

# JASPER

---

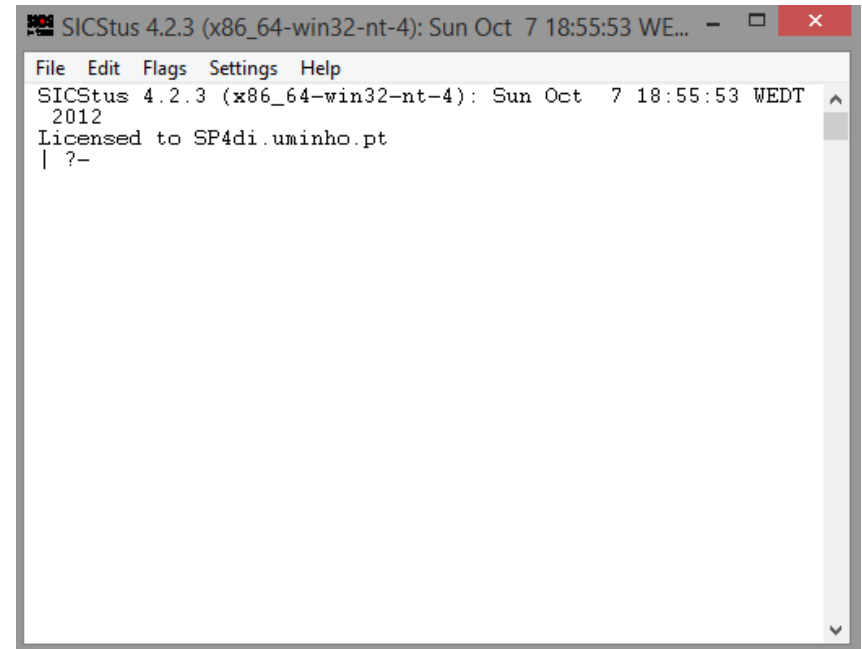
Java API para Prolog

# Índice

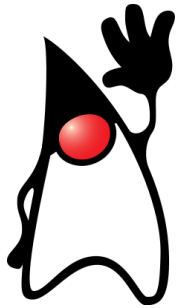
- Prolog
- Java
- Prolog VS Java
- Prolog em Aplicações Java
  - Instalação Jasper
  - Termos (Jasper)
  - Predicados (Jasper)
  - Queries (Jasper)
- Java em Aplicações Prolog
- Conclusão
- Questões

# Prolog VS Java

- Prolog
  - Prolog é uma linguagem declarativa;
  - Tem raízes na lógica de primeira ordem e lógica formal;
  - É expressada em termos e relações, representados por factos e regras.



# Prolog VS Java



- Java:
  - Java é uma linguagem de programação Orientada a Objectos
  - Tem raízes em C e C++
  - É interpretado na Java Virtual Machine

