1. **Exercício UDP Avançado – 3.5 pontos**

Envie ao servidor, dessa vez usando um cliente UDP, o número 0 e o nome do seu Jóquei separados por vírgula (ficando assim: “0,”+\*nome do Jóquei\*), esse será um padrão de comunicação que haverá nesse exercício. O servidor UDP não tem as mesmas características de um TCP, então para saber quem são os seus clientes, um ID é atribuído. Ao usar o 0 antes da mensagem você informa ao servidor que ainda não tem um ID, você vai receber uma mensagem que contém WELCOME + \*numero\*, esse numero será o seu ID e você deve guardá-lo para mensagens subsequentes. Mensagens enviadas ao servidor sem um número vão ser interpretadas como tentativas de registrar Jóqueis. Responda a mensagem de WELCOME com o ID +“,OK”, atente-se a virgula.

Essa atividade continua de forma similar a última, o servidor mandará uma mensagem escrita “DESAFIO,”+\*número\*, multiplique esse número por 10 e então divida-o por 15 (O resultado deve ser um número DOUBLE). Por fim, envie ao servidor a seguinte mensagem

\**ID\*+”,RESPOSTA,” + \*resultado\**

Você vai receber uma mensagem indicando se acertou ou não.

|  |
| --- |
| Dica UDP |
| Comunicação UDP e um pouco mais complicada que a TCP, para começar você vai enviar pacotes de bytes e não strings/int/whatever, para começar você deve criar um buffer de recepção e envio  *byte[ ] sendData;*  *byte[ ] receiveData = new byte[1024];*  Nós definimos o valor do receive data porque não podemos saber seu tamanho até recebe-lo então criamos um tamanho máximo pra ele, o sendData não e necessário pois como somos nós que estamos mandando temos o acesso ao seu tamanho.  Agora vamos falar ao Java que a variável receiveData deve ser aquela que as informações recebidas serão armazenadas.  *DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);*  Com isso configurado podemos criar o socket cliente usando o DatagramSocket, desse modo:  *DatagramSocket clientSocket = new DatagramSocket();*  Agora temos que associar o sendData do mesmo modo que fizemos como o receiveData, como um dos parâmetros da função a seguir e o tamanho do sendData tenha certeza que você JÁ COLOCOU A INFORMAÇÃO QUE QUERIA ENVIAR NELE.  *DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 80);*  Acabando a revisão, lembre-se que para as operações você vai usar o:  *clientSocket.send(sendPacket);*  Para enviar enquanto você usa o:  *clientSocket.receive(receivePacket);*  Para receber, lembre-se que as informações que você está recebendo estão no formato byte[]. |

1. **Exercício Ultimate - 3.5 pontos**

Dessa vez vamos nos comunicar usando objetos em java e não strings, para começar verifique o objeto Challenge que você recebeu do professor, nele devem conter as variáveis

*public double numeroDeChecagem;*

*public String nome;*

*public String mensagem;*

Caso esteja tudo certo podemos começar, set o nome do Joquéi num novo objeto Challenge, e envie para o servidor. Enviar objetos javas e possível usando um tipo de OutputStream único, o ObjectOutputStream, você o cria assim:

*ObjectOutput out =new ObjectOutputStream(clientSocket.getOutputStream());*

E você envia o objeto assim:

*out.writeObject(desafio);*

Dessa vez não precisamos nos preocupar em enviar o tamanho do objeto. Você então vai receber o objeto utilizando um ObjectInputStream, declarado dessa forma:

*ObjectInput in= new ObjectInputStream(clientSocket.getInputStream());*

A leitura se da usando a função:

*desafio = (Challenge) in.readObject();*

(desafio e o objeto Challenge que foi criado anteriormente em que o nome foi adicionado, não esqueça do cast)

Sabendo como se da a comunicação por objetos, você deve fazer um novo desafio, dessa vez você deve pegar a variável numeroDeChecagem que veio no objeto e calcular sua raiz quadrada, coloque o resultado na mesma variável numeroDeChecagem e envie o objeto ao servidor.

Receba novamente o objeto do servidor, e peça para imprimir a string mensagem, nela se informará se o resultado foi correto ou não.