UTFPR – Universidade Tecnológica Federal do Paraná
Prof. Titular Dr. Rogério Santos Pozza • Prof. Adjunto Esp. Djeizon Barros

Enunciado da atividade e observações especiais iniciais

Esta é a última atividade obrigatória da disciplina. Eventuais dúvidas sobre esta atividade deverão ser postadas exclusivamente no fórum da disciplina ou no Discord: **não serão aceitos e-mails individuais ao professor ou adjunto; ou, pedido de suporte individual privado do Discord.** Essa atividade tem a característica exclusiva de ter as dúvidas sanadas na metodologia coletiva e compartilhada. Todos estão convidados a compartilhar as dúvidas.

Uma cidade está realizando uma eleição. Você é o desenvolvedor de uma urna eletrônica a qual possui uma função cujo recurso só deve ser utilizado ao presidente da seção (e, portanto, não ao eleitor). Esta urna especial é capaz de realizar o cômputo de cada outra urna espalhada na cidade – no entanto, o presidente da seção o fará manualmente com entrada de dados.

A implementação deverá ser feita com RMI.

O servidor deverá estar apto a realizar as seguintes funções:

- Contar todos os votos que recebe do cliente, para cada candidato. Esses votos vêm do cliente, que está recebendo os dados com entrada manual.
- Atualizar a apuração dos votos a cada 5 segundos.

O cliente deverá estar apto a realizar as seguintes funções:

- Cada urna deverá enviar os nomes e o número de votos de cada candidato para o servidor. Os nomes de candidatos poderão ser pré-configurados (*hardcoded*) ou poderão ser cadastrados pelo presidente da seção.
- Cada candidato também poderá ter um respectivo número de chapa.
- Não é preciso definir nenhuma sigla de partido político.
- Apesar da possibilidade de cadastrar ou não os candidatos, não pré-programe inserção automática de votos, pois é presidente da seção que deverá fazê-lo.

Atenção para as peculiaridades da urna.

Sua aplicação deverá receber ou conter prenomes reais de candidatos (ao oposto como adjetivos, ironias, substantivos, etc.) – exemplos: **João**, **Maria**, **Vitor**, **Paulo**, etc.

O presidente da seção deverá interagir com seu programa. O nome do candidato (ou o número do mesmo) deverá ser entrado. Em seguida, o presidente entrará uma quantidade de votos determinada àquele candidato.

Por exemplo: se determinado candidato foi votado em Curitiba com 500 votos, o presidente entrará o nome ou número deste candidato e assinará 500 votos ao candidato; e depois, ser capaz de reentrar o mesmo candidato e computar mais 400 votos ao mesmo candidato, e assim por diante, com todos os candidatos.

A simulação é mais parecida com uma apuração interna de **Tribunal Regional Eleitoral**, do que uma simulação de eleição em que se tem eleitor por eleitor. Na votação por eleitor, cada eleitor vota um único voto - nesta implementação, não é o caso – preste bem atenção nisso.

Ao desenvolvedor é solicitado que não se quebre os padrões de programação: **se seus programas possuem interface em português, continue com este idioma.** Não faz o menor sentido o aluno programar uma atividade que é codificada com *strings* em português, e no próximo programa, passar a usar tudo em inglês (o que é indício de cópia). Mesmo não o sendo comprovada a cópia, isso sofrerá desconto.

Exemplo disso:

Atividade anterior – com uso do português.
Urna.java

Atividade de agora – com uso do inglês.
VotingBallot.java

O mesmo se aplica ao idioma do programa – console ou gráfico.

Como anteriormente dito, seu programa deverá permitir que ao presidente da seção, sejam apresentados os candidatos (*hardcoded* ou mediante cadastro), e que depois, o presidente da seção outorgue a cada candidato, um número de votos que vêm recebendo de outros lugares.

Este programa deverá ser desenvolvido da seguinte forma**: servidor em modo console** (obrigatório) e **cliente em modo gráfico**, sendo aceitas as implementações de cliente em modo console, caso o aluno desejar. Nenhum dos dois métodos terá prevalência de nota sobre o outro. O que se pede é apenas que o enunciado está descrevendo.

