JAVATOOL: UMA FERRAMENTA PARA ENSINO DE PROGRAMAÇÃO

Marcelle Pereira Mota[1,2], **Lis W. Kanashiro Pereira**[1,2], **Eloi Luiz Favero**[1,2]

[1]Laboratório de Ensino a Distância (LabEAD)
Universidade Federal do Pará (UFPA)
Belém – PA – Brasil

[2]Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal do Pará (UFPA)
Belém – PA – Brasil

{cellemota, liskanashiro}@gmail.com, favero@ufpa.br

Abstract. *This paper describes a tool called JavaTool, whose objective is to facilitate the programming learning in introductory disciplines of computer science courses. The proposal is based on the development of an interpreter for Java language, to make the animation and the detailed visualization of information about the code and its execution possible. The main goal is to allow the students to have a better performance in programming disciplines.*

Resumo*. Este artigo apresenta a ferramenta JavaTool, cujo objetivo é facilitar a aprendizagem de programação, utilizando Java como linguagem de ensino em disciplinas introdutórias do curso de computação. A proposta baseia-se no desenvolvimento de um interpretador para a linguagem Java, de forma a possibilitar a animação e visualização detalhada de informações do código e de sua execução. A meta final é permitir aos estudantes obter um melhor rendimento nessas disciplinas de programação.*

1. Introdução

O baixo índice de assimilação dos estudantes nas disciplinas cujos requisitos exigem o conhecimento de programação tem sido um grande problema enfrentado em muitas instituições (Santiago e Dazzi, 2004). Dada a importância da atividade de programação em um curso de ciência da computação, uma das abordagens mais comuns para enfrentar esse problema é o desenvolvimento de ferramentas que venham a facilitar o ensino de programação de forma didática e pedagógica.

A aprendizagem de conceitos iniciais de programação é difícil para muitos estudantes iniciantes. Uma das razões, de acordo com Sajaniemi e Kuittinen (2003), é que os programas envolvem entidades abstratas como: construções formais de *loops*, ponteiros, *arrays*, etc., conceitos estes que os estudantes não estão familiarizados. Dessa maneira, métodos e técnicas são necessários para auxiliar os estudantes a alcançarem uma aprendizagem inicial destes conceitos de programação.

Ao longo dos anos, muitas ferramentas têm sido desenvolvidas para o ensino de programação, utilizando as mais diversas abordagens, como animação de algoritmos, utilização de pseudo-linguagens, entre outras (Almeida et. al, 2004; Brown, 1988; Moreno et. al, 2004; Silva e Favero, 2005), entretanto, ainda não existe uma ferramenta

para ensino de conceitos introdutórios de programação utilizando uma sintaxe simplificada da linguagem Java.

Esse artigo apresenta o desenvolvimento da ferramenta JavaTool, cujo objetivo é facilitar o ensino de programação auxiliando professor e estudante no processo de aprendizagem. Busca-se desenvolver uma solução própria que possibilita uma completa exploração do código fonte, via um interpretador.

A idéia desta ferramenta surgiu do sucesso de uma ferramenta similar desenvolvida para linguagem SQL (Lino et. al, 2007) que mostrou uma melhora de performance de 10% a 15% no conceito final dos estudantes, comparando-se turmas que usaram a ferramenta com as turmas que não a utilizaram. O JavaTool aproveitou a infra-estrutura do AVA (Ambiente Virtual de Aprendizagem) do LabSQL. Assim este trabalho foca mais especificamente na ferramenta de animação de código Java.

Esse artigo está organizado em cinco seções incluindo esta. A seção 2 apresenta alguns trabalhos relacionados à ferramenta; a seção 3 apresenta o JavaTool e trata dos seus aspectos de implementação; a seção 4 trata das principais funcionalidades do JavaTool e um exemplo de sua utilização e a seção 5 apresenta as conclusões e trabalhos futuros.

