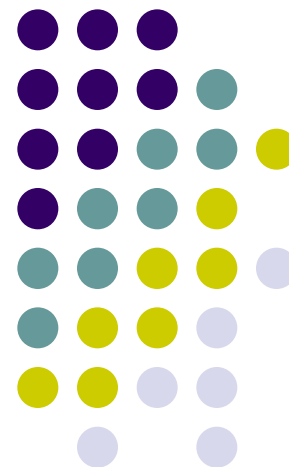
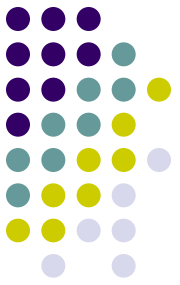


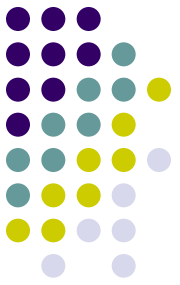
Programação Paralela e Distribuída

Prof. Cidcley
cidcley@ifce.edu.br





Programação com Espaço de Tuplas - JavaSpaces



Espaço de Tuplas

- Conceito introduzido em 1985 para ser utilizado em programação paralela
- Um espaço de tuplas é externo aos programas que realizam o processamento
- Processos se comunicam entre si indiretamente manipulando "tuplas" no espaço de tuplas
- Processos podem ler, escrever e remover tuplas do espaço de tuplas
- Tuplas são imutáveis, assim não podem ser modificadas mas sim renovadas no espaço

Tuplas



- Uma tupla é uma série de dados tipados:
- $\langle 6352.8, \text{"oi"}, 88 \rangle$
- $\langle 0, 50 \rangle$
- Tupas são endereçadas através de casamento de padrões com seus conteúdos
- $\langle \text{String}, 1234 \rangle$ deve casar com
- $\langle \text{"alguma string"}, 1234 \rangle$ ou
- $\langle \text{"outras string"}, 1234 \rangle$

Exemplo de Espaço de Tuplas

write(<"População",
"Brasil", 201000000>)



<"Capital", "Brasil", "Brasilia">

<"Capital", "Argentina", "Buenos Aires">

<"Capital", "USA", "Washington">

<"Capital", "Canadá", "Ottawa">

<"População", "Argentina", 410000000>

<"População", "USA", 314000000>

Exemplo de Espaço de Tuplas

`write(<"População",
"Brasil", 201000000>)`

`<"Capital", "Brasil", "Brasilia">`
`<"Capital", "Argentina", "Buenos Aires">`
`<"Capital", "USA", "Washington">`
`<"Capital", "Canadá", "Ottawa">`
`<"População", "Argentina", 410000000>`
`<"População", "USA", 314000000>`
`<"População", "Brasil", 201000000>`

Exemplo de Espaço de Tuplas

read(<"População",
"Argentina", Integer>)

<"Capital", "Brasil", "Brasilia">
<"Capital", "Argentina", "Buenos Aires">
<"Capital", "USA", "Washington">
<"Capital", "Canadá", "Ottawa">
<"População", "Argentina", 410000000>
<"População", "USA", 314000000>
<"População", "Brasil", 201000000>

Exemplo de Espaço de Tuplas

read(<"População",
"Argentina", Integer>)

<"População", "Argentina",
410000000>

<"Capital", "Brasil", "Brasilia">

<"Capital", "Argentina", "Buenos Aires">

<"Capital", "USA", "Washington">

<"Capital", "Canadá", "Ottawa">

<"População", "Argentina", 410000000>

<"População", "USA", 314000000>

<"População", "Brasil", 201000000>

Exemplo de Espaço de Tuplas

take(<"População", "USA",
Integer>)

<"Capital", "Brasil", "Brasilia">

<"Capital", "Argentina", "Buenos Aires">

<"Capital", "USA", "Washington">

<"Capital", "Canadá", "Ottawa">

<"População", "Argentina", 410000000>

<"População", "USA", 314000000>

<"População", "Brasil", 201000000>

Exemplo de Espaço de Tuplas

take(<"População", "USA",
Integer>)