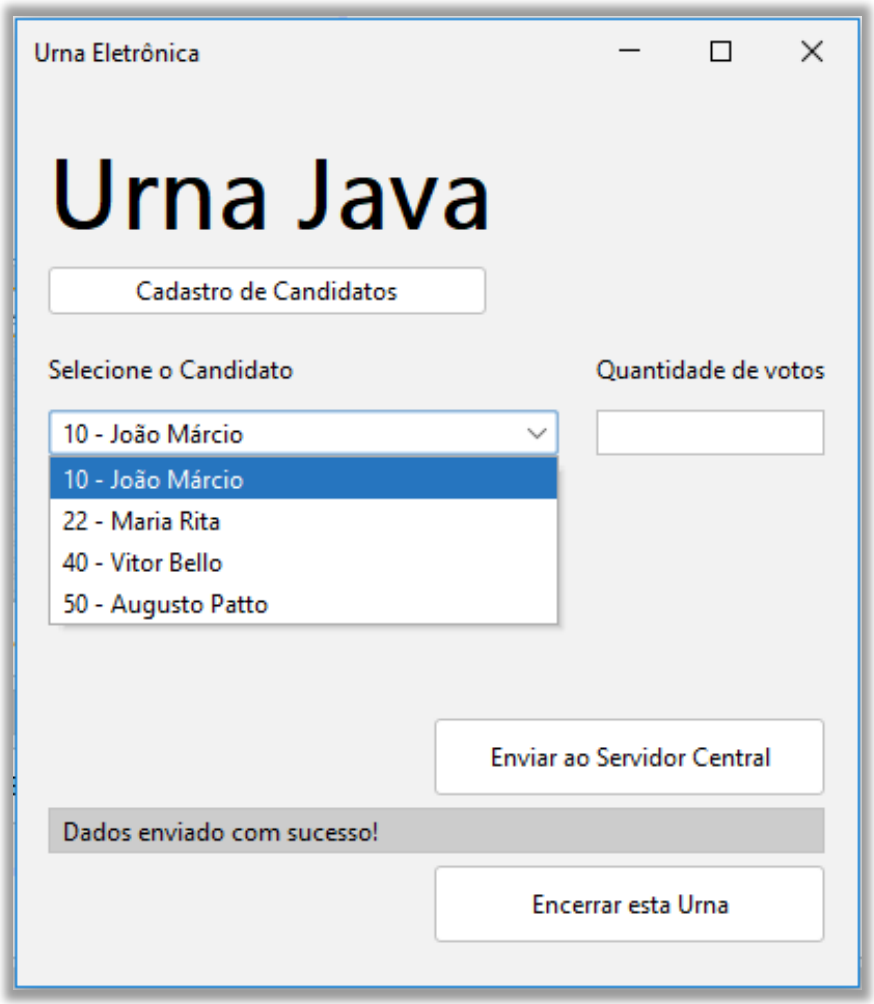
O RMI **não poderá ter sua chamada** via linha de comando para que seu aplicativo funcione – exemplo:

```
rmiregistry 1099
```

Tudo deverá funcionar apenas com a IDE. **O exercício que exigir a abertura de linha de comando acima para depois rodar na IDE, não pontuará nesta atividade.**

Não há necessidade de uso de *Threads* nessa aplicação.

Utilize a figura abaixo como modelo / parâmetro para o cliente gráfico – alguns botões são opcionais (**Cadastro de Candidatos** e **Encerrar esta Urna**):



Utilize a figura abaixo como modelo / parâmetro para o cliente console – a funcionalidade "**Continuar (S/N)?"** é opcional.

```
Urna Java
---------

Candidatos:

10 - João Márcio
22 - Rita Maria
40 - Vitor Bello
50 - Augusto Patto

Entre o número do candidato:

10

Entre o número de votos:

400

Votos enviados.

Continuar (S/N)?
```

Opcionalmente também, pode-se implementar o cadastro no console:

```
Urna Java
---------


1 - Cadastrar.
2 - Realizar cômputo de votos.
3 - Sair da aplicação.

1

Nome: Paola Neves
Número: 60

Candidato cadastrado.

1 - Cadastrar.
2 - Realizar cômputo de votos.
3 - Sair da aplicação.
```

Caso o aluno implemente cadastro, **faça-o como convier**, contanto que se tenha a certeza de não se prejudicar com a funcionalidade principal do programa.