2. Trabalhos relacionados

Na literatura encontram-se diversas ferramentas desenvolvidas para o ensino de programação. Um dos métodos utilizados é o uso do pseudocódigo, uma forma genérica de escrever um algoritmo, utilizando uma linguagem mais informal, podendo ser entendida por qualquer pessoa, sem necessidade de conhecer a sintaxe de nenhuma linguagem de programação. O uso do pseudocódigo também pode auxiliar a escrever um programa menor e mais fácil de ser entendido. Nesta categoria, Almeida et. al, (2004) descrevem o ambiente AMBAP e Silva e Favero (2005) descrevem o ambiente VisualPseudo. A utilização de pseudocódigo também apresenta uma desvantagem, a qual é a falta de padronização. Dada a sua natureza não estruturada, é um tanto difícil de padronizá-lo, logo, um programador pode não entender a lógica de um programa escrito por outra pessoa (Halstead, 2007).

Noutra linha, Gomes e Mendes (2000) descrevem o ambiente SICAS, centrado em fluxogramas com símbolos diagramáticos, focando mais a estrutura dos algoritmos. A linha de visualização de programas vem sendo pesquisada há algum tempo e consiste no uso da computação gráfica e da animação para auxiliar a ilustrar e a apresentar programas de computador, processos e algoritmos. Programas para visualização podem ser utilizados no ensino para ajudar os estudantes a entenderem como os algoritmos funcionam e também podem ser usados para auxiliar os programadores a compreenderem melhor o seu código (Frick, 1997).

A visualização de programas foca na representação gráfica da execução de um programa e seus dados. A representação visual de dados e da seqüência de execução de um programa pode proporcionar aos usuários valiosa informação em muitos aspectos do desenvolvimento e da execução de um programa, como depuração ou comparação entre diferentes algoritmos em termos de desempenho, por exemplo. (Ellershaw e Oudshoorn, 1994).

Uma das abordagens da visualização de programas é a animação de algoritmos, que consiste em produzir visualizações animadas de algoritmos, geralmente durante a execução de um programa. O mais conhecido sistema para visualização de programas é o sistema BALSA (Brown, 1988). Nesta linha, o JEliot (Moreno et. al, 2004) é um sistema que permite a visualização e animação da execução de um código em Java, inclusive mostrando as estruturas de dados. Em termos de linguagem de programação o JavaTool se assemelha ao JEliot, que anima código Java, porém ao mesmo tempo também o JavaTool difere dele, pois permite a animação de código com sintaxe simplificada, mais indicado para o ensino de algoritmos.

Diante desse contexto, a motivação para o desenvolvimento do trabalho apresentado nesse artigo é criar uma ferramenta que tem como objetivo ser um ambiente de desenvolvimento para ensino de conceitos introdutórios de programação utilizando uma sintaxe simplificada da linguagem Java, onde o estudante pode: editar código Java, compilar, depurar e visualizar a execução do código. O JavaTool permite que o usuário crie o código e visualize o funcionamento do algoritmo, com suas respectivas estruturas de dados, facilitando não apenas seu entendimento, mas também a posterior depuração do código.

3. A Ferramenta JavaTool

Uma das maiores dificuldades de um estudante de programação é entender cada passo da execução do programa. Aprender a gramática de uma linguagem de programação, consertar erros de sintaxe em um programa, desenvolver novos algoritmos, escrever um novo programa, depurar e consertar erros em um programa são as principais dificuldades de um estudante que inicia um curso de programação, em ordem crescente de dificuldade (Miyadera, 1999).

De acordo com os resultados obtidos em pesquisas no trabalho de Miyadera (1999), são apresentados alguns requisitos para um sistema educacional: a) o sistema deve exibir explanações escritas junto com a animação do programa; b) os estudantes preferem visualizar o código dos programas, explicações e animações ao mesmo tempo; c) os estudantes gostam de ter controle sobre a execução da animação, o sistema deve então permitir a execução do algoritmo continuamente, passo a passo, *ir* e *voltar*, e deve permitir também o controle da velocidade da execução da animação; d) a interação com a animação melhora a aprendizagem, o sistema deve então permitir entrada de dados pelos usuários. A implementação da ferramenta proposta levou em consideração todos estes requisitos.

