Soket Programlama ile Bilgi Yarışması

Bairam Kuliev
KARABUK UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

Soket Programlama ile Bilgi Yarışması

- Birden fazla istemci, tek bir sunucu olacak ve TCP/IP protokolü üzerinden haberleşecekler.

- Yarışmaya dahil olmak isteyen istemciler sunucuya bağlanacak (İstemci sayısını siz sınırlandırabilirsiniz)

- Sunucu, veritabanındaki sorulardan random seçerek soruları istemcilere sunacak. Bunun için veritabanında soruları ve cevapları önceden oluşturacaksınız.

- Soruya doğru cevap veren istemcilerin bağlantısı devam edecek, yanlış cevap verenlerin bağlantısı kesilecek.

- Bu şekilde son 1 kişi kalana kadar yarışma devam edecek.

Server and Client Side Programming

Server.java clasından sunucuyu çağıralım. Çalıştırdıktan sonra sunucu porta bağlanış sağlıyalım ve sunucuya accept() komutunu çağırarak bağlamayı izin verelim. Sokete gelen isteği kabul etmek için java nın içinde olan accept () fonksiyonu kullanalım. İstemci bağlandıktan sonra SımpleServer tipinden simpleServer objesi oluşturulup sunucuya gönderelim ve start fonksiyonu kullanlarak Thread in başlamasını sağlıyalım.

```
ServerSocket ss = new ServerSocket( port: 8000);
System.out.println("Server is starting!");
System.out.println("Waiting for clients...");
while(true) {
    Socket server = ss.accept();
```

Sunucu tarafından porta bağlanalım.

```
ServerSocket ss = new ServerSocket( port: 8000);
```

```
Socket server = ss.accept();
```

Çalıştırdıktan sonra (Client.java) istemci tarafınının çalıştırlmasını bekleriz. Burada sonsuz while içerisinde 4 obje oluşturulup(4 oyuncu) oyuncuların oyuna devam edip etmeyeceyini kontrol ederiz. Eğer oyucu devam ediyorsa SimpleClient.java sınfını çağırılsın ve sonrasında bu oyuncu bu sınıf tarafından sunucuyla(SimpleServer.java sınıfı) iletişime geçsin.

```
int isClient1=1;
int isClient2=1;
int isClient3=1;
int isClient4=1;
int check;
```

```
SimpleClient client1 = new SimpleClient();
client1.amountOfClient=check;
if(isClient1==1){
    System.out.println("*********First client*********");
    client1.start();
    synchronized (client1){
        client1.wait();
        if(client1.correct == 0)
            isClient1=5;
    }
    check++;
}
else{
    System.out.println("First player lost the game!The next player is continuing...");
    client1.interrupt();
}
SimpleClient client2 = new SimpleClient();
client2.amountOfClient=check;
```

Şimdi bir oyuncunun oyuna devam edip etmemesini nasıl control yapa biliriz? Her oyuncu için bir int tipinden değişken yapıo bu değişkene her hangi bir sayı atıyalım(olsun 1) eğer oyuncu soruya yanlış cevap veriyorsa(yanlış ve ya doğru cevap verip vermediği daha sonra açıklıyacam) o zaman bu değeri başka bi değere eşitle ve her çağırıldığında değer control etsin, 1 den farklı ise demektir ki bu oyuncu oyunu kaybetmiştir ve oyunu bunan sonraki oyuncu devam ettirsin. Peki bir oyuncucn diğer oyuncunun beklemesini nasıl sağlarız, sırası ile cevap vermeleri için. Bunun için java da olan synchronized fonksiyonu her iki sınıfında kullanalım. SimpleClient sınıfı tüm işlemi yaptıktan sonra notify fonksiyonu ile çağırılan sınıfa, yani Client.java ya bildirim gönderilsin. Bildirim geldiğinde hemen sonraki oyuncu oyuna devam ediyor, yani tekrardan SimpleClient sınıfına bağlanıyor ve o sınıftan sunucu ile iletişime geçiyor.

```
@Override
public void run() {
    synchronized (this) {
    }
    notify();
}
```

SimpleClient sınıfını inceliyelim.

