**RESUMO DAS CLASSES CLIENTE- SERVIDOR**

**1.Arquitetura Cliente/Servidor**

Para fazer a **Arquitetura Cliente/Servidor,** precisa ver conceitos sobre Sockets, Threads, classe Conjunto.

O **Servidor** roda antes no aguardo das conexões dos clientes para posterior interação. Um servidor pode ter vários clientes se comunicando com o Servidor.

Os **clientes** mandam solicitações ao servidor.

Os **Sockets** realizam a comunicação entre cliente e servidor, geralmente em máquinas distintas.

As Threads ajudam o servidor a responder as solicitações do cliente.

**1.1.Funcionamento da arquitetura:**

**Item 1:**

Se o servidor captar o comando desativar na sua main, o servidor vai mandar um comunicado do servidor inoperante aos clientes; pede para tentar mais tarde.

**Item 2:**

Ao rodar o programa cliente o mesmo se conecta com o servidor. Dentro do servidor terá a variável double sendo zero para cada cliente gerado. O cliente terá várias solicitações para optar: soma, divisão, multiplicação e adição. O cliente pede ao servidor o valor atual da variável. O servidor responde enviando ao cliente as respostas das solicitações; o programa cliente exibe na tela. O cliente pode pedir para sair, no qual o servidor realiza a desconexão daquele cliente.

**2.Classes comuns:**

São 8 classes:

Todas as classes herdam de Comunicado; e implementam Serializible. O Serializable é um interface nativa de java com o uso da biblioteca Import.java.io.\*; outro uso é a gravação de objetos. Implementar significa herdar de uma interface. O Serializible é usado para transmitir objetos em rede da classe Sockets.

As classes que são ocas NÃO têm métodos e nem atributos.

**2.1. Classe Comunicado**

Classe oca. Implementa Serializable e Cloneable.

**2.2.Classe PedidoDeOperação :** mostra as opções de contas. Faz o construtor, o primeiro valor é o que está na classe servidor. Construtor recebe operação e valor. O servidor recebe a operação, e precisa retornar usando o método getOperacao. Para saber o valor, usa o método getValor.

**2.3.Classe Teclado:** permite o usuário digitar as operações, operandos (na classe Cliente); no servidor digitar o comando desativar

**2.4.Classe Parceiro:**

Se usado dentro de Cliente, o parceiro é o Servidor. Se usada no Servidor, o parceiro é o Cliente. *São atributos do* ***construtor*** *são:* Socket, ObjectInputStream receptor, ObjectOuputStream transmissor, são declarados e após o construtor iniciar ficam instanciados.

Os ***demais atributos*** são: comunicado e semaphore.

Tem como ***demais métodos*** ***(classe Parceiro)***:

**Espie:** qual o comunicado (atributo proximoComunicado) que virá sem consumir informação. Se proximoComunicado for nulo recebe objeto lido do receptor e guardado no atributo; só lê atributo nulo.

**Semaphore**: usa um recurso; ao ser instanciado o próximo solicitante fica bloqueado pois a thread que executar o serializable tem exclusividade sobre o próximo. Quando recurso tiver feito release, o próximo fica liberado.

**Receba:** se eu estou na classe Cliente, estou transmitindo ao parceiro Servidor; de modo que o servidor receba a minha informação.

**Envie:** Vai dizer o que foi mandado , pois faz a leitura daquilo que foi enviado. Se o próximo comunicado for nulo, realiza a leitura pela primeira vez. Em leituras repetidas, guarda o proximoComunicado em ret, fica nulo e retorna ret.

**Receba:** usa o parâmetro Comunicado; o programa envia e o parceiro recebe (com writeObject) e flush no transmissor pra transmissão imediata.

**Adeus:** Desconecta do parceiro; fechando transmissor, receptor e conexão.

**2.5. Classe Resultado**

Tem como atributo valorResultante que será declarado inicialmente e instanciado no seu construtor. O valorResultante é objeto de resposta enviado do Servidor quando o cliente faz o PedidoDeResultado. O método getValorResultante retorna o valor capturado do valorResultante. Faz método toString

**2.6. Classe PedidoParaSair**

É uma classe oca. Quando o Cliente passar t para terminar; será enviado a instancia da classe ao servidor para que o mesmo (o servidor) realize a saída.

**2.7. Classe ComunicadoDeDesligamento**

Classe oca. Objetos da classe são acionados quando o servidor escrever Desativar; o Servidor vai mandar o objeto instanciado da classe ComunicadoDeDesligamento aos seus clientes sobre a inoperância do sistema.

