Faculdade: Universidade Tecnológica Federal do Paraná

Curso : Especialização em Tecnologia Java

Matéria : Java Aplicado A Redes De Computadores

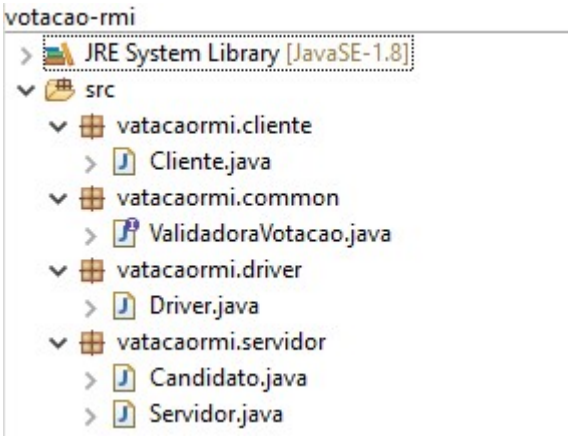
Nome : Erik Eduardo Valcezio RA: 02329611

DATA : 31/08/2021

Atividade 6

- **O servidor RMI deverá estar apto a realizar duas funções:**
- Contar todos os votos;
- Receber votos (contendo os nomes dos candidatos e os números de votos);
- Cada urna (cliente RMI) deverá enviar os nomes e o número de votos de cada candidato para o servidor;
- O servidor deverá exibir a apuração de votos atualizada a cada 5 segundos;

Work Projeto:



GIT: https://github.com/erikvalcezio/java-projects-utfpr/tree/develop/Java-Aplicado-A-Redes-De-Computadores/votacao-rmi

Resumo do projeto:

Interface é o contrato representada pela classe: **ValidadoraVotacao**, que foi implementada para Cliente e o Servidor.

Cliente:

A classe que representa a URNA é a **Cliente** ;

Servidor:

A classe que representa a apuração dos votor a **Servidor** nela contem as implementações dos métodos da interface;

Nesse exemplo foi usado threads para chamar as execuções em simutaneo pela classe **Driver** simulando a **URNA (Cliente)** e **Apuração(Servidor);**

<u>Testes na aplicação em tempo de execução (runtime):</u>

Cada urna (cliente RMI) deverá enviar os nomes e o número de votos de cada candidato para o servidor; *(representada pelo método* `receberVoto(num, "nome do candidato");`

NOTA*: Na classe Cliente é possivel cadastrar o candidado via método remoto* `cadastrarCandidatos(num, "nome do candidato");`

```
Driver (7) [Java Application] C:\Program Files\Java\jdk1.8.0_291\bin\javaw.exe  (31 de ago. de 2021 22:19:52)

=========== Verificar apuração ===========

 Nome Ciro Gomes, Totais de votos 0
 Nome Datena, Totais de votos 0
 Nome Jair Bolsonaro, Totais de votos 0
 Nome João Dória, Totais de votos 0
 Nome Lula, Totais de votos 0
 Nome Sérgio Moro, Totais de votos 0
 Nome Branco / Nulo, Totais de votos 0

=========== consultar um candidato pelo seu número ===========

João Dória
```

Servidor exibe apuração dos Votos a cada 5 segundos (foi realizado loop do lado cliente para fica votando em modo randômico):

```
Apuração atualziada em: Tue Aug 31 22:30:24 BRT 2021
 Nome Ciro Gomes, Totais de votos 84
 Nome Datena, Totais de votos 80
 Nome Jair Bolsonaro, Totais de votos 108
 Nome João Dória, Totais de votos 71
 Nome Lula, Totais de votos 91
 Nome Sérgio Moro, Totais de votos 92
 Nome Branco / Nulo, Totais de votos 100

Apuração atualziada em: Tue Aug 31 22:30:29 BRT 2021
 Nome Ciro Gomes, Totais de votos 84
 Nome Datena, Totais de votos 80
 Nome Jair Bolsonaro, Totais de votos 110
 Nome João Dória, Totais de votos 72
 Nome Lula, Totais de votos 91
 Nome Sérgio Moro, Totais de votos 93
 Nome Branco / Nulo, Totais de votos 101

Apuração atualziada em: Tue Aug 31 22:30:34 BRT 2021
 Nome Ciro Gomes, Totais de votos 84
 Nome Datena, Totais de votos 81
 Nome Jair Bolsonaro, Totais de votos 110
 Nome João Dória, Totais de votos 74
 Nome Lula, Totais de votos 91
 Nome Sérgio Moro, Totais de votos 95
 Nome Branco / Nulo, Totais de votos 101

Apuração atualziada em: Tue Aug 31 22:30:39 BRT 2021
 Nome Ciro Gomes, Totais de votos 85
 Nome Datena, Totais de votos 81
 Nome Jair Bolsonaro, Totais de votos 110
 Nome João Dória, Totais de votos 75
 Nome Lula, Totais de votos 91
 Nome Sérgio Moro, Totais de votos 97
 Nome Branco / Nulo, Totais de votos 102
```

