JEduc: Um ambiente dedicado ao ensino de Java

Cássia Alves Perego
Universidade do Oeste Paulista (UNOESTE), Fac. de Informática de Pres. Prudente,
Presidente Prudente, Brasil, 19050-680
cassia@unoeste.br

and

Silvia de Castro Bertagnolli
Universidade Federal do Rio Grande do Sul (UFRGS), Instituto de Informática,
Porto Alegre, Brasil, 91501-970
silviacb@inf.ufrgs.br

and

Maria Lúcia Blanck Lisbôa
Universidade Federal do Rio Grande do Sul (UFRGS), Instituto de Informática,
Porto Alegre, Brasil, 91501-970
llisboa@inf.ufrgs.br

Abstract

This paper introduces JEduc, an IDE – Integrated Development Environment intended to support teaching the object-oriented programming language Java to beginners. Most of available environment that integrate editing, compiling and execution of Java programs are too complex to be used in introductory courses or not freely available, and they are not intended to act as a pedagogical tool. Therefore, we started to develop JEduc with simplicity, portability and adaptability in mind. JEduc uses a large number of aspects to simplify some Java platform complexities, it is written in Java to achieve portability and it is an open source code to promote adaptability.

Keywords: Teaching Environment, Java Programming Language, Introductory Programming.

Resumo

Este artigo apresenta JEduc, um IDE – Ambiente de Desenvolvimento Integrado destinado ao ensino da linguagem de programação orientada a objetos Java para alunos iniciantes. A maioria dos ambientes disponíveis para edição, compilação e execução são complexos e não oferecem suporte didático sobre a linguagem para professores e alunos. Assim, surgiu a idéia de desenvolver o ambiente JEduc que em sua essência deve ser simples, portável e aberto. JEduc possui várias características visando a simplificação de algumas complexidades encontradas na plataforma Java. Além disso, ele foi completamente construído em Java para oferecer portabilidade e o seu código é aberto visando promover adaptabilidade.

Palavras-chave: Ambiente de Apoio ao Ensino, Linguagem de Programação Java, Introdução à Programação.

1 Introdução

O uso dos computadores no suporte às atividades de ensino superior aumentou significativamente nos últimos anos. Com isso, as instituições educacionais estão procurando preparar alunos e professores para realizar o progresso em informática, um campo que evolui rapidamente [16], além de reconhecer a importância da atualização tecnológica e pedagógica. Os alunos necessitam desenvolver uma habilidade em programação que os permita evoluir da tecnologia de hoje para os desafios do futuro.

O processo de aprender a programar é difícil e árduo para a maioria dos alunos. Mesmo a aprendizagem de conceitos básicos e a sua aplicação na resolução de problemas concretos, introduz problemas difíceis para muitos alunos. O principal deles está relacionado com o um estudo baseado na prática, como escrever códigos com vistas à reutilização, por exemplo.

Além disso, aprender a programar de forma orientada a objetos impõe problemas antes não encontrados: o programador precisa descrever quais entidades fazem parte do universo do programa e como elas interagem entre si para atingir o resultado desejado. Dessa maneira, implementar sistemas orientados a objetos é muito mais do que aprender uma linguagem orientada a objetos; na verdade, consiste em uma nova forma de estruturar e de pensar a solução de um problema.

A consciência dessas dificuldades fez com que educadores de várias partes do mundo [14], [26], [11] e [16] procurassem por estratégias e materiais que minimizassem as dificuldades encontradas por muitos alunos e professores de programação. Nota-se que pode ser uma tarefa dura e desafiadora a introdução e/ou migração de um conteúdo conhecido para conceitos e ferramentas pedagógicas mais novas, mas isso deve ser feito para acompanhar os avanços na tecnologia.