O JavaTool é uma ferramenta para auxílio ao ensino de programação, para ser utilizada nas disciplinas iniciais de programação, nas aulas práticas de laboratório. Futuramente a ferramenta JavaTool deverá ser integrada a um ambiente de ensino *Web*, como por exemplo o *Moodle* (Moodle, 2008), podendo assim ser utilizada como uma ferramenta de um laboratório virtual de programação. No ambiente *Web*, o estudante poderá (i) examinar a animação de um exemplo de algoritmo presente no conteúdo ministrado pelo professor e /ou poderá (ii) selecionar um exercício proposto pelo professor e desenvolver uma solução. O estudante poderá editar código Java, compilar, depurar o código e visualizar a animação do código.

No laboratório, a idéia central do ambiente é animar códigos de algoritmos escritos com uma sintaxe reduzida da linguagem Java em um curso inicial de

programação, sem abordar conhecimentos relacionados à programação orientada a objetos. A ferramenta tem como principal objetivo apoiar estudantes de forma didática durante o início da aprendizagem de programação utilizando a linguagem Java. Isto é proposto através do método de visualização de programas, em forma de animações 2D, fazendo com que a abstração dos códigos de algoritmos seja visível para o aprendiz.

Para realizar os objetivos a ferramenta JavaTool implementa um editor que conhece a sintaxe do código Java e facilita a correção de erros de sintaxe. Depois permite animar o código, via um interpretador. Dá suporte à animação da maior parte dos conceitos que são ensinados em disciplinas de programação; permite que os dados sejam representados em vários formatos para exibição (binário, hexadecimal, etc.); permite a visualização de um histórico da execução do programa de forma gráfica e também na forma de texto.

Na Figura 1 é apresentada a interface do JavaTool. Como a maioria dos aplicativos, contém um *menu* com funções básicas como: criar novo arquivo, abrir arquivo, salvar arquivo, editar arquivo, imprimir, um *menu* de ajuda, entre outras funções. O código é escrito na área de EDIÇÃO e os resultados de saída do programa, incluindo possíveis erros que serão exibidos no CONSOLE. A partir do momento que o usuário clica o botão *play* no *menu* da área de ANIMAÇÃO o código é animado. A área designada HISTÓRICO corresponde a uma descrição textual do que aconteceu durante cada passo da animação. A área ANIMAÇÃO também exibe o histórico na forma de imagens, podendo ser visualizadas por meio dos botões no *menu* de animação: *seta para direita* e *seta para esquerda.*