<"População", "USA",
314000000>

<"Capital", "Brasil", "Brasilia">

<"Capital", "Argentina", "Buenos Aires">

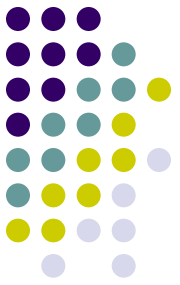
<"Capital", "USA", "Washington">

<"Capital", "Canadá", "Ottawa">

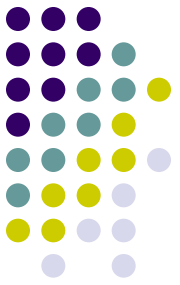
<"População", "Argentina", 410000000>

<"População", "Brasil", 201000000>

Linguagem LINDA



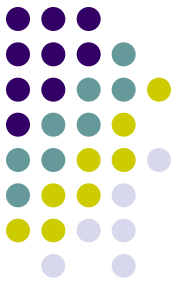
- Desenvolvida pela Universidade de Yale
- Linda é uma Linguagem de Coordenação
 - Separa comunicação de computação
- Define um modelo de comunicação entre processos paralelos
- Utiliza um único e global espaço de tuplas



JavaSpaces

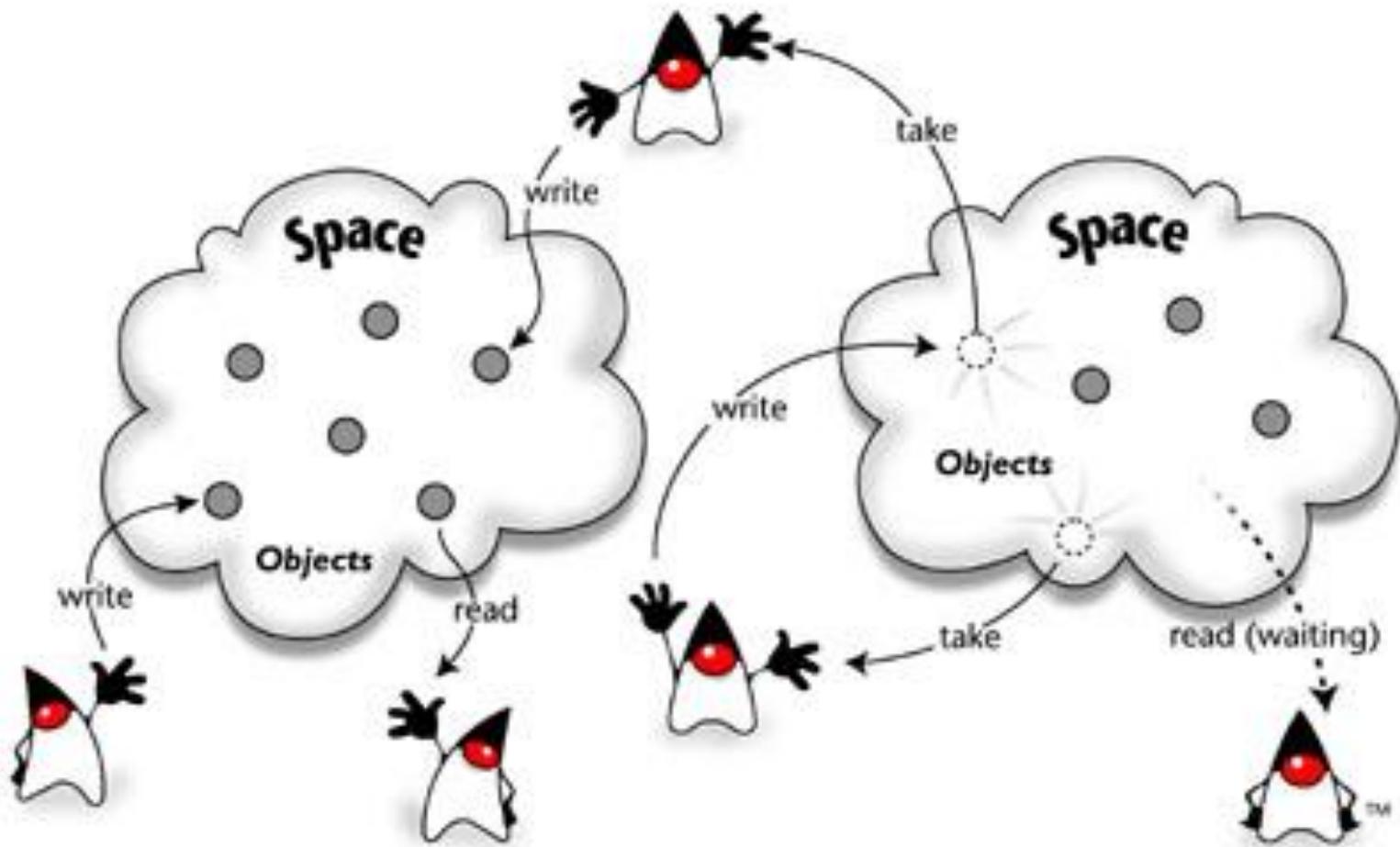
- **Implementação do conceito de Espaço de Tuplas em Java**
- **Existem outras implementações**
 - TSpaces
 - Gigaspaces
 - ...

