

MAC0316/5754
Exercício Programa 3
Data de entrega: 17/12/2020

Instruções:

1. Você deve entregar seu programa pelo PACA em um **único arquivo .rkt** contendo as definições do interpretador.
2. **Programas atrasados terão uma penalidade de 20% na pontuação POR DIA.**

Interpretador (10 pontos)

Nesta parte vamos implementar uma versão da linguagem **Smalltalk**. As grandes modificações serão na **linguagem de entrada** e no **processo de resolução de métodos**. Utilizaremos resolução *dinâmica* de métodos mas usaremos um padrão “híbrido” onde inteiros e suas operações são primitivos.

Sintaxe da linguagem (descrita como uma gramática):

```
<input>          --> <expression> | <class-def>

<class-def>      --> (class <class-id> <inst-var> <method-def> <method-def>)

<class-id>       --> um identificador (para nome da classe)

<inst-var>       --> um identificador (para nome da variável de instância)

<method-def>     --> (method <method-id> <method-param> <expression>)

<method-id>      --> um identificador (para nome do método)

<method-arg>     --> um identificador (para nome do parâmetro do método)

<expression>    --> <arith-exp> | <if-exp> | <seq-exp> | <set-exp> | <let-exp> |
                    <new-exp> | <send-exp> | <value>

<arith-exp>      --> (<arith-op> <expression> <expression>)

<arith-op>       --> + | - | * | /
```

```

<if-exp>      --> (if <expression> <expression> <expression>)

<seq-exp>     --> (seq <expression> <expression>)

<set-exp>     --> (set! <inst-var> <value>) | (set! <method-arg> <value>)

<let-exp>     --> (let <class-id> <value>)

<new-exp>     --> (new <class-id> <value>)

<send-exp>    --> (send <expression> <method-id> <value>)

<value>       --> <integer> | <object> | <class> | <method> | <object>

```

- a) **(2 pontos):** Acrescente o valor `classV` no interpretador. Implemente o comando `class`, **parte da linguagem**, que possui 4 parâmetros: nome da classe pai, variável de instância, definição de método 1, e definição de método 2. Note que o `class` devolve um `classV`, cujo nome poderá ser registrado no ambiente usando o comando `let`.
- b) **(2 pontos):** Substitua o valor `closV` pelo valor `methodV`, que é semelhante aos fechamentos mas possui um nome (para envio de mensagens) e **não guarda um ambiente**. Troque `lamc` e `lams` por `methodc` e `methods` para substituir o comando `lambda` por `method`. Note que `methodV` será um componente de `classV` quando o usuário definir uma classe. `methodV` terá acesso a um ambiente (que incluirá o atributo de instância da classe) apenas quando o método for chamado.
- c) **(2 pontos):** Acrescente o valor `objetcV` no interpretador e adicione uma classe pai-de-todos `Object` no ambiente global inicial. Ou seja, quando o usuário for rodar `interpS`, ele deve poder usar `Object` em seu código. Isso é necessário porque o comando `class` exige que toda classe feita pelo usuário tenha uma classe pai. Desse modo, todas as classes criadas herdarão de `Object`.
- d) **(2 pontos):** Implemente o operador primitivo `new` que recebe como parâmetro o **nome de uma classe** e um **valor**, e cria um `objetcV` daquela classe cujo atributo é inicializado com o valor. Lembre-se que a criação de um objeto da classe filha leva a criação de objetos das suas classes ancestrais (cujos atributos podem ser inicializados com o valor 0). Além disso, lembre que o objeto filho usará esses objetos para acessar atributos de instância e métodos das classes ancestrais.

Finalmente, adicione um caso especial quando a classe analisada for `Object`, que não terá classe pai.

- e) **(2 pontos)**: Substitua o operador primitivo `call` pelo `send`, que recebe como parâmetro um `objetcv`, um nome de método e um argumento e que executa o método correspondente com o argumento. Lembre-se de que, se o objeto da classe filha não conhecer a mensagem, ele deve repassá-lo para a classe pai. Além disso, lembre que se a classe filha tiver um método de mesmo nome que um da classe pai, o método da classe filha deve ter preferência. Finalmente, adicione um caso especial quando a classe analisada for `Object`, lançando um erro de “mensagem desconhecida”.

Note que toda classe possui 2 métodos. Isso permitirá testar a redefinição de métodos da classe pai.

IMPORTANTE: Não será dado **crédito parcial** aos itens acima. Apenas aqueles que funcionarem receberão pontos.

IMPORTANTE: Você deve usar a versão `plai-typed` (**não** `plai`) do interpretador *`closureTyped.rkt`* (disponibilizado com este enunciado).