CEFET-MG	Centro Federal de Educação Tecnológica de Minas Gerais Engenharia da Computação - Departamento de Computação Disciplina: Laboratório de Linguagens de Programação

Professor: César Francisco de Moura Couto
Trabalho Prático II – 30 pontos Data de Entrega: 17/12/2013

1 Comentários iniciais sobre Java

1. Exemplo de declaração de subclasses:

```
class A {
     private int x;
     public A(int x1) {  // construtora da classe A
         x = x1;
     }
}

class B extends A { // B é subclasse de A
     private int y;
     public B(int x1, int y1) {  // construtora da classe B
         super(x1); // chama construtora da superclasse
         y = y1;
     }
}
```

2. A classe pré-definida `Object` é a raiz da hierarquia de classes em Java. Toda classe, direta ou indiretamente, estende `Object`. No exemplo acima, `A` é implicitamente uma subclasse de `Object`. E como `B` é uma subclasse de `A`, `B` é também descendente de `Object`. Assim, uma referência para `Object` pode se referir a objetos de quaisquer classes.

3. Existe em Java uma classe pré-definida, chamada `Vector`, que implementa uma lista encadeada. Para usar esta classe, deve-se importá-la na primeira linha do programa através do seguinte comando:

```
// o comando import de Java é similar ao #include de C++
import java.util.Vector;
```

Seja então o trecho de programa abaixo:

```
// declara e cria um Vector
Vector v = new Vector();

// adiciona ao vetor uma referência para objeto da classe A
v.add(new A(....));

// adiciona ao vetor uma referência para objeto da classe B
v.add(new B(....));

// Vector é uma lista de objetos e, portanto, pode armazenar referências
```

```
// para objetos de quaisquer classes.

// numero de elementos em v
int tam= v.size();
A a;
for (int i= 0; i < tam; i++) {
   a = (Object) v.elementAt(i);
// retorna uma referência para o i-ésimo elemento em v
// Veja que o casting é necessário para converter de Object para A e viabilizar
// a atribuição
          ..................
}
```

2 Enunciado

Implemente em Java um interpretador para a linguagem "tiny", que possui a seguinte gramática:

```
<programa>       ::=  <lista_comandos> "endp"
<lista-comandos> ::= <comando> | <comando> <lista-comandos>
<comando>        ::= <cmd-print> | <cmd-println> | <cmd-readInt> |
                     <cmd-atrib> | <cmd-while>
<cmd-print>      ::= "print" "(" <palavra> ")" | "print" "(" <variavel> ")"
<cmd-print>      ::= "println"
<cmd-readInt>    ::= "readInt" "(" <variavel> ")"
<cmd-atrib>      ::= <variavel> "="  <inteiro> |
                     <variavel> "=" <variavel> <op_arit> <variavel> |
                     <variavel> "=" <variavel> <op_arit> <inteiro>
<cmd-while>      ::= "while" "(" <variavel> <op_rel> <variavel> ")"
                         <lista_comandos>
                     "endw"
<variavel>       ::= "a" | "b" | ... | "a" <variavel>| "b"<variável> | ... |
<op-rel>         ::= "=" | ">" | "<"
<op_arit>        ::= "+" | "-" | "*" | "\"
<palavra>        ::= "\""<variavel>"\""
<inteiro>        ::= <digito><inteiro> | <digito>
<digito>         ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

O interpretador deverá possuir uma interface de linha de comando, lendo o programa de um arquivo texto. Maiores informações sobre a implementação do trabalho serão dadas em sala de aula.

Aula	Comandos a serem implementados	Valor (pts)
01	Comando print, println e readInt	7,5
02	Comando atrib	7,5
03	Comando while	15