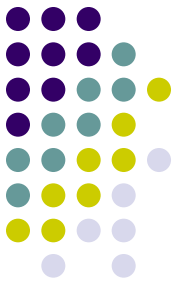
Space



- Um *space* é um lugar na rede para compartilhar e guardar objetos
- É a implementação da especificação do JavaSpace
- Pode ser visto como compartilhamento de memória em uma rede

Exemplo de Spaces

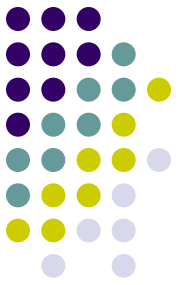




Propósito do JavaSpace

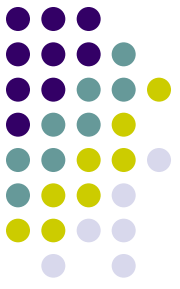
- Persistência Distribuída
- Outra forma de construir algoritmos distribuídos
 - Ao invés de invocação de métodos, fluxo de objetos,...
- É projetado para dar suporte a aplicações que trabalham com fluxo de objetos entre servidores

net.jini.space.JavaSpace



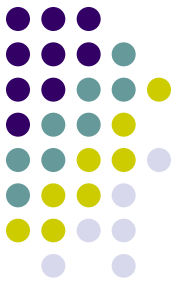
- Interface do JavaSpace
- Métodos da Interface
 - read, readIfExists
 - take, takeIfExists
 - write
 - notify

Entry



- Todos os objetos do *Space* são *entries*.
- Uma Entry é um grupo de objetos tipados expressos em uma classe que implementa a interface net.jini.core.entry.Entry.
- Campos do tipo Entry devem ser *public*.
- Classes do tipo Entry devem possuir um construtor *public* sem argumentos.
- Todos os atributos de uma Entry devem ser objetos.

Exemplo de Entry



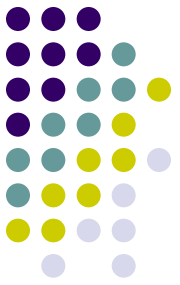
```
import net.jini.core.entry.*;

public class Converter implements Entry {
    public Float real;
    public Float dolar;
    public Boolean done;

    public Converter() {}

    public Converter(float r, boolean d) {
        real = new Float(r);
        done = new Boolean(d);
    }
}
```

Read

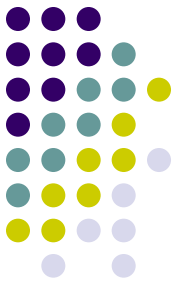


```
public Entry read(Entry tmpl, Transaction txn,  
    long timeout)
```

```
public Entry readIfExists(Entry tmpl, Transaction  
    txn, long timeout)
```

Obs.: timeout é o tempo, em milisegundos, que vai ficar tentando ler uma tupla do espaço.
Long.MAX_VALUE indica um tempo indeterminado.

Take

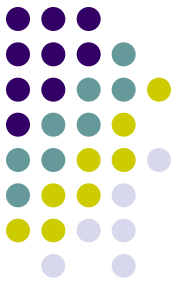


- Mesma operação do read, mas remove Entry do *Space*

```
public Entry take(Entry tmpl, Transaction txn,  
long timeout)
```

```
public Entry takelfExists(Entry tmpl, Transaction  
txn, long timeout)
```

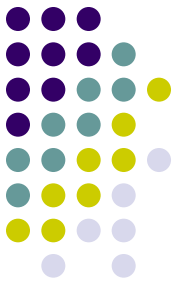
Write



```
public lease write(Entry tmpl, Transaction txn, long  
    lease)
```

Obs.: timeout é o tempo, em milisegundos, em que uma entry vai permanecer no espaço.
Lease.FOREVER pode ser usado para indicar que a tupla vai ficar por um tempo indeterminado.