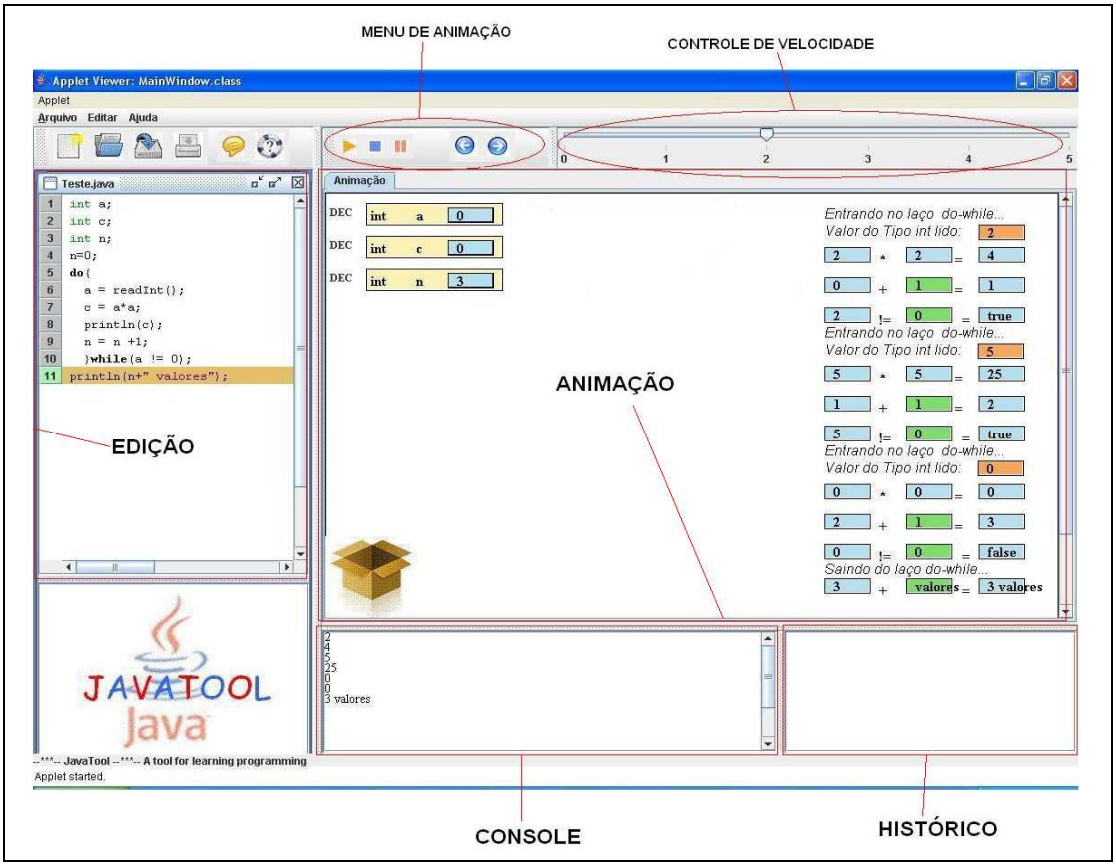

Figura 1. Interface do JavaTool

3.1. Aspectos de Implementação

Foi desenvolvido um compilador próprio para a ferramenta, composto de três partes: analisador léxico, analisador sintático e analisador semântico. O analisador léxico foi gerado por meio de um gerador de analisador léxico e sintático (JavaCC) e foi adaptado para a ferramenta. Já os analisadores sintático e semântico foram desenvolvidos de modo que todos os parâmetros pudessem ser controlados para que se pudesse extrair o máximo de informação do código fonte para a posterior animação da execução. O desenvolvimento dos analisadores sintático e semântico foi importante também no que diz respeito à depuração do código fonte e tratamento de erros, para a exibição de mensagens de erro de forma que o usuário fosse melhor direcionado na correção do(s) erro(s).

A Figura 2 mostra as etapas de processamento que acontecem durante a execução da ferramenta. Após sua edição, o código fonte passa pela análise léxica, onde é transformado em uma "fita de *tokens*", que serve de entrada para a análise sintática. Na análise sintática são reconhecidos erros da sintaxe da linguagem. Nesse momento de transição é gerado um código intermediário para o avaliador. O avaliador é quem realiza as operações e atribuições especificadas no código fonte e ao mesmo tempo gera um código intermediário para a animação.