Obje çalıştıktan sonra SimpleClient sınıfı çağırılır.

İlk başta sunucuya localhso adresinden ve porttan bağlanalım.

```
Socket s = new Socket( host: "localhost",  port: 8000);
```

Sonrasında Server sınıfında accept fonksiyonu çalışmış olacak ve sonraki itersyona geçecek

```
System.out.println("Client connected");
SimpleServer simpleServer = new SimpleServer(server);
simpleServer.start();
```

Burda SimpleServer sınıfı çağırılacak ve sonrasında SimpleServer ve SimpleClient,yani sunucu ve istemci arasında mesajlaşma yapılacak.

İstemci tarafından oyuncu sayısını sunucuya gönderelim. Bunu ne amaçla yaptığımı aşağıda açıklıyacağım.

```
//amount of active client
PrintWriter prwrt = new PrintWriter(s.getOutputStream());
prwrt.println(amountOfClient);
prwrt.flush();
```

Ondan sonra SimpleServer a DB sınıfından veriyi çekip istemciye gönderelim. Bunun için değişken yapıp ve DB sınıfından get metotlarını çağırıp bu değişkenlere verileri atıyalım.

```
id = client.getId();
answer = client.getAnswer();
question = client.getQuestion();
```

Sonra soruyu sunucu tarafından ekrana yazdırıp soruyu istemciye gönderelim.

```
System.out.println("Question: " + question);

PrintWriter prr = new PrintWriter(server.getOutputStream());
prr.println(question);
prr.flush();
```

Gönderilen mesajı istemci tarafından yakalıyalım ve istemci tarafından ekrana soruyu yazdıralım.

```
//question sent by server
InputStreamReader inn = new InputStreamReader(s.getInputStream());
BufferedReader bff = new BufferedReader(inn);
String strr = bff.readLine();
System.out.println("Question: " + strr);
```

Soruya cevaplmak için sunucuya mesajımızı gönderelim.

```
//answer of client
PrintWriter pr = new PrintWriter(s.getOutputStream());
System.out.println("Answer of question: ");
pr.println(sc.next());
pr.flush();
```

Sunucu tarafından cevabı yakalıyalım ve cevabı ekrana yansıtalım.

```
InputStreamReader in = new InputStreamReader(server.getInputStream());
BufferedReader bf = new BufferedReader(in);
String str = bf.readLine();
System.out.println("Answer of client: " + str);
```

Bunan sonra gelen cevabı kontrol edelim, yanlış mı doğru mu. Bunun için java nın içinde olan hazır equalsIgnoreCase fonksiyonu kulanalım, bu fonksiyon büyük küçük harf farkına bakmadan kontrol eder. Eğer ki yanlış cevap geldi ise o zaman oyuncu kaybetti mesajını ekrana yazdırsın ve doğru cevabı o oyuncuya yansıtsın. Sonrasında bu mesajı istemciye göndersin ve istemsi tarafında doğru ve yayanlış cevap verdiğini ekrana yazdıralım.

```
PrintWriter pr = new PrintWriter(server.getOutputStream());
System.out.print("Message: ");
if(str.equalsIgnoreCase(answer)){
    System.out.println("the client answered the question correctly.");
    pr.println("your answer is true, so you continue the game...");
    pr.flush();
}
else{
    System.out.println("the client did not answer the question correctly. Game is over!");
    pr.println("your answer is false...Right answer is " + answer+". Game is over for you!!!");
    pr.flush();
}
```

```
//message from server
InputStreamReader in = new InputStreamReader(s.getInputStream());
BufferedReader bf = new BufferedReader(in);
String str = bf.readLine();
System.out.println("Server: " + str);
```

Bundan sonra sunucu tarafından tekrar yanlış ve ya doğru cevap verildiğini kontrol edip bu bilgiyi istemci tarafa gönderelim.

```
PrintWriter prwrt = new PrintWriter(server.getOutputStream());
if(str.equalsIgnoreCase(answer)){
    prwrt.println("true");
    prwrt.flush();
}
else{
    prwrt.println("false");
    prwrt.flush();
}
```