# Prolog VS JAVA

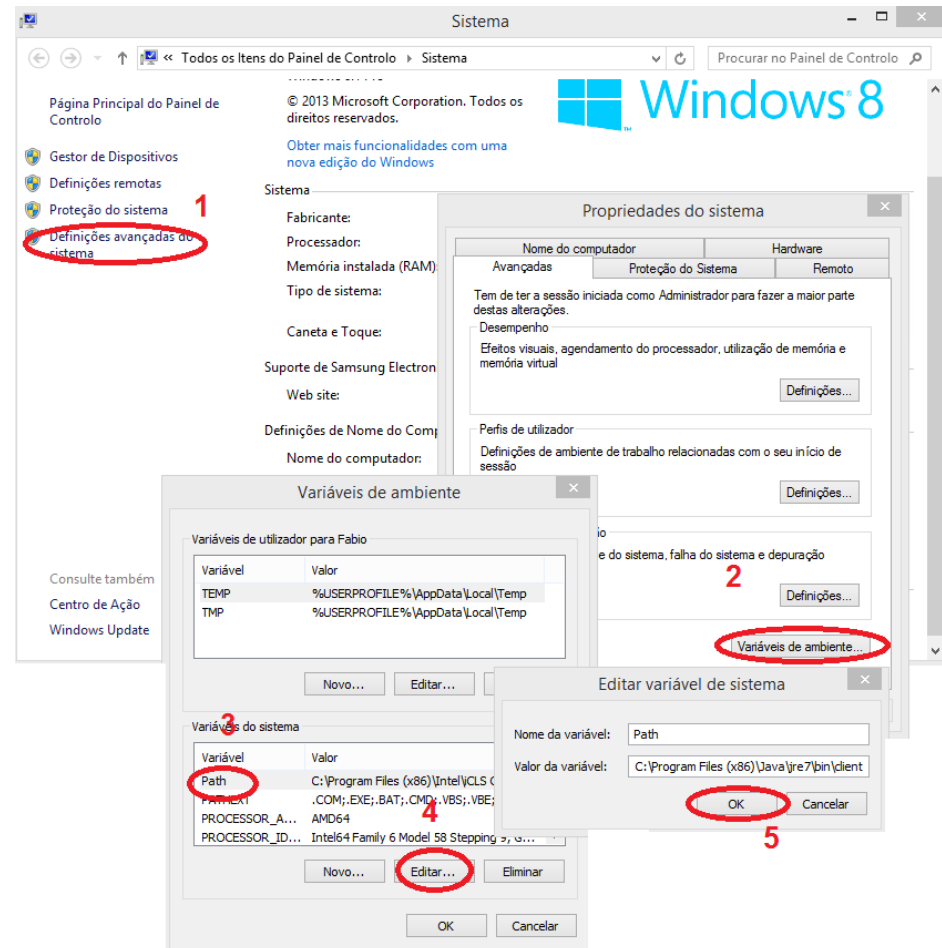
- Prolog
  - Programas sucintos;
  - Pode-se raciocinar sobre programas como objectos algebraicos;
  - É fácil ler e escrever programas com estruturas de dados;
  - Melhor para responder a problemas lógicos.
- Java
  - Código de programas complexos, tende a ser maior;
  - É difícil raciocinar sobre objectos java;
  - É difícil representar conhecimento de acordo com os princípios de lógica estendida.

# Prolog em Aplicações Java

- Q: Pode-se usar Prolog em Java?
- R: Sim, com as bibliotecas Jasper ou PrologBeans.

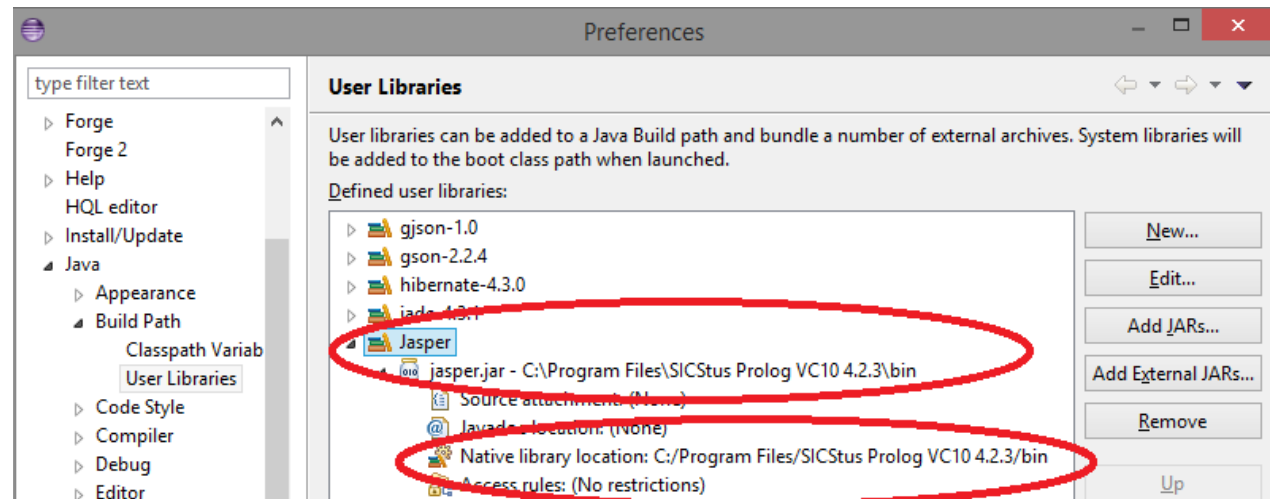
# Prolog em Aplicações Java

- Adicionar SICStus Prolog e Java ao %PATH%
- Java:
  - 32 bits
    - “C:\Program Files (x86)\Java\jre7\bin\client”
  - 64 bits
    - “C:\Program Files\Java\jre7\bin\server”
- SICStus:
  - “C:\Program Files\SICStus Prolog VC10 4.2.3\bin”