Ao longo dos anos, diversas ferramentas foram propostas com o objetivo de apoiar/facilitar o esforço de aprendizagem da programação pelos alunos [17]:

(i) mini-linguagens: as linguagens de programação, geralmente, costumam ser rígidas, extensas e complicadas para um aluno inexperiente. Para reduzir esses problemas uma nova abordagem tem sido proposta: criar mini-linguagens baseadas em subconjuntos de comandos de linguagens de programação convencionais. O MiniJava [26] é um exemplo de uma mini-linguagem baseada em Java;

(ii) mundos programáveis: o objetivo principal é permitir a localização, em um ambiente simples, da ação do programa. Estes programas utilizam conceitos básicos de programação, explorando a capacidade dos alunos no desenvolvimento de programas em um ambiente simples. Um exemplo desse tipo de ferramenta é o “Karel the robot” [19];

(iii) ambientes de desenvolvimento controlados: compreendem ambientes com um agregado de ferramentas voltadas para o desenvolvimento de sistemas profissionais. No entanto, esses ambientes são complexos para a maioria dos alunos que iniciam na aprendizagem da programação. Assim, têm sido propostos alguns ambientes simplificados, que incluem poucas funcionalidades, as quais oferecem suporte para algumas atividades dos alunos. O jGRASP [6], Ready [24] e o BlueJ [12] são exemplos de ambientes dessa categoria.

O objetivo desse trabalho é apresentar o ambiente JEduc [8], [22] e [4] cuja idéia central é simplificar, de forma transparente, algumas características da linguagem Java, no intuito de agilizar o processo de desenvolvimento dos primeiros programas orientados a objetos pelos alunos. Busca reduzir a quantidade de codificação exigida para a implementação e de conhecimento minucioso das bibliotecas de componentes da linguagem.

Vários pesquisadores e professores [5], [14], [15], [10], [25] e [16] acreditam que Java oferece o suporte adequado ao ensino dos conceitos do paradigma orientado a objetos, e também é aplicável no desenvolvimento da habilidade de solucionar problemas com alunos das disciplinas introdutórias. Optou-se pela linguagem de programação Java principalmente pelo fato dela apresentar características desejáveis para um ensino conceitual bastante atual, pois oferece suporte aos mais diversos tipos de aplicações, tais como: concorrentes, distribuídas, Internet, entre outras. Disso decorre que Java pode ser usada ao longo de todo um curso de graduação, em diferentes disciplinas, bem como pode ser estendida e especializada através do desenvolvimento de bibliotecas de classes particulares que atendam necessidades específicas das disciplinas.

Este artigo prossegue apresentando, na seção 2, alguns problemas encontrados nos IDEs atuais que dificultam o aprendizado de programação por alunos iniciantes. Na seção 3, são abordados aspectos na construção de IDEs adaptáveis com vistas à reutilização. Nas seções 4 e 5, são discutidas as simplificações necessárias para o uso de Java em disciplinas introdutórias e, em decorrência disso, as características implementadas no ambiente JEduc. Na seção 6 estão as conclusões deste artigo.

2 Ambientes de Programação

Existem os mais diversos ambientes de desenvolvimento de programas Java, variando dos mais simples e gratuitos até ambientes profissionais com ferramentas complexas. Ao considerar a adoção desses ambientes para uso em disciplinas de introdução à programação, emergem algumas dificuldades, tais como:

(i) grande exigência de recursos de memória: devido principalmente ao grande número de componentes gráficos e recursos disponíveis;

(ii) dificuldade de instalação: devido principalmente a um número elevado de componentes e propriedades a serem configuradas;

(iii) excessiva quantidade de ícones e menus: a grande quantidade de elementos visuais sobrecarrega a interface com características desnecessárias para o propósito do ensino introdutório de programação;

(iv) exigência de criação de "projetos": para o aluno iniciante em programação, torna-se um fator "complicador" adicional ter que trabalhar com o conceito de projetos. O aluno deve ter a oportunidade de visualizar apenas unidades simples de implementação.