Ve istemci tarafından mesajı alıp correct değişkenine 0 ve ya 1 atıyalım.

```
//to see the correctness of answers
InputStreamReader inpt = new InputStreamReader(s.getInputStream());
BufferedReader bfrd = new BufferedReader(inpt);
String strg = bf.readLine();
if (strg.equals("false"))
    correct=0;
else
    correct=1;
```

Bunun ana amacı Client.java sınıfından obje üzerinden correct değerini kontrol edip ve eğer ki correct değeri 0 eşit ise o zaman o oyuncu yanlış cevap verdi isClient değerini 5 yaparak sonraki oyunda bu oyuncuya girişi engelleriz.

```
if(isClient1==1){
    System.out.println("*********First client*********");
    client1.start();
    synchronized (client1){
        client1.wait();
        if(client1.correct == 0)
            isClient1=5;
    }
    check++;
}
```

Bunları çalıştırdıktan sonra tüm açtığımız BufferedReader fonksiyonlarını kapatalım sunucu ve istemci taraflarında ve elave sunucu tarafından sunucuyu da kapatalım. Bir sonraki oyunda diğer oyuncu için sunucu tekrar açılacak.

```
bfrd.close();
bff.close();
bf.close();
```

```
bfread.close();
bf.close();
server.close();
```

Tüm oyuncular soruları cevapladıktan sonra ve while dögünün sonuna erdikten sonra kontrol yapalım, eğer 4 kişiden 3 yanlış cevap verdi ise o zaman son doğru cevap veren kazanır, eğer sonunda herkes yanlış cevap verdi ise kimse kazanmaz, eğer ki doğru cevap veren oyuncu sayısı ikiye eşit ve ya fazla ise döngü devam eder.

```
if (isClient1 == 5  && isClient2 == 5 && isClient3 == 5 && isClient4 == 5) {
    System.out.println("Nobody is win!!!");
    break;
}
else if (isClient2 == 5 && isClient3 == 5 && isClient4 == 5) {
    System.out.println("First client is win!!!");
    break;
}
else if (isClient1 == 5 && isClient3 == 5 && isClient4 == 5) {
    System.out.println("Second client is win!!!");
    break;
}
else if (isClient1 == 5 && isClient2 == 5 && isClient4 == 5) {
    System.out.println("Third client is win!!!");
    break;
}
else if (isClient1 == 5 && isClient2 == 5 && isClient3 == 5) {
    System.out.println("Fourth client is win!!!");
    break;
}
```

Database Server Side Programming

MySQL database bağlantıyı sağlamak için giriş yapalım.

```
private static final String URL = "jdbc:mysql://localhost:3306/test?useUnicode=true&useSSL=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=U
private static final String user = "root";
private static final String password = "admin";
```

Database kisminda veriyi rasgele çekmek için Java random fonksiyonu kullanarak bir ve 20 arasında sayı elde edelim, sonra ise o sayıyı id numaraya atıyalım, fakat bir şartla, rasgele seçilen sayı tekrar seçilemez, soruyu sadece tek kez sorup o soruyu tekrardan soramayız. Bu durumu engellemek için bir dizi oluşturup, boyutu soru çeşiti kadar(20 soruluk), o dizin içerisinde her sorulduğunda 0 olan değerleri 1 yapıp böylelikle her zaman if koşulu ile kontrol ederiz, eğer ki gelen sayının dizide ki o sayının indis karşılığında bulunan değer 0 ise o zaman sıfırı bire değiştir ve döngüden çık, fakat 1 ise yeni sayı üret ve tekrarlanmamış sayıya ulaşana kadar döngüye devam et.

```
private static int array[][]={{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}};
public DB(int count) {
    try {
        if (count==0){
            this.num = rand.nextInt( bound: 20) + 1;
            while(true) {
                if (array[1][num - 1] == 0) {
                    array[1][num-1]=1;
                    break;
                }
                else {
                    this.num = rand.nextInt( bound: 20) + 1;
                }
            }
        }
```

