfece.
Alege: hartie
Jucatorul 1 a castigat runda!
Scor: Jucator 1: 2 | Jucator 2: 0
Alege: piatra, hartie sau foarfece.
Alege: piatra
Jucatorul 2 a castigat runda!
Scor: Jucator 1: 2 | Jucator 2: 1
Alege: piatra, hartie sau foarfece.
Alege: foarfece
Jucatorul 2 a castigat runda!
Scor: Jucator 1: 2 | Jucator 2: 2
Alege: piatra, hartie sau foarfece.
Alege: hartie
Egalitate!
Scor: Jucator 1: 2 | Jucator 2: 2
Alege: piatra, hartie sau foarfece.
Alege: hartie
Jucatorul 2 a castigat runda!
Jucatorul 2 a castigat jocul cu un scor de 3 puncte!
rebecca@DESKTOP-UV7C23M:~/rebe$
```

Cod:

UDPserver.c:

```
import socket
```

```
import random
```

```
HOST = '127.0.0.1'
```

```
PORT = 12345
```

```
OPTIONS = ['piatra', 'foarfece', 'hartie']
```

```
client_scores = {"server": 0, "client": 0}
```

```
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```
server_socket.bind((HOST, PORT))
```

```
print(f"Serverul a pornit pe {HOST}:{PORT}")
```

```
while True:
```

```
    while client_scores["server"] < 3 and client_scores["client"] < 3:
```

```
        data, address = server_socket.recvfrom(1024)
```

```
        client_choice = data.decode()
```

```
        server_choice = random.choice(OPTIONS)
```

```
    if client_choice in OPTIONS:
```

```
        if client_choice == server_choice:
```

```
            result = "Egalitate"
```

```
        elif (
```

```
            (client_choice == "piatra" and server_choice == "foarfece")
```

```
            or (client_choice == "foarfece" and server_choice == "hartie")
```

```
            or (client_choice == "hartie" and server_choice == "piatra")
```

```
        ):

```

```
            result = "Clientul a castigat runda"
```

```
            client_scores["client"] += 1
```

```

        else:
            result = "Serverul a castigat runda"
            client_scores["server"] += 1
        else:
            result = "Optiune invalida"

    print(f"Serverul a ales: {server_choice}")
    print(result)

    response = f"Serverul a ales: {server_choice}\n{result}\nScorul este: Server - {client_scores['server']} | Client - {client_scores['client']}"
    server_socket.sendto(response.encode(), address)

    if client_scores["server"] == 3 or client_scores["client"] == 3:
        winner = "Server" if client_scores["server"] == 3 else "Client"
        final_result = f"Jocul s-a incheiat. {winner} a câștigat cu 3 puncte!\nScorul final: Server - {client_scores['server']} | Client - {client_scores['client']}"
        server_socket.sendto(final_result.encode(), address)
        break

```

UDPclient.c:

```

import socket

SERVER_HOST = '127.0.0.1'
SERVER_PORT = 12345

client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

while True:
    choice = input("Alege dintre piatra, foarfece sau hartie: ")
    client_socket.sendto(choice.encode(), (SERVER_HOST, SERVER_PORT))

```



```
data, _ = client_socket.recvfrom(1024)
```

```
result = data.decode()
```

```
print(result)
```

```
if "Jocul s-a incheiat" in result:
```

```
    break
```

```
client_socket.close()
```

TCPserver.java:

```
import java.io.*;
```

```
import java.net.*;
```

```
public class TCPserver {
```

```
    public static void main(String[] args) {
```

```
        try {
```

```
            ServerSocket serverSocket = new ServerSocket(12345);
```

```
            System.out.println("Serverul este pornit. Asteptand jucatori...");
```

```
            Socket player1Socket = serverSocket.accept();
```

```
            System.out.println("Jucatorul 1 s-a conectat.");
```

```
            Socket player2Socket = serverSocket.accept();
```

```
            System.out.println("Jucatorul 2 s-a conectat.");
```

```
            BufferedReader player1In = new BufferedReader(new  
InputStreamReader(player1Socket.getInputStream()));
```