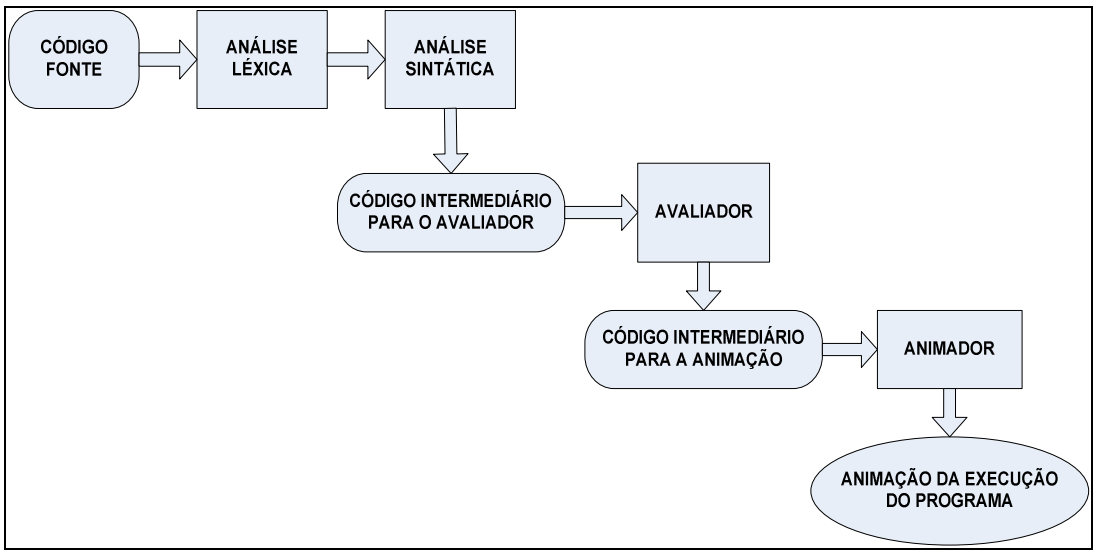

Figura 2. Etapas de processamento

4. Interface e Funcionalidades do JavaTool

Buscou-se projetar uma interface que não fosse complexa para estudantes iniciantes no estudo de programação. Além disso, procurou-se também seguir os requisitos de uma ferramenta educacional propostos por Miyadera (1999). Os recursos da interface propostos pelo JavaTool são:

- Animação do código: ao clicar no botão *play*, após a edição do código fonte, o código é compilado e, caso não existam erros de compilação, a animação é exibida no painel de animação. A interface possui quatro áreas, baseando-se no trabalho de Moreno et. al (2004): a) área para variáveis do tipo *String* e tipos primitivos, onde são declaradas as variáveis; b) área para vetores, onde são declarados e inicializados os vetores, c) área para constantes, de onde saem as

constantes que são utilizadas no código fonte; d) área para avaliação: onde são avaliadas as expressões e ficam registradas graficamente e textualmente as operações que são realizadas durante a animação(avaliação de expressões, entrada de dados, chamadas de métodos, etc.). Durante a animação, a linha do código fonte que está sendo animada é destacada com a cor laranja.

- Entrada de dados: durante a operação de leitura de dados via teclado, exibe-se uma janela para que o usuário entre com o valor. Ao terminar de digitar o valor o usuário pressiona a tecla *Enter* do teclado ou clica com o mouse sobre o botão *OK* para que a animação prossiga.
- Visualização dos dados: durante a animação ou após esta, ao clicar em cima de uma variável no painel de animação, o usuário poderá visualizar o valor desta em diversos formatos (decimal, binário, hexadecimal ou octal), dependendo do tipo da variável. Esta funcionalidade serve, por exemplo, para que o aluno visualize operações com bits (operações de deslocamento de bits, etc.). Outro recurso é a utilização de cores diferentes para cada tipo (*int, double*, etc) e também para valores constantes, valores provindos da entrada de dados, valores resultantes da avaliação de uma expressão e valores das variáveis.
- Histórico da execução: com essa função, o usuário não precisa executar a animação novamente caso queira rever algum detalhe, ficando todas as informações registradas visualmente e também na forma de texto. O usuário pode visualizar o histórico passo a passo, bastando clicar nos botões *seta para direita* e *seta para esquerda*.
- Mensagens de erro: O JavaTool exibe, no painel de animação, os erros de compilação e de execução, caso existam no código fonte. São exibidas mensagens que contém uma descrição do erro (por exemplo: “divisão por zero”), a linha que contém o erro e o caractere onde possivelmente se encontra o erro.

O JavaTool implementa alguns recursos da linguagem Java. Os recursos que são possíveis de se animar no JavaTool são:

- Todos os tipos primitivos(*byte, short, int,long, float, double, char, boolean*) e *String*;
- Arrays do tipo unidimensional;
- Estruturas de seleção *if-then, if-then-else* e *switch*;
- Estruturas de repetição *while, do-while* e *for*;
- Chamada de métodos.
- Alguns métodos da classe *Math*(*ceil, floor, max, min, sqrt, pow, random, round*) e da classe *String(length, charAt, toUpperCase, toLowerCase, substring, trim, replace, valueOf)*;

