DCC019

Linguagem de Programação

Relatório
Trabalho Prático 2

Aluno: Rômulo Luiz Araujo Souza Soares - 201665219C

1 - Introdução

Este relatório é referente ao trabalho prático 2 da disciplina de Linguagem de Programação, que descreve a implementação de uma linguagem OOP (ICLASSES) utilizando de uma linguagem de programação funcional em Racket, focando na execução de expressões e instruções. O código fornecido define um interpretador para uma linguagem que suporta expressões aritméticas, lógicas, controle de fluxo, declarações locais, e orientação a objetos com herança. O objetivo é fornecer uma visão geral da implementação, destacando os principais componentes e conceitos envolvidos.

2 - Como Executar

É necessário baixar o código presente no repositório dcc019 fornecido pelo professor, presente no link (https://github.com/lvsreis/dcc019). Depois de copiar o arquivo interpreter.rkt para a pasta "*/dcc019/exercise/iclasses/"*, você pode executar os exemplos presentes na pasta "*dcc019/exercise/iclasses/examples/"* para executar e testar o código.

3 - Descrição da implementação

Para a realização do código foi utilizado de base o código disponibilizado pelo GitHub e para realizar a implementação das funções necessárias para a execução da linguagem ICLASSES foi utilizando as informações e pseudo códigos fornecidos no livro Essentials of Programming Languages, em específico o capítulo 9.

Funções:

- value-of
    - **Objetivo**: Avaliar o valor de uma expressão.
    - **Parâmetros**: A expressão a ser avaliada (exp) e o ambiente atual (Δ).
    - **Funcionalidade**: Percorre a expressão e realiza as operações correspondentes de acordo com as regras da linguagem, como operações aritméticas, lógicas, condicionais, etc.
- result-of
    - **Objetivo**: Executar uma instrução.
    - **Parâmetros**: A instrução a ser executada (stmt) e o ambiente atual (Δ).
    - **Funcionalidade**: Realiza a execução de diferentes tipos de instruções, como atribuições, impressões, blocos, estruturas condicionais e loops.
- value-of-program
    - **Objetivo**: Executar o programa principal.
    - **Parâmetros**: O programa principal, contendo declarações de classes e um bloco de código (prog).
    - **Funcionalidade**: Inicializa o ambiente de classes com as informações das classes declaradas e executa as instruções do bloco principal.
- make-self

    - **Objetivo**: Retorna a referência ao objeto atual (self).
    - **Parâmetros**: O ambiente atual (env).
    - **Funcionalidade**: Obtém a referência ao objeto atual no ambiente.
- make-send
    - **Objetivo**: Realiza uma chamada de método em um objeto.
    - **Parâmetros**: O objeto, o nome do método, os argumentos e o ambiente atual.
    - **Funcionalidade**: Aplica dinamicamente o método correspondente ao objeto, passando os argumentos.
- make-super
    - **Objetivo**: Realiza uma chamada de método na superclasse do objeto atual.
    - **Parâmetros**: O nome do método, os argumentos e o ambiente atual.
    - **Funcionalidade**: Obtém a referência à superclasse no ambiente e aplica dinamicamente o método correspondente, passando os argumentos.
- make-new
    - **Objetivo**: Criar uma nova instância de uma classe.
    - **Parâmetros**: O nome da classe, os argumentos e o ambiente atual.
    - **Funcionalidade**: Cria um novo objeto com base na classe especificada, inicializando seus campos e aplicando o método de inicialização.
- new-object
    - **Objetivo**: Criar uma nova instância de um objeto.
    - **Parâmetros**: O nome da classe.
    - **Funcionalidade**: Inicializa um novo objeto com referências a campos não inicializados.
- apply-method
    - **Objetivo**: Aplicar um método a um objeto.
    - **Parâmetros**: O método, o objeto, os argumentos e o ambiente atual.
    - **Funcionalidade**: Executa o corpo do método no contexto do objeto, passando os argumentos.
- extend-envs
    - **Objetivo**: Estende o ambiente com múltiplas variáveis e valores.
    - **Parâmetros**: Lista de variáveis, lista de valores e o ambiente a ser estendido.
    - **Funcionalidade**: Adiciona cada par variável-valor ao ambiente.
- add-to-class-env!
    - **Objetivo**: Adicionar uma classe ao ambiente de classes.
    - **Parâmetros**: Nome da classe e a própria classe.
    - **Funcionalidade**: Atualiza o ambiente de classes com informações da nova classe.
- lookup-class
    - **Objetivo**: Busca informações de uma classe no ambiente de classes.
    - **Parâmetros**: Nome da classe.
    - **Funcionalidade**: Retorna a classe correspondente ao nome fornecido.
- initialize-class-env

    - **Objetivo**: Inicializa o ambiente de classes com base nas declarações de classes.
    - **Parâmetros**: Lista de declarações de classes.
    - **Funcionalidade**: Itera sobre as declarações de classes e adiciona cada classe ao ambiente de classes.
- initialize-class-decl
    - **Objetivo**: Inicializa uma classe específica no ambiente de classes.
    - **Parâmetros**: Declaração de classe.
    - **Funcionalidade**: Extrai informações da declaração de classe e adiciona a classe ao ambiente de classes.
- append-field-names
    - **Objetivo**: Concatena os nomes dos campos de uma classe com os novos campos.
    - **Parâmetros**: Nomes dos campos da superclasse e novos campos.
    - **Funcionalidade**: Garante que os campos da superclasse não duplicam os novos campos.
- find-method
    - **Objetivo**: Encontrar um método específico em uma classe.
    - **Parâmetros**: Nome da classe e nome do método.
    - **Funcionalidade**: Retorna o método correspondente ao nome fornecido na classe especificada.
- method-decls-method-env
    - **Objetivo**: Converte declarações de métodos para um ambiente de métodos.
    - **Parâmetros**: Declarações de métodos, nome da superclasse e nomes dos campos.
    - **Funcionalidade**: Converte cada declaração de método para um par nome-método no ambiente de métodos.
- merge-method-envs
    - **Objetivo**: Combina dois ambientes de métodos.
    - **Parâmetros**: Ambiente de métodos da superclasse e ambiente de métodos da classe atual.
    - **Funcionalidade**: Combina os ambientes de métodos, priorizando os métodos da classe atual.
- values-of-exps
    - **Objetivo**: Avalia valores de uma lista de expressões.
    - **Parâmetros**: Lista de expressões e ambiente atual.
    - **Funcionalidade**: Avalia cada expressão na lista e retorna a lista de valores correspondentes.