DCC019 - Linguagens de Programação
Departamento de Ciência da Computação — DCC - UFJF
Professor: Leonardo Vieira dos Santos Reis — e-mail:lvsreis@ice.ufjf.br
3º Trabalho Prático

1 Instruções Importantes

Nessa seção são apresentadas diversas informações relevantes referentes a entrega do trabalho e orientações a serem seguidas durante a implementação do mesmo. Leia atentamente antes de começá-lo.

1.1 Equipe de Desenvolvimento

O trabalho será desenvolvido individualmente ou em dupla.

1.2 Linguagem de Programação

O trabalho deverá ser desenvolvido na linguagem Racket.

1.3 Artefatos a Serem Entregues

Os artefatos a serem entregues são:

- código fonte do interpretador contido no arquivo *interpreter.rkt* e módulos adicionais que por venham a ser criados;
- documentação do trabalho em formato pdf.

Antes de enviar seu trabalho para avaliação, assegure-se que:

1. seu código executa no REPL de Racket. Programas com erros de sintaxe receberão nota zero;
2. todos os fontes a serem enviados têm, em comentário no início do arquivo, nome e matrícula do autor do trabalho;
3. arquivo de documentação tenha a identificação do autor do trabalho;

1.4 Critérios de Avaliação

A avaliação será feita mediante análise do código fonte, documentação e apresentação do trabalho (entrevista). Os seguintes fatores serão observados na avaliação do código fonte: corretude do programa, estrutura do código, redigibilidade e legibilidade. A corretude se refere à implementação correta de todas as funcionalidades especificadas, i.e., se o programa desenvolvido está funcionando corretamente e não apresenta erros. Os demais fatores avaliados no código fonte são referentes a organização e escrita do trabalho.

A documentação do trabalho deverá relatar as decisões de projeto e estratégia intelectual para solução do trabalho. Espera-se que na documentação tenha informações a respeito das estruturas de dados utilizadas e uma descrição em alto nível da solução. Recomenda-se a utilização de exemplos para ilustrar e/ou detalhar a abordagem utilizada para resolver o problema.

O trabalho deverá ser apresentado ao professor da disciplina e, só será avaliado após a realização da entrevista, i.e., trabalhos que não forem apresentados não terão nota. Na entrevista, o discente deverá elucidar, ao menos, como modelou e resolveu o problema. A entrevista também tem a finalidade de avaliar a confiabilidade e segurança do autor do código em explicar pontos relevantes do trabalho desenvolvido.

Assim, a entrevista influenciará na avaliação dos artefatos entregues. Portanto, a nota final será dada a partir da avaliação do conjunto do código fonte, documentação e entrevista. **É de responsabilidade do discente solicitar a marcação do dia e horário da entrevista com o professor da disciplina**.

Atrasos serão penalizados por uma função exponencial de dias de atraso, i.e., será reduzido da nota um percentual referente a exponencial na base 2 dos dias de atraso. A tabela a seguir mostra a nota em função dos dias de atraso:

Observe que a partir do 7º dia de atraso seu trabalho não será mais avaliado.

Dias de Atraso	Nota
1	n*0.98
2	n*0.96
3	n*0.92
4	n*0.84
5	n*0.68
6	n*0.36
7	0

Program	→	{ClassDecl}* Expression
ClassDecl	→	**class** Identifier **extends** Identifier {**field** Identifier}* {MethodDecl}*
MethodDecl	→	**method** Identifier ({Identifier}*(,)) Expression
Expression	→	**new** Identifier (**(**{Expression}*(,) **)**
	|	**send** Expression Identifier (**(**{Expression}*(,) **)**
	|	**super** Identifier (**(**{Expression}*(,) **)**
	|	**self**
	|	Number
	|	**-(**Expression **,** Expression**)**
	|	**zero? (** Expression **)**
	|	**if** Expression **then** Expression **else** Expression
	|	Identifier
	|	**let** Identifier = Expression **in** Expression
	|	**proc (** Identifier **)** Expression
	|	**(** Expression **,** Expression **)**
	|	**letrec** Identifier **(** Identifier **)** = Expression **in** Expression
	|	**set** Identifier = Expression
	|	**begin** {Expression}*(;) **end**

Figura 1: Sintaxe da linguagem CLASSES

2 A Linguagem CLASSES

CLASSES é uma linguagem orientada a objetos. Um programa na linguagem consiste de uma sequência de declarações de classes seguida por uma expressão a ser executada, a qual poderá fazer uso das classes declaradas. Uma declaração de classe tem um nome imediatamente seguido pelo nome da superclasse, contém zero ou mais declarações de campos e zero ou mais definições de métodos. Uma declaração de um método contém um nome, uma lista de parâmetros formais e um corpo.

A linguagem apresenta quatro expressões para manipular classes e objetos. A expressão **new** cria um objeto da referida classe, a partir da execução do método **initialize** da classe com os respectivos argumentos da expressão. A expressão **self** retorna o objeto no qual o método corrente está operando. A expressão **send** consiste de uma expressão que deve ser avaliado para um objeto, um método e uma sequência de zero ou mais operandos. O método a ser executado é obtido da classe do objeto e, então, é executado com os argumentos referentes a avaliação dos operandos. Por fim, uma expressão **super** tem o efeito de executar um método da hierarquia de classe do objeto corrente, buscando o método em questão a partir da superclasse do objeto.

A gramática da Figura 1 especifica a sintaxe concreta da linguagem, no formato EBNF, destacando em negrito os símbolos terminais. Ressalta-se que a notação {A}* denota uma sequência do elemento A e a notação {A}*(,) denota uma sequência de elementos A separados por vírgula.

```
class c1 extends object
  method initialize () 1
  method m1 () send self m2()
  method m2 () 13
class c2 extends c1
  method m1 () 22
  method m2 () 23
  method m3 () super m1 ()
class c3 extends c2
  method m1 () 32
  method m2 () 33
let o3 = new c3 ()
in send o3 m3()
```

Figura 2: Exemplo de programa na linguagem CLASSES

Um exemplo de programa na linguagem é apresentado na Figura 2.

3 Especificação Técnica do Trabalho

Com base no conteúdo estudado na disciplina, nesse trabalho pede-se para implementar a linguagem CLASSES usando Racket. Para realizar a implementa deste trabalho, você deverá usar o código disponível no github da disciplina [1]. O código disponibilizado traduz a sintaxe concreta da linguagem para uma representação abstrata (*AST*), descrita no arquivo *ast.rkt*. Esta *AST* será a interface que vocês deverão usar para realizar a implementação do interpretador para a linguagem CLASSES. Para implementar o interpretador, você deverá modificar as funções *value-of-program* e *value-of*, do arquivo *interpreter.rkt*, e implementar as funcionalidades desejadas para a correta execução de um programa na linguagem CLASSES.

Na pasta *examples* há um conjunto de programas em CLASSES que podem ser utilizados para testar a sua implementação. O único aquivo que deve, e pode, ser alterado é o *interpreter.rkt*. Os demais arquivos não devem sofrer alterações e não precisam ser entregues. Caso julgar necessário, você pode criar outros arquivos.

4 Entrega do Trabalho

A data da entrega do trabalho será até o dia **15 de janeiro de 2023**, via plataforma do GoogleClassroom. A entrevista deverá ser realizada juntamente com o professor da disciplina até o dia **20 de janeiro de 2023.**

[1] https://github.com/lvsreis/dcc019/classes