Utilize a figura abaixo como modelo / parâmetro para o servidor console. **Não serão aceitos servidores gráficos.**

```
Eleição
-------

Votos apurados:

10 João Márcio --- 400 votos
22 Rita Maria ---- 500 votos
40 Vitor Bello --- 200 votos
50 Augusto Patto - 200 votos

Votos apurados:

10 João Márcio --- 400 votos
22 Rita Maria ---- 500 votos
40 Vitor Bello --- 600 votos
50 Augusto Patto - 200 votos

Votos apurados:

10 João Márcio --- 900 votos
22 Rita Maria ---- 550 votos
40 Vitor Bello --- 600 votos
50 Augusto Patto - 200 votos
```

As regras para construir o programa

A centralização de tela da janela gráfica (caso opte por cliente gráfico) não será exigida nessa atividade, pois o aluno terá a escolha de realizar cliente no **modo console** ou **modo gráfico.** No entanto, no mundo e no mercado, **o capricho em seus programas** fará o diferencial entre um mar de pessoas despreparadas.

Lembre-se que o cliente não deve nunca iniciar o servidor. O servidor roda um serviço que é independente do cliente, **portanto o servidor deverá ser iniciado primeiro.**

- O seu projeto deverá ter **seu nome como autor**, em um comentário. A classe que não tiver seu nome como autor, renderá desconto global no exercício. Exemplo:

```
/**
 *
 * @author Djeizon Barros
 *
 */
```

- O seu projeto deverá ser entregue com a package **`local.javaredes`**

```
package local.javaredes;
```

  O aplicativo que for entregue com *package* diferente, receberá desconto.

- Atenção, para este exercício, **não fechar a conexão do servidor** e **não precisa fechar do cliente,** porém, se preferir, implementar o fechamento da urna. Não implementar fechamento do servidor. O fechamento da conexão do servidor após receber dados renderá um desconto de **25%.** Não é necessário mais do que um cliente, porém ambos rodam para alimentar o sistema de entrada e cômputo de votos.

- **É preciso que o RMI seja implementado**, baseado no material didático da semana. **Se não houver a implementação de RMI, o programa receberá a nota zero (0).**

- **Apenas um cliente (urna) é suficiente.**

Dica adicional: Utilize de todo o seu conhecimento para a criação de classes e interfaces que utilizou durante toda a disciplina, tal como por exemplo, uma classe **Candidato.java** separada, para tratar deste objeto, e assim, sucessivamente.

Regras da Entrega

Anexe na atividade, os arquivos fonte **.java** e eventuais arquivos *.form*.

Há alunos que conseguem desenvolver esse exercício sem a necessidade de arquivos .form, gerados pela IDE, porém se não for o seu caso, não se esqueça desses arquivos.

Não compactar nenhum arquivo. O envio compactado, como é exigido nas outras disciplinas, **mas não nesta**, acarretará desconto.

Submeta **os arquivos** no ambiente de ensino.
Não deixar a tarefa em Modo Rascunho.
Clique no botão **Enviar Tarefa por Definitivo**.

Atenção para as penalidades no exercício

Quando o exercício recair em uma dessas situações e acumular erros, **os descontos abaixo serão aplicados.**

Situação de descontos cumulativos	**Desconto**
Classes (ou uma delas) não possuem um comentário indicando sua autoria.	10,00
Entregou o exercício com *package* diferente.	15,00
Entregou o exercício em formato ZIP ou RAR ou outro, que é pedido nas outras disciplinas, **mas não nesta.**	15,00
Quebrou o paradigma de programação (idioma diferente do usualmente utilizado em outros programas na disciplina).	30,00
Não foi utilizada a porta 1099, registrada para o RMI.	30,00
Foi utilizado um IP de LAN – e não um endereço do escopo 127/8 – o programa só funciona com a correção para o *localhost*.	30,00
O servidor não atualiza a apuração a cada 5 segundos.	50,00
Há problemas no cliente e/ou no servidor que são passíveis de desconto, em que, no código, se observa a falta de atenção e a falta de leitura do material didático; das explicações das videoaulas e do entendimento do funcionamento do RMI.	60,00
O código não compila. O código é incompreensível.	100,00
Cópia de outro aluno, não importando se tudo está correto.	100,00
O programa foi entregue, mas não há implementação de **RMI.**	100,00