**3.Tarefas Simultâneas**

**Item 1:**

Interagir com o cliente em loop ao demonstrar as opções que possui, solicitar um valor que será o segundo operando se chamar PedidoDeOperação; ou mostrar o Resultado se chamar PedidoDeResultado. Demais caso, mostrar erros senão estiver na guia de opções. **As solicitações passam opção e valor.**

Essas ações são feitas no fim da main que rodam junto com a thread depois das tarefas de iniciação.

**Item 2:**

Aguardar em loop e ***um eventual ComunicadoDeDesligamento*** que vier do Servidor para que o Cliente atue. No início da main do cliente, instancia a thread tratadoraDeComunicadoDeDesligamento.

**4.CLASSE CLIENTE**

**Classes usadas para construir o Cliente.java (original do Fazedor de Continhas):**

Socket, ObjectOutputStream (biblioteca java.net.\*), ObjectInputStream (biblioteca java.net.\*), biblioteca java.io.\*, Parceiro, TratadoraDeComunicadoDeDesligamento, Teclado, PedidoDeOperacao, Comunicado, Resultado, PedidoDeResultado, PedidoParaSair.

Classes:

Socket, ObjectOutputStream , ObjectInputStream e Parceiro --> Lunara

TratadoraDeComunicadoDeDesligamento --> Elys

Teclado --> Elys

PedidoDeOperacao --> Lunara

Comunicado --> oca

Resultado --> Elys

PedidoDeResultado --> Lunara

PedidoParaSair --> oca

4.1 **Ações feitas no começo da main**

**Item 1:**

Checar se main, onde tem o vetor args; o ***cliente indicou a porta e máquina usadas pelo servidor.*** Se passar 0 parâmetros, serão adotadas as máquina (localhost) e porta padrões (3000); se passar 1 parâmetro, o outro será o padrão (exemplo: se passou apenas porta, a máquina será localhost; se passou a máquina, a porta será padrão); se passar 2 parâmetros eles serão substituídos no construtor (em relação aos padrões); se passar 3 parâmetros darão inválidos.

**Item 2:**

Fazer conexão com o Servidor, Declarar e instanciar: transmissor (ObjectOutputStream), receptor (ObjectInputStream). Parar com os objetos citados. Instanciar o Parceiro servidor (parâmetros: conexão, receptor e transmissor) que será a comunicação contínua com o servidor. Isso se deve ao fato de que o cliente é um programa proativo que se conecta com o reativo (servidor) com instanciações e declarações de socket e construtor (parâmetros: host, porta).

**Item 3:**

Declara e instancia o objeto tratadoraDeComunicadoDeDesligamento do tipo TratatadoraDeComunicadoDeDesligamento. Passa o Parceiro servidor; main compartilhou o objeto pelo parceiro. Após, startar o objeto para começar o método run da thread referida (a thread é o tipo/classe do objeto).

**4.2.COMO O USUÁRIO VAI RODAR OS PROGRAMAS CLIENTE E SERVIDOR?**

Não pode rodar em ambiente de desenvolvimento, como compilar e executar.

Então, devemos usar a **prompt de comando.**

Fazemos:

\>;C // o \ é para indicar a raíz, :\> é cursor piscando; C: é a pasta C (a maior hierarquia).

Fazer :D + enter trocar de diretório;

Fazer CD nomeDaPasta é para trocar de pasta dentro do diretoria.

Fazer Java Servidor + enter faz rodar o servidor Java.

**Prompt de comando Servidor:**

\>C:

C:\ >CD “ Fazedor de Continhas Cliente-Servidor”

C:\ Fazedor de Continhas Cliente- Servidor > CD Servidor

C:\ Fazedor de Continhas Cliente- Servidor\Servidor> JAVA Servidor

O servidor esta ativo! Para desativa-lo,

Use o comando “desativar”

>desativar

>O servidor foi desativado!

*Para passar as informações para o programa:*

**Prompt de comando Cliente:**

C:\ Fazedor de Continhas Cliente- Servidor > Cliente> java Cliente

Indique o servidor e a porta corretos!

C:\ Fazedor de Continhas Cliente- Servidor > Cliente> JAVA Cliente localhost 7000

Sua opção (+, -, \*, /, =, [T]erminar)?

Passar as informações para o vetor args: localhost vai na posição zero (nome da máquina ou ip); enquanto que o 7000 vai para a posição 1 (número int da porta).

**5.CLASSE SERVIDOR**

**Classes usadas para construir o Servidor.java (original do Fazedor de Continhas):**

Biblioteca java.util.\*, Arraylist <Parceiro>, AceitadoraDeConexao, Teclado, equals, ComunicadoDeDesligamento, SupervisoraDeConexao

**5.1.Ações de inicialização a serem realizadas no servidor;**

**Item 1**

Ver os parâmetros vindos para args para que o usuário diga a porta para aceitar as conexões do Cliente. Sinalizar erros; se a porta for nula, adotar o padrão 3000.