Bu koşullar dışında önemli nokta daha var, her 4 istemciye sorulan soru aynı olacak, o yüzden bu durumda num değerimiz DB.java classını çağırdığımızda DB yaoıcısını bir kere çağırırız, tüm istemcilere soru sorulduktan sonra. Bu durumu kontrol etmek için bir if yapısında ihtiyac duyoruz. İf in içerisinde count değişkeni tutalım ve eğer count değeri 0 ise döngüye girmesin, böylelikle num değişkeni değiştirmesin, var olan num değerini saklasın ve her zaman databesden aynı soruyu istemciye iletsin, eğer 0 değil ise demektir ki daha soruya cevap vermeyen istemciler bulunmaktadır.

```
int check;
while(true) {
    check=0;
    SimpleClient client1 = new SimpleClient();
    client1.amountOfClient=check;
    if(isClient1==1){
        System.out.println("*********First client*********");
        client1.start();
        synchronized (client1){
            client1.wait();
            if(client1.correct == 0)
                isClient1=5;
        }
        check++;
    }
    else{
        System.out.println("First player lost the game!The next player is continuing...");
        client1.interrupt();
    }
    SimpleClient client2 = new SimpleClient();
    client2.amountOfClient=check;
    if (isClient2 == 1) {
        System.out.println("*********Second client*********");
        client2.start();
        synchronized (client2) {
            client2.wait();
            if (client2.correct == 0)
                isClient2 = 5;
        }
        check++;
```

Peki şimdi count değerini nasıl değiştirecez, bu kısmını istemci tarafında(Client.java) check değişkenide tutacaz. Her zaman döngü başlatıldığında check değerimiz sıfırlansın, yani demektir ki tüm istemciler soruları doğru veya yanlış cevapladılar ve sunucu soruyu değiştirip farklı soru sunması gerekmektedir. Check değerini yanlız ve yanlız istemci doğru cevap verdiğinde, yani oyunu devam ettirdiğinde artıra biliriz(bu durumu daha sonra açıklıyacağım), aksi taktirde check değeri her zaman 0 dan fazla olur bu değer count değerine gider ve hiç bir zaman 0 olmadığından dolayı ya soru değişmez ya da null değeri atar, yani yanlış sonuca varırız.Peki bu değeri DB.java classına nasıl iletiyorum? Bunun için SimpleClient.java clasında `public int amountOfClient;` değişkeni oluşturalım ve buraya check değerini Client.java clasından atıyalım. Bundan sonra

amountOfClient değerini servera gönderelim. İstemci tarafından göndermek için kullanılan fonksiyonları yazdıralım.

```
//amount of active client
PrintWriter prwrt = new PrintWriter(s.getOutputStream());
prwrt.println(amountOfClient);
prwrt.flush();
```

Sunucu tarafından(SimpleServer.java) alınacak mesajı okumak için kullanılan fonksiyonları yazdıralım.Bize istemciden amountOfClient gelen değerini amount değişkenine atıyıp sonrasında ise DB.java clasında iletilerek DB ye ulaştırmış oluyoruz.

```
//amount of clients
InputStreamReader instr = new InputStreamReader(server.getInputStream());
BufferedReader bfread = new BufferedReader(instr);
int amount = bfread.read()-48;
DB client = new DB(amount);
```

Rasgele sayı üretilip üretilmiyecek koşulu kontrol etikten sonra DB bağlanıp verileri tek tek id, question ve answer değişkenlerine atıyalım.

```
Connection connection = DriverManager.getConnection(URL, user, password);
Statement statement = connection.createStatement();
String query = "select * from usersquest where id="+num;
ResultSet resultSet = statement.executeQuery(query);
while (resultSet.next()) {
    this.id = resultSet.getInt( columnIndex: 1);
    this.question = resultSet.getString( columnIndex: 2);
    this.answer = resultSet.getString( columnIndex: 3);
}
```