# Prolog em Aplicações Java

- Instalar Jasper em Eclipse
  - Adicionar Jasper Library às User Libraries
  - Adicionar path to SICStus à “Native library location:”





# Prolog em Aplicações Java

- Iniciar Prolog em Java

```
//Java Object to Interact with SICStus virtual Machine
SICStus sp;

//Initialize SICStus virtual machine
public void loadSICStus() throws SPEException {
    sp = new SICStus();
}

//Load SICStus script
public void loadSICStusScrpT(String pathToFile) throws SPEException
{
    sp.load(pathToFile);
}
```

# Prolog em Aplicações Java

- Termos

Prolog

```
predicate('x').
```

```
Predicate(X).
```

Java

```
String termString = "x";  
SICStus sp;  
  
//Termo Definido  
SPTerm term = new SPTerm(sp, termString);  
  
//Termo Indefinido (variável)  
SPTerm term = new SPTerm(sp).putVariable();
```

# Prolog em Aplicações Java

- Predicados

Prolog

```
predicate(_).
```

Java

```
String predicateName = "predicate";  
Int numTerms;  
SICStus sp;  
  
SPPredicate predicate = new SPPredicate (sp,  
predicateName, numTerms, "");
```

# Prolog em Aplicações Java

- Queries – 1ª Versão

Prolog

```
predicate('term',X).
```

Java

```
SPTerm [] terms = {  
    new SPTerm("term"),  
    new SPTerm().putVariable()  
};  
  
SICStus sp;  
  
SPQuery query = new SPPredicate (sp, terms,  
numTerms, "");  
query.close();
```

# Prolog em Aplicações Java

- Queries – 2ª Versão (RECOMENDADA)

Prolog

```
predicate('term',X).
```

Java

```
String queryS = "predicate('term',X).";  
SICStus sp;  
  
HashMap map = new HashMap();  
Query query = sp.openPrologQuery(queryS,map);  
while (query.nextSolution()) {  
    System.out.println(map.toString());  
}  
query.close();
```

# Prolog em Aplicações Java

- Queries – 3ª Versão

Prolog

```
predicate('term','term2').
```

Java

```
String query = "predicate('term','term2').";  
SICStus sp;
```

```
HashMap map = new HashMap();  
boolean b = sp.query(query,map);
```

# Prolog em Aplicações Java

- Questões – 4ª Versão

Prolog

```
predicate('term',X).
```

Java

```
String queryS = "predicate('term',X).";  
SICStus sp;
```

```
HashMap map = new HashMap();  
sp.queryCutFail(queryS,map);
```

# Prolog em Aplicações Java

- Questões
  - Problema
    - Formato das Strings de output
    - Requer parsing das repostas
  - Possível Resposta
    - Usar java regex;
    - Usar manipulação de strings.



# Prolog em Aplicações Java

- Caso de Estudo
  - Iniciar Prolog em Java com solução para a ficha01

Solução:

```
SICStus sp;  
  
Public void loadScript() throws SPEException {  
    sp = new SICStus(argv, null);  
    sp.load("ficha01.pl");  
}
```

# Prolog em Aplicações Java

- Caso de Estudo
  - Com o módulo de Prolog da ficha01, recriar as questões seguintes em Java utilizando Jasper:
    - pai(jose,X).
    - descendente(jose,X).
    - avo(X,joão).