Esses problemas são decorrentes principalmente do fato que grande parte dos IDEs atuais estão voltados para o desenvolvimento de sistemas profissionais. Logo, isso os torna complexos para a maioria dos alunos que iniciam na aprendizagem da programação. Por outro lado, parece haver um consenso a respeito dos requisitos essenciais que uma ferramenta deve possuir para facilitar a aprendizagem da programação, entre eles:

(i) permitir a interação do aluno com o ambiente de forma que ele possa atuar e desempenhar um papel ativo no processo de aprendizagem;

(ii) seguir um padrão de interface visual comum à maioria das aplicações existentes, exigindo, assim, pouco tempo de aprendizagem e ao mesmo tempo tornando seu uso atrativo;

(iii) possibilitar a instalação nos mais diversos sistemas operacionais e arquiteturas de hardware existentes atualmente;

(iv) baixos custos de aquisição fazem com que um elevado número de alunos e instituições adotem o produto como ferramenta didática.

A partir dos problemas e dos requisitos enumerados anteriormente conclui-se que para uso acadêmico o ambiente de desenvolvimento não pode ser complexo sob pena de fazer com que os professores percam muito tempo para explicar o funcionamento do IDE, ao invés de concentrar-se nos exemplos e conceitos de programação. Assim, este artigo apresenta o ambiente JEduc, um IDE projetado para o ensino da linguagem Java em disciplinas introdutórias de programação, que, para assegurar essa característica, possui apenas as ferramentas mínimas para o propósito do ensino.

3 Construção de Ambientes Adaptáveis

Segundo Ossher [18], é importante inovar na implementação de *software* e ferramentas, buscando facilitar tanto o desenvolvimento inicial quanto uma posterior adaptação e integração a novos contextos. A reutilização é uma das formas mais adotadas para produzir *software* de maneira rápida, com isso minimizando custos. A aplicação mais conhecida de reutilização consiste na engenharia de componentes. Construir aplicações a partir de componentes reutilizáveis surgiu como uma alternativa para a complexidade encontrada nas fases de desenvolvimento de software [23]. Os componentes possibilitam um aumento na reutilização e na qualidade do software, diminuem o tempo de desenvolvimento e direcionam o objetivo do desenvolvimento da aplicação [28]. A reutilização de componentes envolve alguns problemas: (i) em um grande conjunto de componentes é necessário identificar, rapidamente, aquele cujas funcionalidades atendam aos requisitos do sistema, (ii) a adaptação de um componente deve preservar as funcionalidades pré-existentes, e garantir que falhas e erros não sejam introduzidos nessa adaptação [7].

O JEduc é, na verdade, uma derivação da ferramenta IDECoRe – IDE Core Reuse [3] – que foi desenvolvida com base no princípio de reutilização. Sua implementação consiste na idéia de reutilizar não apenas componentes, mas uma ferramenta inteira. O objetivo é habilitar a criação de ferramentas mais flexíveis e reusáveis, as quais foram denominadas de derivações de IDECoRe. A motivação para construir a IDECoRe surgiu durante o desenvolvimento do ambiente JReflex [1], [9], um ambiente interativo dedicado à reflexão computacional. Nesse momento, foi detectado que para construir um IDE com funcionalidades básicas não existiam ferramentas e/ou componentes de simples compreensão e fácil adaptação. Com isso, toda uma estrutura foi desenvolvida vislumbrando obter a reutilização futura do núcleo da ferramenta, e, principalmente, que isso ocorresse de maneira rápida. Na verdade, foi exatamente o que aconteceu quando o ambiente JEduc [8] começou a ser desenvolvido.

IDECoRe é uma ferramenta totalmente desenvolvida em Java que possui funcionalidades básicas, comuns à maioria dos IDEs, e permite que algumas características sejam customizadas de acordo com a linguagem que se pretende utilizar. Por exemplo, selecionar o compilador e/ou interpretador, APIs ou bibliotecas disponíveis na linguagem. A ferramenta IDECoRe oferece algumas funcionalidades básicas descritas a seguir:

(i) componentes para a utilização de arquivos, cujas principais funcionalidades compreendem abrir, salvar e fechar arquivos, entre outras;

(ii) componentes para edição, através de um editor básico que oferece suporte para as ações de copiar, colar, recortar, entre outras;

(iii) componentes para compilação/execução, interrelacionados com os componentes usados para capturar as mensagens geradas pelo compilador e/ou interpretador;

(iv) componentes para exibir, sob a forma de árvore (*tree view*), APIs – *Application Programming Interfaces* - arquivos ou qualquer outra informação;

(v) componentes GUI – *Graphical User Interface* – responsáveis pela interface gráfica do ambiente.