Sonrasında bu değişkenler için get metodu oluşturup verileri kolaylıkla DB clasından istediğimiz zaman çeke biliriz.

```
public int getId(){
    return id;
}
public String getQuestion(){
    return  question;
}
public String getAnswer(){
    return answer;
}
```

Source Codes

1) Server Side

Server.java

```
import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;


public class Server {

   public static void main(String[] args) throws IOException {


      ServerSocket ss = new ServerSocket(8000);

      System.out.println("Server is starting!");

      System.out.println("Waiting for clients...");

      while(true) {

         Socket server = ss.accept();

         System.out.println("Client connected");

         SimpleServer simpleServer = new SimpleServer(server);

         simpleServer.start();

      }

   }

}
```

SimpleServer.java

```
import java.io.BufferedReader;
```

```
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

class SimpleServer extends Thread {

   int id;
   String answer;
   String question;
   private Socket server;

   public  SimpleServer(Socket server){
      this.server=server;
   }
   @Override
   public void run(){
      try{
         //amount of clients
         InputStreamReader instr = new InputStreamReader(server.getInputStream());
         BufferedReader bfread = new BufferedReader(instr);
         int amount = bfread.read()-48;
         DB client = new DB(amount);
         id = client.getId();
         answer = client.getAnswer();
         question = client.getQuestion();

         System.out.println("Question: " + question);

         PrintWriter prr = new PrintWriter(server.getOutputStream());
         prr.println(question);
         prr.flush();

         InputStreamReader in = new InputStreamReader(server.getInputStream());
         BufferedReader bf = new BufferedReader(in);
         String str = bf.readLine();
         System.out.println("Answer of client: " + str);
```

```
        PrintWriter pr = new PrintWriter(server.getOutputStream());
        System.out.print("Message: ");
        if(str.equalsIgnoreCase(answer)){
           System.out.println("the client answered the question correctly.");
           pr.println("your answer is true, so you continue the game...");
           pr.flush();
        }
        else{
           System.out.println("the client did not answer the question correctly. Game is over!");
           pr.println("your answer is false...Right answer is " + answer+". Game is over for you!!!");
           pr.flush();
        }
        PrintWriter prwrt = new PrintWriter(server.getOutputStream());
        if(str.equalsIgnoreCase(answer)){
           prwrt.println("true");
           prwrt.flush();
        }
        else{
           prwrt.println("false");
           prwrt.flush();
        }
        bfread.close();
        bf.close();
        server.close();
      }
      catch(Exception e){
        e.printStackTrace(System.out);
      }
   }
}
```

DB.java

```
import java.sql.*;
import java.util.Random;
public class DB {
```

```
    private static final String URL =
"jdbc:mysql://localhost:3306/test?useUnicode=true&useSSL=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=fal
se&serverTimezone=UTC";

    private static final String user = "root";

    private static final String password = "admin";

    private int id;

    private String answer;

    private String question;

    private Random rand = new Random();

    private static int num;

    private static int array[][]={{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}};

    public DB(int count) {

        try {

            if (count==0){

                this.num = rand.nextInt(20) + 1;

                while(true) {

                    if (array[1][num - 1] == 0) {

                        array[1][num-1]=1;

                        break;

                    }

                    else {

                        this.num = rand.nextInt(20) + 1;

                    }

                }

            }

            Connection connection = DriverManager.getConnection(URL, user, password);

            Statement statement = connection.createStatement();

            String query = "select * from usersquest where id="+num;

            ResultSet resultSet = statement.executeQuery(query);

            while (resultSet.next()) {

                this.id = resultSet.getInt(1);

                this.question = resultSet.getString(2);

                this.answer = resultSet.getString(3);

            }

        } catch (SQLException e) {

            System.out.println("Failed to load driver class");

        }
```

```
    }
    public int getId(){
        return id;
    }
    public String getQuestion(){
        return  question;
    }
    public String getAnswer(){
        return answer;
    }
}
```