Para facilitar a edição do código Java no JavaTool, optou-se por simplificar em alguns aspectos a escrita da linguagem Java utilizada. A Figura 3 mostra um exemplo de um código Java normal e a Figura 4 mostra um código com as mesmas operações com a sintaxe simplificada do JavaTool:

```
1 import java.util.Scanner;
2 public class Teste{
3    public static void main(String[] args){
4        int num;
5        Scanner s = new Scanner(System.in);
6        num = s.nextInt();
7        System.out.println(num);
8    }
9 }
```

Figura 3. Código Java

```
int num;
num = readInt();
println(num);
```

Figura 4. Código no JavaTool

No código da Figura 4, a definição de classe e método *main* não é necessária, o código ficou reduzido para apenas três linhas. As linhas 1, 5 e 6 são relativas à leitura de um inteiro. Não é necessário o uso das três primeiras linhas do código anterior, com a declaração *import*, definição da classe e do método principal. Para a leitura do valor da variável *num* (valor do tipo *int*), utiliza-se o método *readInt*, implementado para o JavaTool (caso fosse um valor do tipo *double*, utiliza-se o método *readDouble*, para *char*, *readChar*, etc) , para não se envolver o conceito de instanciação de um objeto (*new Scanner*, no código da Figura 3, na linha 5), e nem fazer uso da classe *System*. Idem para a saída de dados: foram implementados os métodos *println*, para o JavaTool, que equivale ao comando *System.out.println* no Java e *print*, que equivale a *System.out.print* no Java.

A sintaxe simplificada tem um caráter pedagógico, focada para o ensino de programação/algoritmos no início de um curso. Assim, permite-se rodar uma seqüência de linhas de código sem especificação de classes ou métodos, omitindo-se as declarações para importação de classes, etc. A idéia da sintaxe simplificada é focalizar os conceitos fundamentais de programação sem complicar o estudante com os conceitos de programação que focam a programação em larga escala, que acontecerá logo mais adiante no curso. Assim o aluno terá uma boa base para cursar mais adiante uma disciplina do tipo “Programação em Java”.

4.1. Exemplo de utilização do JavaTool

Aqui é mostrado um exemplo de utilização do JavaTool na resolução de um problema comum abordado em cursos introdutórios de programação envolvendo o tópico Estruturas de Repetição. Segue o enunciado do problema:

“Deseja-se ler as notas dos alunos de uma turma e, após isso, dizer qual a quantidade de notas superiores a cinco; o programa termina com a digitação de um valor negativo”.

O código correspondente editado no JavaTool é exibido na Figura 5:

```
double nota;
int n = 0;
do {
        nota = readDouble ();
        if (nota > 5)
            n = n+1;
} while (nota > 0);
println("Notas > 5"+n);
```

Figura 5. Código no JavaTool

A variável *n* representa o número de notas superiores a cinco. A condição para sair do laço é a digitação de um valor negativo. Ao se entrar com o valor 9.5, a condição *9.5 > 5* é então avaliada. Como se trata de um valor maior que cinco, a variável *n* é incrementada em uma unidade (*0+1 =1*). Em seguida, entra-se com o valor 4.5, menor que cinco, assim a variável *n* não é incrementada. Por fim, entra-se com o valor -1 e a repetição se encerra. O resultado obtido no painel de animação do JavaTool é exibido na Figura 6, e o resultado no *console* é exibido na Figura 7.