Notify



- ✌ Método para registrar listeners que serão informados das mudanças no *space*

```
public EventRegistration notify(Entry tmp1,  
    Transaction txn, RemoteEventListener,  
    long lease, MarshalledObject handback)
```

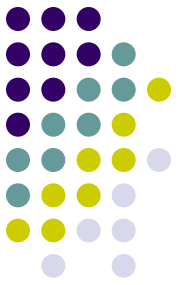
Observação: Transações



- ✌ Transações podem ser utilizadas para a realização de operações no espaço de tuplas
- ✌ Operações que utilizam transações são efetivas no espaço somente depois de “comitadas”

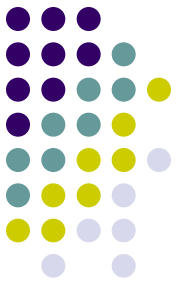
```
Transaction txn = ...;  
txn.commit;  
txn.abort;
```

Relacionamento com Jini



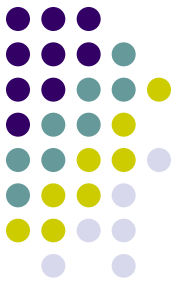
- É um serviço da tecnologia Jini
- Uso do modelo de programação do Jini
 - Transações
 - Leasing
 - Eventos

Pacotes do Jini



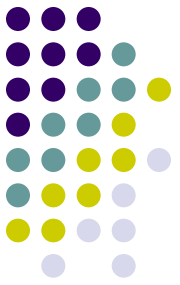
- `net.jini.core.transaction`
- `net.jini.core.entry`
- `net.jini.core.lease`
- `net.jini.core.event`

Implementação



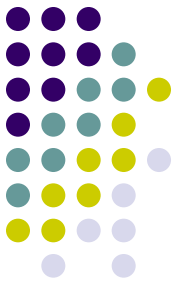
- Utilizar o Apache River (implementação Open Source do Jini)
- Aplicação Exemplo:
 - Escrever uma Mensagem no space
 - Ler uma Mensagem do space

Implementação



- Arquivos Necessários
 - Implementação da Entry
 - Implementação do Writer
 - Implementação do Reader

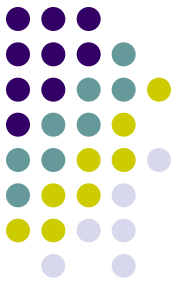
Implementação da Entry



```
import net.jini.core.entry.Entry;

public class Message implements Entry {
    public String content;
    public Message() {
    }
}
```

Implementação do Writer



```
import net.jini.space.JavaSpace;

import java.util.Scanner;

public class WriteMessage {

    public static void main(String[] args) {

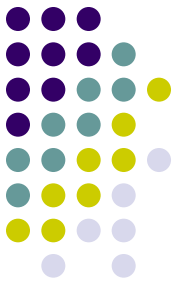
        try {

            System.out.println("Procurando pelo servico
JavaSpace...");

            Lookup finder = new Lookup(JavaSpace.class);

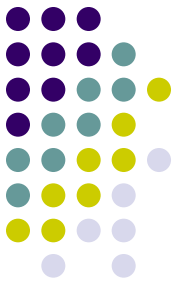
            JavaSpace space = (JavaSpace) finder.getService();
```

Implementação do Writer



```
if (space == null) {  
    System.out.println("O servico JavaSpace nao foi  
    encontrado. Encerrando...");  
    System.exit(-1);  
}  
  
System.out.println("O servico JavaSpace foi  
encontrado.");
```

Implementação do Writer



```
Scanner scanner = new Scanner(System.in);
```

```
while (true) {
```

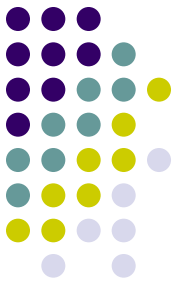
```
    System.out.print("Entre com o texto da mensagem  
(ENTER para sair): ");
```

```
    String message = scanner.nextLine();
```

```
    if (message == null || message.equals("")) {
```