2) Client Side

Client.java

```
public class Client {
    public static void main(String[] args) throws InterruptedException {
        int isClient1=1;
        int isClient2=1;
        int isClient3=1;
        int isClient4=1;
        int check;
        while(true) {
            check=0;
            SimpleClient client1 = new SimpleClient();
            client1.amountOfClient=check;
            if(isClient1==1){
                System.out.println("*********First client*********");
                client1.start();
                synchronized (client1){
                    client1.wait();
                    if(client1.correct == 0)
                        isClient1=5;
                }
                check++;
```

```
}
else{
   System.out.println("First player lost the game!The next player is continuing...");
   client1.interrupt();
}
SimpleClient client2 = new SimpleClient();
client2.amountOfClient=check;
if (isClient2 == 1) {
   System.out.println("*********Second client*********");
   client2.start();
   synchronized (client2) {
      client2.wait();
      if (client2.correct == 0)
         isClient2 = 5;
   }
   check++;
}
else {
   System.out.println("Second player lost the game!The next player is continuing...");
   client2.interrupt();
}
SimpleClient client3 = new SimpleClient();
client3.amountOfClient=check;
 if (isClient3 == 1) {
    System.out.println("*********Third client*********");
    client3.start();
    synchronized(client3){
       client3.wait();
       if (client3.correct == 0)
          isClient3 = 5;
    }
    check++;
 }
 else{
    System.out.println("Third player lost the game!The next player is continuing...");
```

```
    client3.interrupt();
 }
SimpleClient client4 = new SimpleClient();
client4.amountOfClient=check;
if (isClient4 == 1) {

   System.out.println("*********Fourth client*********");
   client4.start();
   synchronized (client4) {
      client4.wait();
      if (client4.correct == 0)
         isClient4 = 5;
   }
}
else{
   System.out.println("Fourth player lost the game!The next player is continuing...");
   client4.interrupt();
}
if (isClient1 == 5  && isClient2 == 5 && isClient3 == 5 && isClient4 == 5) {
   System.out.println("Nobody is win!!!");
   break;
}
else if (isClient2 == 5 && isClient3 == 5 && isClient4 == 5) {
   System.out.println("First client is win!!!");
   break;
}
else if (isClient1 == 5 && isClient3 == 5 && isClient4 == 5) {
   System.out.println("Second client is win!!!");
   break;
}
else if (isClient1 == 5 && isClient2 == 5 && isClient4 == 5) {
   System.out.println("Third client is win!!!");
   break;
}
else if (isClient1 == 5 && isClient2 == 5 && isClient3 == 5) {
```

```
            System.out.println("Fourth client is win!!!");
            break;
          }
        }
      }
}
```

SimpleClient.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

class SimpleClient extends Thread{

   public int amountOfClient;
   public int correct;
   @Override
   public void run() {
      synchronized (this) {
         try {
            Socket s = new Socket("localhost", 8000);
            Scanner sc = new Scanner(System.in);

            //amount of active client
            PrintWriter prwrt = new PrintWriter(s.getOutputStream());
            prwrt.println(amountOfClient);
            prwrt.flush();

            //question sent by server
            InputStreamReader inn = new InputStreamReader(s.getInputStream());
            BufferedReader bff = new BufferedReader(inn);
            String strr = bff.readLine();
            System.out.println("Question: " + strr);
```

```
            //answer of client
            PrintWriter pr = new PrintWriter(s.getOutputStream());
            System.out.println("Answer of question: ");
            pr.println(sc.next());
            pr.flush();

            //message from server
            InputStreamReader in = new InputStreamReader(s.getInputStream());
            BufferedReader bf = new BufferedReader(in);
            String str = bf.readLine();
            System.out.println("Server: " + str);

            //to see the correctness of answers
            InputStreamReader inpt = new InputStreamReader(s.getInputStream());
            BufferedReader bfrd = new BufferedReader(inpt);
            String strg = bf.readLine();
            if (strg.equals("false"))
               correct=0;
            else
               correct=1;
            bfrd.close();
            bff.close();
            bf.close();
         } catch (IOException e) {
            e.printStackTrace(System.out);
         }
         notify();
      }
   }
}
```