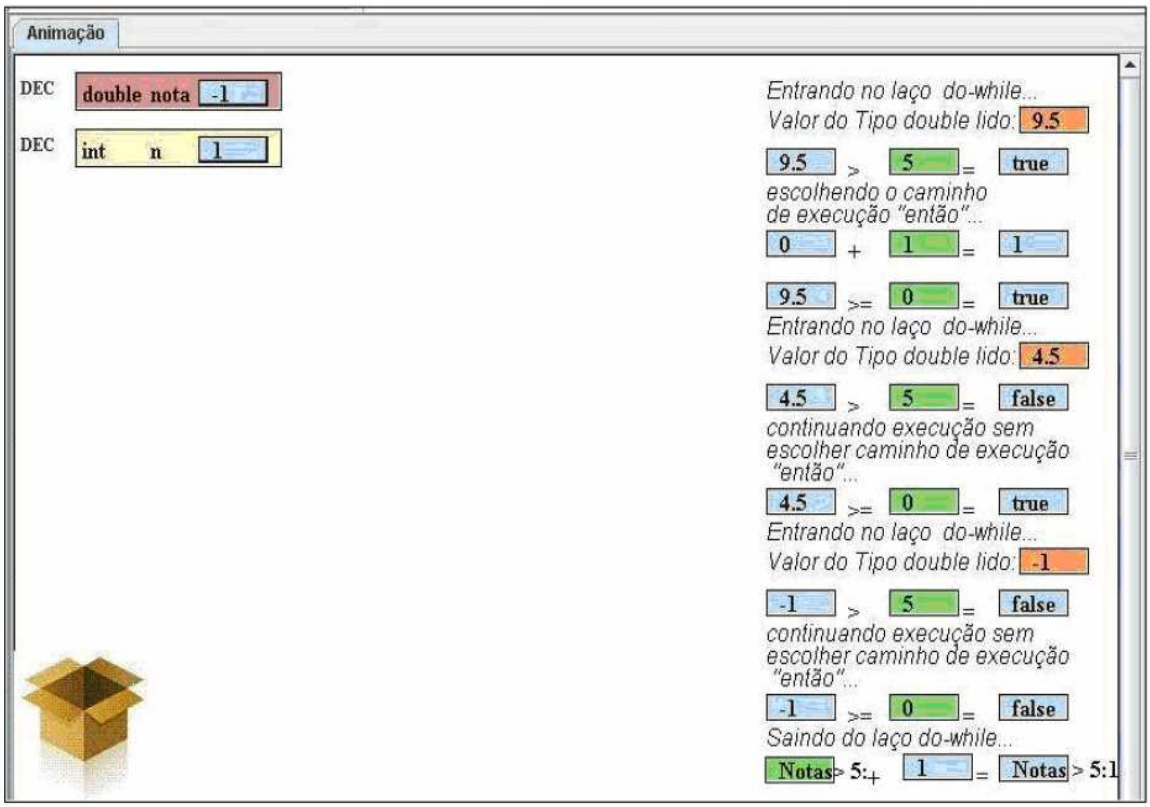

Figura 6. Resultado da execução no painel de animação

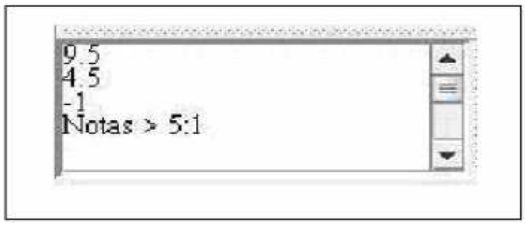

Figura 7. Resultado da execução no console

5. Conclusões

O JavaTool foi desenvolvido para minimizar as dificuldades de aprendizes em disciplinas de programação, por meio da animação da execução do código. A ferramenta busca tornar possível a visualização do comportamento da execução do código fonte. O JavaTool foi desenvolvido após uma experiência de sucesso de nosso grupo de pesquisa com um ambiente similar para programação de SQL, que já está num estado mais desenvolvido. O ambiente para SQL (Lino et. al, 2007) foi utilizado nos últimos dois anos em dezenas de turmas e a sua avaliação mostrou uma melhora de performance de 10% a 15% no conceito final dos estudantes, comparando-se turmas que usaram a ferramenta contra turmas que não usaram a ferramenta, para a mesma disciplina.

Acreditamos que a utilização do JavaTool terá no mínimo um desempenho similar. A partir deste ano estaremos utilizando como ferramenta nas atividades práticas de laboratório. Por outro lado estamos trabalhando na integração dele com uma plataforma virtual de ensino, por exemplo, o *Moodle*.