**Item 2**

Declarar e instanciar a estrutura de dados ArrayList de nome usuários para agrupar objetos da classe Parceiro; nessa estrutura terá os acessos dos Clientes conectados. Comunicar os clientes.

**Item 3**

Declarar, iniciar e statar uma AceitadoraDeConexao (parâmetros: porta, usuários). O objeto aceitadoraDeConexao vai startar uma supervisoraDeConexao (tipo/classe SupervisoraDeConexao) . Será 1 aceitadora: 1 supervisora.

**5.2. Ações das Tarefas Simultâneas (após as ações de 5.1.)**

**Item 1**

***O item 1 acontece no fim da main após as threads startadas.***

Em loop interagir com o usuário administrador, avisando que o servidor está ativo e se quiser pode desativar o servidor pelo mesmo comando (desativar). Espera o usuário digitar; se acionado o comando, mandar um objeto comunicadoDeDesligamento (tipo ComunicadoDeDesligamento) a todos os usuários. (Parceiros clientes de ArrayList).

**Item 2**

***Uso da thread AceitadoraDeConexao startada.***

Inicialmente, declarar e instanciar o ServerSocket conexão (porta) para aceitar pedidos de conexões com o método accept; guarda os dados no atributo usuários (vindo da main depois de instanciar a thread AceitadoraDeConexao). Depois, continuamente no método run; aceita novas conexões solicitadas pelos clientes, declara, instancia e starta uma supervisoraDeConexao.

**Item 3**

***Uso da thread SupervisoraDeConexao startada.***

*Inicialmente:*

Declara e instanciam os objetos transmissor (ObjectOutputStream) e receptor (ObjectInputStream) associados a conexão que ela supervisiona. A conexão veio como parâmetro no construtor SupervisoraDeConexao após ser mandada pela thread AceitadoraDeConexao.

*Depois:*

Declarar e instanciar o objeto base de todo o programa: Parceiro cliente. Mantém o cliente na estrutura de dados cliente afim de receber os comunicados do Servidor. (O Servidor manda um ComunicadoDeDesligamento aos seus clientes).

*Após:*

Entra num loop para receber o que foi enviado pelo Parceiro cliente; atende as solicitações e o servidor envia o Resultado ao cliente que deseja; outros pedidos que podem vir do cliente: PedidoDeResultado, PedidoDeOperacao. PedidoParaSair

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**6. CLASSES DO JOGO 1 DE FORCA (feito NO ECLIPSE)**

**6.1.Pasta forca2**

BancoDePalavras, ControladorDeErros, ControladorDeLetrasJaDigitadas, Forca, Palavra, Teclado, Tracinhos

**6.2. Pasta forca2Testes**

TesteControladorDeErros, TesteControladorDeLetrasJaDigitadas, TestePalavra, TesteTracinhos.

**6.3. VER QUEM É CLIENTE E SERVIDOR E SUBDIVIDIR AOS GRUPOS:**

**CLIENTE:**

**SERVIDOR:**

**6.4. Observações da Aula de 10-11-2021**

* A cada 3 jogadores fecha uma sala para jogar a Forca em rede.
* Agrupar por ordem de pedidoDeConexao: fazer vetor a partir do zero. Por exemplo, 0,1,2 serão o grupo 1.
* Um jogador só perde se: errar a palavra ou errar a última letra.
* A ordem do pedidoDeConexao determina a ordem das partidas dentro da sala.
* Se o jogador errar a letra e não for a última letra, é a vez do próximo dentro da sala.
* Ganha o jogo quem acertar a palavra ou acertar a última letra da palavra primeiro.
* Colocar palavras grandes no BancoDePalavras (exemplos: paralelepípedo, otorrinolaringologista).
* O primeiro dos três dentro da sala que perder, vai sair do jogo; e o jogo continua para os dois que restaram.
* Usar a classe Conjunto para agrupar o vetor e fazer conjunto de conjuntos, tal como:

Conjunto<Parceiro> grupo = new Conjunto<Parceiro> (3);  
Conjunto<Conjunto<Parceiro>> todoMundo =  
new Conjunto<Conjunto<Parceiro> (50);

* Um jogador pode: PedidoDeConexao (fica esperando ter uma sala vaga pra ter um grupo) , PedirParaJogar (se quer uma palavra ou letra) , PedidoDeResultado (ver pontuação, quantas letras faltam, se o desenho/traços está completo, revelar para todos quantas letras tem repetidas quando acerta a letra) PedirParaSair.
* Adaptar as classes de Pedidos conforme as classes do Jogo 1 de Forca.