

Laboratório 2

Java RMI

Alunos: Arthur João Lourenço e Larissa G. Rosa

Disciplina: INE 5418 - Computação Distribuída

Como Rodar:

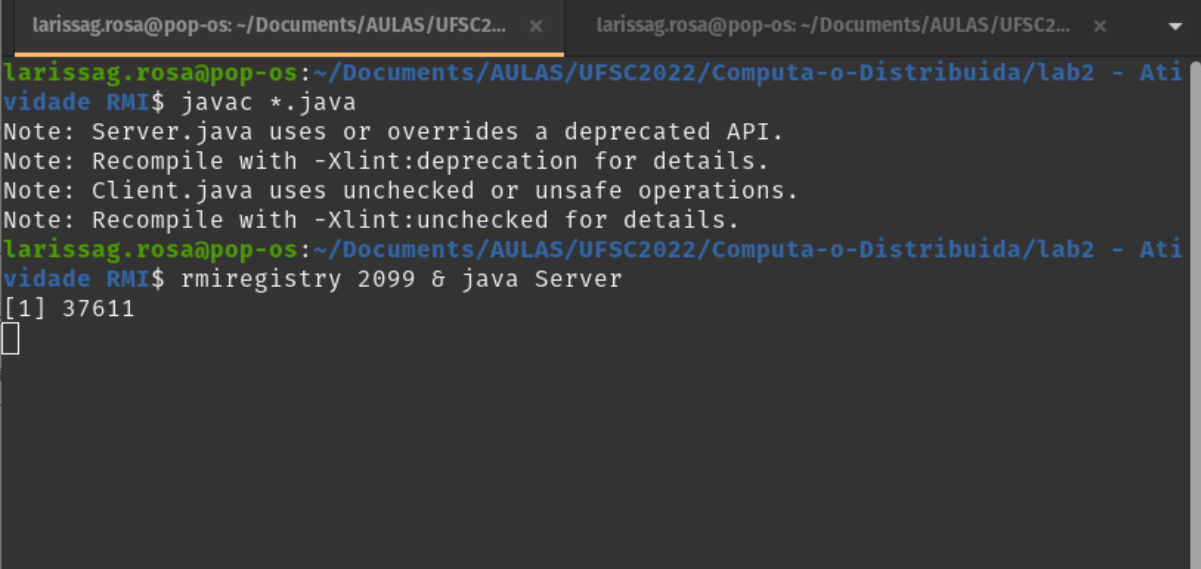
Para rodar o projeto, é preciso estar rodando a versão 11 do Java OpenJDK, compilar as classes, iniciar o registry e então, inicializar as classes Server e Client, como segue:

1. Em uma aba do terminal, colocar o comando **javac *.java**;
2. Inicializar o RMI registry e rodar a classe do Server: **rmiregistry 2099 & java Server**
 - Note que a porta precisa estar aberta e não estar sendo utilizada por outro processo, caso essas condições não sejam satisfeitas, é preciso rodar o servidor em outra porta ou finalizar o processo rodando atualmente, através do comando **kill**.
3. Em outra aba do terminal, digitar **java Client**.

O servidor iniciará corretamente caso não apresente nenhuma mensagem de erro. Já o cliente deve abrir um menu com 5 opções de operações: add, get, remove, size e sair.

Exemplos de Uso:

Como citado no item anterior, primeiro compilamos as classes java e inicializamos o servidor, como exemplo abaixo:



```
larissag.rosa@pop-os: ~/Documents/AULAS/UFSC2022/Computa-o-Distribuida/lab2 - Atividade RMI$ javac *.java
Note: Server.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: Client.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
larissag.rosa@pop-os: ~/Documents/AULAS/UFSC2022/Computa-o-Distribuida/lab2 - Atividade RMI$ rmiregistry 2099 & java Server
[1] 37611
█
```

Depois, inicializamos o cliente em uma segunda aba. Na interação abaixo, adicionamos dois itens, removemos o primeiro item adicionado, pedimos o tamanho da lista e, por fim, saímos do sistema.

```

Saindo do programa
larissag.rosa@pop-os:~/Documents/AULAS/UFSC2022/Computa-o-Distribuida/lab2 - Atividade RMI$ java Cli
ent
Entre com um dos comandos a seguir:
    add <key> <valor>
    get <key>
    remove <key>
    size
    sair
size
result: 1
get 1
result: primeiro
add 2 segundo
Adicionou item
size
result: 2
remove 1
result: primeiro
size
result: 1
sair
larissag.rosa@pop-os:~/Documents/AULAS/UFSC2022/Computa-o-Distribuida/lab2 - Atividade RMI$

```

Análise de problemas de concorrência:

O controle de concorrência é feito na classe `ActivityList`, através de um `Lock` sobre todo acesso à lista, seja inserir, pegar um item ou remover um item é protegido pelo `Lock`. Portanto este controle impede apenas que 2 clientes realizem as operações sobre a lista ao mesmo tempo.

Um problema desta implementação é a falta de controle da informação da lista após um cliente obtê-la, ou seja, caso um cliente 1 leia um item realiza alguma operação sobre ele, e depois insere o item alterado, caso um outro cliente 2 tivesse inserido um novo item na mesma posição que o item em que o cliente 1 estava manipulando, este item inserido pelo cliente 2, seria perdido após o cliente 1 inserir o seu item novamente.