Acreditamos também que o uso de uma sintaxe reduzida da linguagem Java no início de um curso de programação permite focalizar mais nos conceitos essenciais de programação; o estudante é liberado da memorização de detalhes excessivos da sintaxe da linguagem. Por outro lado, com o uso de uma sintaxe simplificada do Java, o estudante está sendo preparado já com conceitos da linguagem Java a ser estudada mais adiante em disciplinas de programação mais avançadas.

As principais funcionalidades do JavaTool são: edição de código Java, compilação, depuração, visualização da animação do código, exibição dos dados em vários formatos (binário, hexadecimal e octal) e a visualização do histórico de execução da animação tanto gráfico como textual.

O JavaTool foi pensado como uma ferramenta para uso em aulas de prática de programação, sejam presenciais ou na modalidade a distância. Por exemplo, poderá animar os exemplos expostos pelo professor bem como os exemplos existentes em um conteúdo digital de uma disciplina de programação. Como trabalhos futuros são considerados os seguintes itens:

- Melhorar o editor de código Java, incluindo mais funcionalidades, como o recurso de auto-completar código e análise sintática em tempo de digitação, entre outras;
- Incluir uma funcionalidade de análise de performance de um código Java, permitindo ao estudante aprender a otimizar o código desenvolvido, fazendo versões alternativas mais eficientes, considerando o tempo de execução e consumo de memória;

- No ambiente AVA permitir a comparação em termos de complexidade e de performance com a(s) proposta(s) pelo professor.

Referências

Almeida, E. S., Herreral, J.D., Filho, L.J. S., Almeida, H. O., Costa, E.B., Vieira, B.L., Melo, M.D. (2004) "Um Ambiente Integrado para auxílio ao Ensino de Ciência da Computação", In: Revista Digital da CVA, Vol. 2 – nº8.

Brown, M. H. (1988) "Algorithm Animation". MIT Press.

Ellershaw, S., Oudshoorn, M. "*Program Visualization*" - The State of the Art. Department of Computer Science, University of Adelaide, 1994. Disponível em: <http://citeseer.ist.psu.edu/ellershaw94program.html>. Acesso em: 11 de Novembro de 2007.

Frick, A. "*Software Visualization*". 1997. Disponível em: <http://www.info.uni-karlsruhe.de/~frick/SoftVis/>. Acesso: 30 de Outubro de 2007.

Gomes, A., Mendes, A., (2000) "Suporte à aprendizagem da programação com o ambiente SICAS", V Congresso Ibero-Americano de Informática Educativa, Vina del Mar - Chile.

Halstead, N. "*Uses of Pseudo Code in Development*". 2007. Disponível em: <http://blog.assembleron.com/2007/06/03/uses-of-pseudo-code-in-development/3>. Acesso em: 11 de Novembro de 2007.

Lino, A. D. P., Silva, A. S., Harb, M.P.A.H., Favero, E.L., Brito, S.R. (2007) "Avaliação automática de consultas SQL em ambiente virtual de ensino-aprendizagem". Conferencia Ibérica de Sistemas y Tecnologías de la Información. CISTI .

Miyadera, Y., Huang, N., Yokoyama, S. "*A programming language education system based on program animation*". Tokyo Gakugei University, 1999. Disponível em <http://www.ifip.org/con2000/iceut2000/iceut08-04.pdf>. Acessado em 12 de Novembro de 2007.

Moodle (2008) Disponível em: <http://moodle.org/>. Acessado em 12 de maio de 2008.

Moreno, A., Myller, N., Sutinen, E., Ben-Ari, M. (2004) "*Visualizing Programs with Jeliot 3*", Proceedings of the Advanced Visual Interfaces.

Sajaniemi, J., Kuittinen, M. (2003) "*Program Animation Based on the Roles of Variables. Proceedings of the 2003 ACM symposium on Software visualization*". San Diego, California. P. 7 – ff.

Santiago, R., Dazzi, R.,Rudimar L.S. (2004) "Ferramenta de apoio ao ensino de algoritmos" In: SEMINÁRIO DE COMPUTAÇÃO, Blumenau.

Silva, M.A.B., Favero, E.L.(2005) "Compiladores e Interpretadores Uma Abordagem Prática".