Codificação do projeto abaixo na ordem das classes

Driver(run),

ValidadoraVotacao(Interface),

Cliente(URNA),

Servidor(Implementação),

Candidato(Entidade/Servidor);

```java
package vatacaormi.driver;

import vatacaormi.servidor.Servidor;
import vatacaormi.cliente.Cliente;

public class Driver {
    /**
     * @author Erik Eduardo Valcezio RA: 02329611 Faculdade: Universidade
     *         Tecnológica Federal do Paraná Curso: Especialização em
     Tecnologia
     *         Java/Java Aplicado A Redes De Computadores
     *
     * @version 1.0.0, 29/08/2021
     * @throws InterruptedException
     */

    /*
     * O servidor RMI deverá estar apto a realizar duas funções: Contar
     todos os
     * votos; Receber votos (contendo os nomes dos candidatos e os números
     de votos)
     * Cada urna (cliente RMI) deverá enviar os nomes e o número de votos de
     cada
     * candidato para o servidor; O servidor deverá exibir a apuração de
     votos
     * atualizada a cada 5 segundos;
     */

    public static void main(String[] args) throws InterruptedException {
        new Thread(() -> Servidor.iniciarServidorRmi(), "SERVIDOR
     VOTACAO").start();

        new Thread(() -> Cliente.iniciarClienteRmiUrna(), "URNA").start();

    }
}
```

```java
package vatacaormi.common;

import java.rmi.Remote;

public interface ValidadoraVotacao extends Remote {
    /**
     * usada como common entre cliente servidor para simular execercio proposto
     */

    /**
     * Mostra apuração dos votos
     * @throws RemoteException
     */
    void apuracaoDeTodosVotos() throws RemoteException;

    /**
     * Receber o parâmetros e contabilizar no sistema de votação
     * Obs: não foi implementado com stream devido ao custo que tem na aplicação.
     * @param numeroCandidato
     * @param nomeCandidato
     * @return
     * @throws RemoteException
     */
    boolean receberVoto(int numeroCandidato, String nomeCandidato) throws RemoteException;

    /**
     * Retorna o candidato pelo seu número
     * @param numeroCandidato
     * @return
     * @throws RemoteException
     */
    String consultarCandidatos(int numeroCandidato) throws RemoteException;

    /**
     * Exibe todos candidos
     * @throws RemoteException
     */
    void consultarCandidatos() throws RemoteException;

    /**
     * Cadastra o candidato para participação da votação
     * @param nome
     * @param numero
     * @throws RemoteException
     */
    void cadastrarCandidatos(String nome, int numero) throws RemoteException;
```

```
48
49 }
50
```

```java
package vatacaormi.cliente;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

import vatacaormi.common.ValidadoraVotacao;

public class Cliente {

    public static void iniciarClienteRmiUrna() {

        try {

            //Permissão para acesso externo
            //System.setProperty
("java.security.policy","file:rmiprj.policy");

            //acessa o serviço de registro
            Registry registro = LocateRegistry.getRegistry("127.0.0.1",
Registry.REGISTRY_PORT);

            //procurar a referencia
            ValidadoraVotacao stub = (ValidadoraVotacao) registro.lookup
("validadoraVotacao");

            //cadastrar candidatos
            stub.cadastrarCandidatos("Ciro Gomes", 12);
            stub.cadastrarCandidatos("Datena", 17);
            stub.cadastrarCandidatos("Jair Bolsonaro", 51);
            stub.cadastrarCandidatos("João Dória", 45);
            stub.cadastrarCandidatos("Lula", 13);
            stub.cadastrarCandidatos("Sérgio Moro", 66);
            stub.cadastrarCandidatos("Branco / Nulo", 0);

            System.out.println("\n========== Verificar apuração ==========
\n");
            stub.apuracaoDeTodosVotos();

            System.out.println("\n========== consultar um candidato pelo
seu número ==========\n");
            System.out.println(stub.consultarCandidatos(45));

            System.out.println("\n========== consultar todos os candidatos
==========\n");
            stub.consultarCandidatos();

            System.out.println("\n========== SIMULAÇÃO - VOTAÇÃO NA URNA
```

```
  ===========\n");
45              for (int i = 0 ; i < 10000 ; i++) {
46              List <Integer> rangeCandidatos = Arrays.asList(12 , 17, 51, 45,
  13, 66, 0);
47              Collections.shuffle(rangeCandidatos);
48              stub.receberVoto(rangeCandidatos.get(0),stub.consultarCandidatos
  (rangeCandidatos.get(0)));
49              Thread.sleep(1000);
50              }
51
52          } catch (Exception e) {
53              System.err.println("Excecao: " + e.toString());
54              e.printStackTrace();
55          }
56
57      }
58 }
59
```

```java
package vatacaormi.servidor;

import java.rmi.RemoteException;

public class Servidor implements ValidadoraVotacao {

    public Servidor() throws RemoteException {
        super();
    }

    private static final List<Candidato> apuracacao = new ArrayList<>();

    @Override
    public void apuracaoDeTodosVotos() throws RemoteException {
        apuracacao.forEach(candidato -> System.out.println(
                String.format(" Nome %s, Totais de votos %s", candidato.getNome(), candidato.getNumeroTotalDeVotos())));
    }

    @Override
    public boolean receberVoto(int numeroCandidato, String nomeCandidato) throws RemoteException {
        for (Candidato candidatos : apuracacao) {
            if (candidatos.getNumero() == numeroCandidato && candidatos.getNome().equals(nomeCandidato)) {
                candidatos.setNumeroTotalDeVotos(candidatos.getNumeroTotalDeVotos() + 1);
                return true;
            }
        }
        return false;
    }

    @Override
    public String consultarCandidatos(int numeroCandidato) throws RemoteException {
        for (Candidato candidatos : apuracacao) {
            if (candidatos.getNumero() == numeroCandidato) {
                return candidatos.getNome();
            }
        }
        return String.format("Não localizado candidato com o número: %s", numeroCandidato);
    }

    @Override
    public void consultarCandidatos() throws RemoteException {
        apuracacao.forEach(System.out::println);
    }