# Prolog em Aplicações Java

- Caso de Estudo
  - Com o módulo de Prolog da ficha03, recriar as questões seguintes em Java utilizando Jasper:
    - `pertence(1,[1,2,3]).`
    - `pertence(X,[1,2,3]).`

# Java em Apilcações Prolog

- Q: E se for preciso usar recursos de java em Prolog?
- A: Pode-se usar Jasper para importar métodos Java para Prolog.

# Java em Aplicações Prolog

- Vamos criar uma classe exemplo em JAVA

```
public class Simple{  
  
    public int simpleMethod(int x){  
        return x+1;  
    }  
  
}
```

- Compilar
  - javac Simple.java

# Java em Aplicações Prolog

- Importing the Java class into a Prolog module

```
:- module(simple, [simple/2]).  
  
:- use_module(library(jasper)).  
:- load_foreign_resource(simple).  
  
foreign(method('Simple', 'simpleMethod', [static]), java,  
         simple(+integer,[-integer])).  
  
foreign_resource(simple,[ method('Simple', 'simpleMethod', [static]) ]).
```

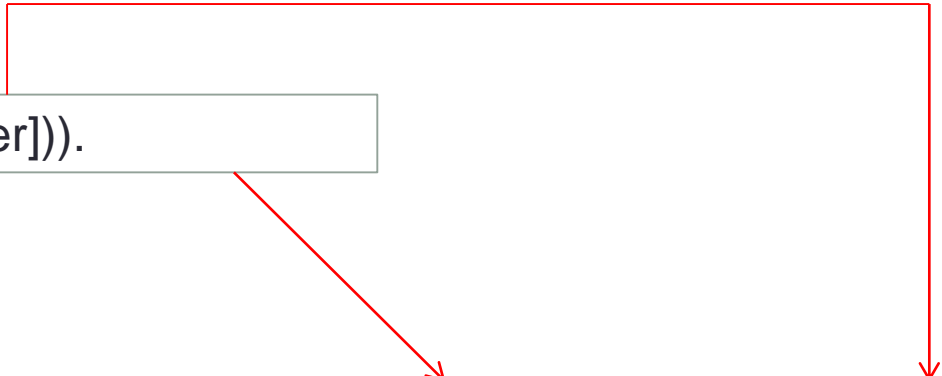
- Criar recurso para Prolog
  - splfr simple simple.pl

# Java em Apilcações Prolog

- Métodos Java em Prolog.

```
simple(+integer,[-integer])).
```

```
public int simpleMethod(int x){  
    return x+1;  
}
```



# Java em Aplicações Prolog

- Carregar o módulo de Prolog e usar os métodos da classe em Java como predicados em Prolog.

```
increment(1,X).
```



# Java em Apilcações Prolog

- Caso de Estudo
  - Criar classes em Java com métodos para user em Prolog
    - Ex:
      - Importar o gerador de números de Java e usa-lo para testar se um número gerado está presente numa lista de números usando o predicado pertence da ficha03.

# Conclusões

- É possível combinar Java e Prolog;
- Jasper é o middleware responsável por permitir o uso de SICStus prolog com e em Java;
- Existem alternativas para a interoperabilidade com outras linguagens de programação, contudo Jasper é recomendado para o caso de Java.

# Referências

- Exemplos SICStus Jasper:
  - [https://sicstus.sics.se/sicstus/docs/latest4/html/sicstus.html/lib\\_002djasper.html#lib\\_002djasper](https://sicstus.sics.se/sicstus/docs/latest4/html/sicstus.html/lib_002djasper.html#lib_002djasper)
  - [http://sicstus.sics.se/sicstus/docs/3.7.1/html/sicstus\\_12.html](http://sicstus.sics.se/sicstus/docs/3.7.1/html/sicstus_12.html)
  - [http://sicstus.sics.se/sicstus/docs/latest/html/sicstus/lib\\_002djasper.html#lib\\_002djasper](http://sicstus.sics.se/sicstus/docs/latest/html/sicstus/lib_002djasper.html#lib_002djasper)
- Manual SICStus
  - <http://www.informatik.uni-hamburg.de/RZ/software/sprachen/sicstus4/relnotes.pdf>

# Questões

- Email: [fabiosilva@di.uminho.pt](mailto:fabiosilva@di.uminho.pt)

