Universidade Federal da Fronteira Sul

Ciência da Computação

Programação I

Prof. Fernando Bevilacqua

Trabalho 1 (T1)

Data de entrega: 22/04/2015 (quarta-feira)

Descrição

O trabalho pode ser em **dupla** ou **trio**. Implementar um programa em Java que interprete uma linguagem de programação criada pelos alunos. O programa deve funcionar da seguinte forma pela linha de comando:

```
java NomePrograma arquivo.ext
```

sendo que o parâmetro `arquivo.ext` será o arquivo a ser interpretado pelo programa, que pode ter qualquer nome e qualquer extensão, a critério do aluno. O interpretador deve ser capaz de analisar e reagir corretamente às seguintes situações no arquivo fonte que ele esteja interpretando:

- Declaração de variáveis;
- Atribuição de valor a variável;
- Expressões com no mínimo dois operandos (ex.: a + b);
- Operações de adição, subtração, divisão e multiplicação;
- Laço;
- Comando de saída (ex.: mostrar algo na tela);
- Controlador de fluxo (ex.: if);
- Aninhamento de comandos (ex.: ifs dentro de ifs, laços dentro de laços).

Avaliação

Os seguintes elementos serão considerados durante a correção:

- Funcionamento correto (o programa precisa cumprir seu objetivo conforme a descrição do trabalho);
- Legibilidade do código (nomes de classes com a primeira letra maiúscula, métodos e propriedades no formado nomeFormaCamelo, **indentação correta**, etc). Trabalhos mal indentados sofrerão desconto considerável em sua nota;
- Utilização de forma satisfatória dos conceitos de orientação a objetos vistos em aula;
- Comentários (o código fonte deve conter um bloco de comentário no começo informando o propósito do programa e o nome/email do seu autor).
- O interpretador deve ser testado, no mínimo, com dois programas: um que calcula a média de dois números e outro que diz se um número é primo ou não.
- Histórico do controle de versão é condizente com o trabalho realizado (Ex. vários commits espalhados ao longo do tempo, não um único commit com diversos arquivos).
- Mostrar domínio sobre a implementação durante o seminário de apresetantação para a turma.

Observações

- Haverá um desconto de 50% da nota do trabalho por dia de atraso na entrega, com prazo máximo de 3 dias de atraso;
- Trabalhos copiados receberão nota zero e o nome dos envolvidos será levado ao colegiado do curso;
- Programas que não compilarem receberão nota zero instantânea (nenhuma avaliação será realizada).
- ***Esse trabalho será utilizado como base para trabalhos futuros.***

Universidade Federal da Fronteira Sul

Ciência da Computação

Programação I

Prof. Fernando Bevilacqua

Tabela de pontuação

Funcionalidade / critério	**Nota (dupla)**	**Nota (trio)**
Declaração de variáveis; Atribuição de valor a variável; Expressões com no mínimo dois operandos; Operações de adição, subtração, divisão e multiplicação; Laço; Comando de saída (ex.: mostrar algo na tela); Controlador de fluxo (ex.: if);	**0 a 6**	**0 a 3**
Aninhamento de comandos (ex.: ifs dentro de ifs, laços dentro de laços).	**6 a 8**	**3 a 5**
Comandos de entrada	**+ 0.5**	**5 a 6**
Sintaxe flexível (ex. não exigir número determinado de espaços entre comandos)	**+ 0.5**	**6 a 8**
Funções (com escopo)	**+10%**	**+5%**
Vetores	**+5%**	**0.5**
Utilização e demonstração de entendimento dos conceitos de orientação a objetos.	**8 a 10**	**8 a 10**

Forma de entrega

O trabalho deve ser enviado desenvolvido num sistema de controle de versão online, como Github. O repositório deve ter a seguinte estrutura (imagine que o nome da linguagem criada para esse interpretador chama-se "blah"):

```
blah
  |
  +-- fonte/
  +-- exemplos/
  +-- manual.pdf
  +-- blah.jar
```

O arquivo **`manual.pdf`** é um documento descrevendo a linguagem. Ele deve ter os comandos da linguagem, sua sintaxe, exemplos, etc, de forma que um programador consiga escrever um programa utilizando ***APENAS*** o manual como guia. O arquivo **`blah.jar`** é o interpretador empacotado no formato JAR (vide Capítulo 10 da apostila). A pasta **`fonte`** contém o código fonte do interpretador. A pasta **`exemplos`** contém programas de exemplo escritos na linguagem do interpretador.

Trabalhos que não seguirem esse formato receberão **nota zero**.

Forma de apresentação

O trabalho deve ser apresentado, em aula, em um seminário realizado para a turma inteira. Cada grupo terá direito a 15 min de apresentação, na qual deverão, obrigatoriamente, cobrir os seguintes tópicos:

- Breve apresentação da linguagem (nome, sintaxe, funcionalidades, repositório);
- Como cada linha do código-fonte é analisada (ex. como os elementos são “quebrados” em partes, como o interpretador sabe que uma linha é uma atribuição, etc.);
- Como funciona, em termos de implementação, o sistema de variáveis;
- Como funciona, em termos de implementação, o sistema de expressões;
- Como funciona, em termos de implementação, ifs e laços.

- Demonstrar o intepretador funcionando com, no mínimo, 2 programas. Durante a demonstração, alterações serão exigidas (ex.: “crie uma laço naquela posição”, “adicione uma variável naquela expressão”, etc).

A nota da apresentação será individual. Cada membro do grupo será questionado sobre partes do trabalho, tendo que responder sozinho à pergunta.

Dicas

O programa deve ser capaz de interpretar qualquer cenário possível dentro dos comandos que disponibiliza, ou seja, ele precisa interpretar qualquer código escrito com a linguagem criada, não apenas um programa em específico.

Para armazenar as variáveis, talvez seja interessante criar uma classe `Variavel` com atributos `nome` e `valor`. O programa principal terá um vetor de objetos `Variavel`, sendo que cada entrada nesse vetor corresponde a uma variável do programa sendo interpretado.

Universidade Federal da Fronteira Sul

Ciência da Computação

Programação I

Prof. Fernando Bevilacqua

Para garantir que seu interpretador funciona em diversos cenários, teste ele com os códigos abaixo (que foram escritos em C, você precisa converter para a sua linguagem):

Olá Mundo

```
#include <stdio.h>

void main() {
	printf("Ola mundo!");
}
```

Declarações e atribuições

```
#include <stdio.h>

void main() {
	int a;

	a = 2;
	printf("%d", a);
}
```

Laço

```
#include <stdio.h>

void main() {
	int i;

	i = 0;
	while(i < 10) {
		printf("%d", i);
		i++;
	}
}
```

Expressões

```
#include <stdio.h>

void main() {
	int a, b;

	a = 1 + 1;
	b = 3 + a;

	printf("%d", a);
	printf("%d", b);
}
```

Ifs

```
#include <stdio.h>

void main() {
	int a;
	a = 1;

	if(a > 1) {
		printf("Maior");
	} else {
		printf("Menor");
	}
}
```

Aninhamento

```
#include <stdio.h>

void main() {
	int i;
	int j;
	i = 0;
	j = i;

	while(i < 3) {
		if(i == 0) {
			printf("Zero");
		} else {
			printf("Outro");
		}
		i++;
	}

	if(i == 3) {
		if(j == 0) {
			printf("Ok!");
		} else {
			printf("Erro");
		}
	}
}
```