```

```
53      @Override
54      public void cadastrarCandidatos(String nome, int numero) throws
  RemoteException {
55          apuracacao.add(new Candidato(nome, numero));
56
57      }
58
59      public static void iniciarServidorRmi() {
60
61          try {
62              // lancar ao servidor de registro
63              Registry registro =
  LocateRegistry.createRegistry(Registry.REGISTRY_PORT);
64
65              // criar skeleton
66              Servidor servidor = new Servidor();
67              ValidadoraVotacao skeleton = (ValidadoraVotacao)
  UnicastRemoteObject.exportObject(servidor, 0);
68
69              // Inserir no servidor de registro uma referencia aos metodos
70              registro.rebind("validadoraVotacao", skeleton);
71
72              while (true) {
73                  Thread.sleep(5000);
74                  System.out.println("\nApuração atualziada em: " +
  Calendar.getInstance().getTime());
75                  servidor.apuracaoDeTodosVotos();
76              }
77
78          } catch (Exception e) {
79
80              System.err.println("Excecao: " + e.toString());
81              e.printStackTrace();
82
83          }
84
85      }
86
87  }
88
```

```java
package vatacaormi.servidor;

import java.io.Serializable;

public class Candidato implements Serializable {

    /**
     * Armazena apuração dos candidatos
     */
    private static final long serialVersionUID = 5029210186399830500L;

    private String nome;
    private int numero;
    private int numeroTotalDeVotos;

    Candidato() {
    }

    public Candidato(String nome, int numero) {
        this.nome = nome;
        this.numero = numero;
        this.numeroTotalDeVotos = 0;
    }

    public Candidato(String nome, int numero, int voto) {
        this.nome = nome;
        this.numero = numero;
        this.numeroTotalDeVotos = voto;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getNumero() {
        return numero;
    }

    public void setNumero(int numero) {
        this.numero = numero;
    }

    public int getNumeroTotalDeVotos() {
        return numeroTotalDeVotos;
    }

```

```
51      public void setNumeroTotalDeVotos(int numeroTotalDeVotos) {
52          this.numeroTotalDeVotos=numeroTotalDeVotos;
53      }
54
55      @Override
56      public String toString() {
57          return "Nome do Candidato: " + nome +", número do candidato: " +
  numero + "\n";
58      }
59
60  }
61
```