```
            PrintWriter player1Out = new PrintWriter(player1Socket.getOutputStream(), true);
```

```
BufferedReader player2In = new BufferedReader(new
InputStreamReader(player2Socket.getInputStream()));

PrintWriter player2Out = new PrintWriter(player2Socket.getOutputStream(), true);

int player1Score = 0;
int player2Score = 0;

while (true) {
    player1Out.println("Bine ai venit, esti jucatorul 1.");
    player2Out.println("Bine ai venit, esti jucatorul 2.");

    while (player1Score < 3 && player2Score < 3) {
        player1Out.println("Scor: Jucator 1: " + player1Score + " | Jucator 2: " + player2Score);
        player2Out.println("Scor: Jucator 1: " + player1Score + " | Jucator 2: " + player2Score);

        player1Out.println("Alege: piatra, hartie sau foarfece.");
        player2Out.println("Alege: piatra, hartie sau foarfece.");

        String player1Choice = player1In.readLine();
        String player2Choice = player2In.readLine();

        if (player1Choice.equals(player2Choice)) {
            player1Out.println("Egalitate!");
            player2Out.println("Egalitate!");
        } else if (
            (player1Choice.equals("piatra") && player2Choice.equals("foarfece")) ||
            (player1Choice.equals("foarfece") && player2Choice.equals("hartie")) ||
            (player1Choice.equals("hartie") && player2Choice.equals("piatra"))
        ) {
            player1Out.println("Jucatorul 1 a castigat runda!");
```

```

        player2Out.println("Jucatorul 1 a castigat runda!");
        player1Score++;
    } else {
        player1Out.println("Jucatorul 2 a castigat runda!");
        player2Out.println("Jucatorul 2 a castigat runda!");
        player2Score++;
    }
}

if (player1Score == 3) {
    player1Out.println("Jucatorul 1 a castigat jocul cu un scor de 3 puncte!");
    player2Out.println("Jucatorul 1 a castigat jocul cu un scor de 3 puncte!");
} else {
    player1Out.println("Jucatorul 2 a castigat jocul cu un scor de 3 puncte!");
    player2Out.println("Jucatorul 2 a castigat jocul cu un scor de 3 puncte!");
}

player1Socket.close();
player2Socket.close();
break;
}
} catch (IOException e) {
    e.printStackTrace();
}
}
}

```

TCPClient1.java:

```
import java.io.*;
```

```

import java.net.*;

public class TCPClient1 {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("127.0.0.1", 12345);

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            String response;
            BufferedReader consoleInput = new BufferedReader(new InputStreamReader(System.in));

            while (true) {
                response = in.readLine();
                System.out.println(response);

                if (response.equals("Bine ai venit, esti jucatorul 1.")) {
                    continue;
                }

                if (response.contains("Alege:")) {
                    System.out.print("Alege: ");
                    String choice = consoleInput.readLine();
                    out.println(choice);
                }

                if (response.contains("Jucatorul 1 a castigat jocul") || response.contains("Jucatorul 2 a
castigat jocul")) {
                    break;
                }
            }
        }
    }
}

```

```

    }

    socket.close();
} catch (IOException e) {
    e.printStackTrace();
}
}
}

```

TCPclient2.java:

```

import java.io.*;
import java.net.*;

public class TCPclient2 {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("127.0.0.1", 12345);

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            String response;

            BufferedReader consoleInput = new BufferedReader(new InputStreamReader(System.in));

            while (true) {
                response = in.readLine();
                System.out.println(response);
            }
        }
    }
}

```

```
        if (response.equals("Bine ai venit, esti jucatorul 2.")) {  
            continue;  
        }  
  
        if (response.contains("Alege:")) {  
            System.out.print("Alege: ");  
            String choice = consoleInput.readLine();  
            out.println(choice);  
        }  
  
        if (response.contains("Jucatorul 2 a castigat jocul") || response.contains("Jucatorul 1 a  
castigat jocul")) {  
            break;  
        }  
    }  
  
    socket.close();  
} catch (IOException e) {  
    e.printStackTrace();  
}  
}  
}
```