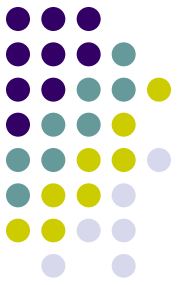
```
        System.exit(0);
```

```
}
```



Implementação do Writer

```
        Message msg = new Message();  
        msg.content = message;  
        space.write(msg, null, 60 * 1000);  
    }  
} catch (Exception e) {  
    e.printStackTrace();  
}  
}  
}
```

Implementação do Reader

```
import net.jini.space.JavaSpace;

public class ReadMessage {

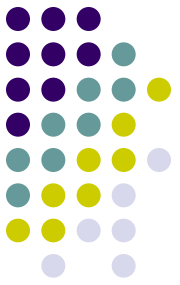
    public static void main(String[] args) {

        try {

            System.out.println("Procurando pelo servico
JavaSpace...");

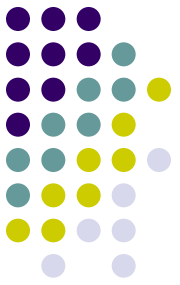
            Lookup finder = new Lookup(JavaSpace.class);

            JavaSpace space = (JavaSpace) finder.getService();
```



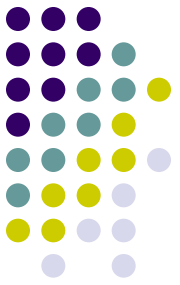
Implementação do Reader

```
if (space == null) {  
    System.out.println("O servico JavaSpace nao foi  
    encontrado. Encerrando...");  
    System.exit(-1);  
}  
  
System.out.println("O servico JavaSpace foi  
encontrado.");
```



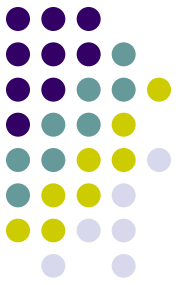
Implementação do Reader

```
while (true) {  
    Message template = new Message();  
  
    Message msg = (Message) space.take(template, null,  
60 * 1000);  
  
    if (msg == null) {  
        System.out.println("Tempo de espera esgotado.  
Encerrando...");  
        System.exit(0);  
    }  
}
```



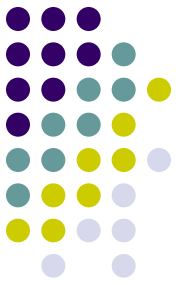
Implementação do Reader

```
        System.out.println("Mensagem recebida: "+
msg.content);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}
```



Executando Apache River

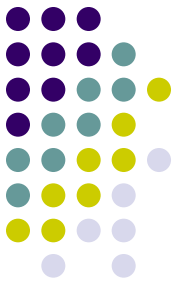
- ✌ Executar scripts e inicialização dos serviços (diretório examples/space/scripts)
- ✌ Executar o servidor web
 - *httpd.bat ou sh httpd.sh*
- ✌ Executando serviço de localização
 - *jrmf-reggie.bat ou sh jrmf-reggie.sh*



Executando Apache River

- ✌ Executando JavaSpace
 - jrmp-outrigger-group.bat ou jrmp-sh
outrigger-group.sd
- ✌ Para iniciar inicializar todos os scripts de uma vez (diretório examples/space)
 - start-services.bat ou start-services.bat

Compilação e Execução das Aplicações



✌️ Compilar arquivos

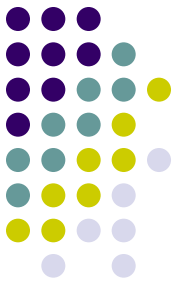
- `compile-app` (diretório `examples/space`)

✌️ Executar Writer

- `start-writer`(diretório `examples/space`)

✌️ Executar Reader

- `start-reader`(diretório `examples/space`)

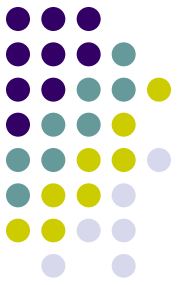


Referências

[http://www.artima.com/javaseminars/modules/J
avaSpaces/](http://www.artima.com/javaseminars/modules/JavaSpaces/)

<http://www.artima.com/jini/jiniology/js1P.html>

<http://river.apache.org/>



Fim