Um outro aspecto desse ambiente reside no fato dele ser uma ferramenta adaptável e customizável. Para oferecer essas características foi utilizado o mecanismo de reflexão computacional. O uso da reflexão computacional tem sido enfatizado para a construção de sistemas adaptáveis, pois ela possibilita separação de aspectos e transparência. Ao contrário dos mecanismos que são comumente usados na construção de programas em linguagens orientadas a objetos, a reflexão computacional pode ser vista como uma técnica que tem a sua maior aplicabilidade na construção de bibliotecas de componentes abertos que objetivam facilitar o desenvolvimento de aplicações. No ambiente JEduc a reflexão computacional foi utilizada tanto na construção do ambiente como no desenvolvimento dos componentes por ele utilizados.

Na próxima seção, serão apresentadas as principais funcionalidades do ambiente JEduc e uma descrição de como JEduc relaciona-se com IDECoRe.

4 Funcionalidades do JEduc

No decorrer da construção do ambiente JEduc foram desenvolvidos dois módulos: (i) o do aluno e (ii) o do professor. No módulo do aluno será apresentado um ambiente leve de Java, específico para iniciantes em programação, com possibilidade de utilização das classes "simplificadoras" (Figura 1), filtro de mensagens de erro e acesso à documentação de apenas um subconjunto dos pacotes da API de Java. Já no módulo do professor, além de estar disponível todo o ambiente do módulo do aluno sem filtros, será apresentado, sob a forma de documentação HTML, todo um estudo de transcrição dos exemplos de Pascal para Java do conteúdo a ser ensinado numa disciplina introdutória de programação em um curso de ciência da computação. Considerando que a linguagem imperativa Pascal tem sido amplamente adotada em cursos introdutórios, esta abordagem visa simplificar a migração de professores de Pascal para Java, apontando semelhanças e diferenças conceituais entre estas linguagens que representam diferentes paradigmas de programação. A seção 5 oferece mais detalhes sobre estas idéias.

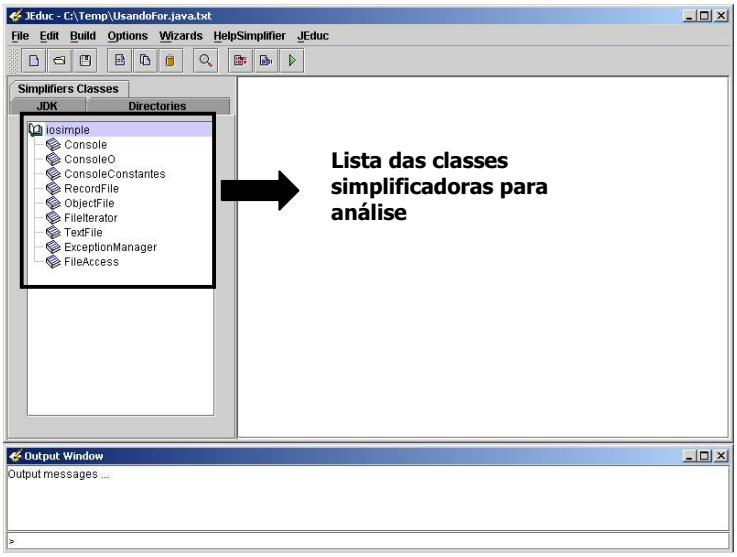

Figura 1 – JEduc: Usando as Classes Simplificadoras

Conforme abordado na Seção 3, o JEduc foi implementado realizando uma extensão da ferramenta IDECoRe. Várias funcionalidades ligadas ao ensino da linguagem Java foram incluídas, mas as funcionalidades básicas do IDECoRe foram preservadas. A seguir são listadas as principais características do ambiente, onde são apontadas as preservadas do IDECoRe e indicadas as adaptadas ou adicionadas:

(i) editor de programas – característica preservada: permite tanto ao ambiente quanto ao usuário fazer alterações no código fonte dos programas, que, posteriormente serão compilados e executados pela máquina virtual Java;

(ii) visualizador de pacotes – adaptação do componente "*tree view*": exibe as propriedades (variáveis, métodos e construtores) de qualquer classe disponível nos pacotes fundamentais da API padrão da linguagem Java. No módulo do aluno, estão apenas disponíveis as classes dos pacotes `java.io`, `java.lang` e `java.util`; já no módulo do professor estão disponíveis todas as classes da API padrão;

(iii) gabaritos sintáticos – funcionalidade adicionada: visa reduzir a ocorrência de erros e facilitar o uso da linguagem Java;

(iv) classes "simplificadoras" – adaptação do componente "*tree view*": exibe a lista das classes implementadas para facilitar a entrada/saída de tipos primitivos, *String*, tipos invólucros, objetos definidos pelo aluno e arquivos na linguagem Java;

(v) diretório de trabalho – adaptação do componente "*tree view*": lista para o usuário os arquivos `.java` existentes no atual diretório de trabalho;

(vi) tratamento das mensagens de erro – funcionalidade adicionada: em seu módulo desenvolvido para os alunos existe um filtro, reduzindo o número de linhas das mensagens de erros a serem exibidas;

(vii) assistente para o professor – funcionalidade adicionada: um assistente, no formato de páginas HTML, cujo objetivo é auxiliar o ensino da linguagem Java como primeira linguagem de programação. Contém um catálogo de programas com exemplos de codificação em Pascal e em Java, apontando diferenças conceituais.

As principais vantagens no uso desse ambiente são: exigência de pouca memória para execução; instalação simples (é necessário apenas informar a localização física das ferramentas – interpretador e compilador – na máquina); reduzido número de ícones e menus, visando simplificar a interface visual; os programas são desenvolvidos de forma independente sem a necessidade de vinculação à "projetos"; e, devido à simplicidade proporcionada, permitir rápida adaptação do usuário ao ambiente. Desta forma, pretende-se assegurar a solução de alguns problemas descritos na Seção 2.

5 Ambiente JEduc e o Ensino de Java

A idéia central desta seção é apresentar algumas características da linguagem Java que tornam o ensino-aprendizagem dessa linguagem, em disciplinas introdutórias, uma atividade difícil. O objetivo é mostrar como um IDE simples, contemplando apenas funcionalidades básicas, pode facilitar aos professores a administração das tarefas de ensino no momento do desenvolvimento dos primeiros programas pelos alunos.

JEduc foi desenvolvido visando minimizar alguns problemas relacionados ao ensino de Java como primeira linguagem de programação, bem como procurar oferecer apoio à migração do modelo orientado a procedimentos para o modelo de objetos. Dentre os diversos problemas levantados, esse ambiente procura resolver os seguintes:

(i) amenizar a complexidade da transição pedagógica de modelos de programação;

(ii) simplificar algumas características da linguagem Java;

(iii) agilizar o processo de desenvolvimento dos primeiros programas em Java pelos alunos;

(iv) reduzir o conhecimento minucioso das bibliotecas de componentes da API da linguagem Java, principalmente as de entrada e saída;

(v) disponibilizar um ambiente e um pacote de classes para auxiliar alunos e professores;

(vi) facilitar aos professores a administração das tarefas de ensino.

Um dos motivos do desenvolvimento do ambiente JEduc e do suporte que ele oferece foi a constatação que um grande número de cursos de graduação em computação adota Pascal como linguagem introdutória [21], aliada à observação de uma tendência de crescimento de Java como linguagem acadêmica. Prevê-se que muitos professores devem migrar de Pascal para Java. A migração envolve, além da capacitação na linguagem, a adoção de novo referencial conceitual. Propõe-se amenizar esta transição de três maneiras: (i) por comparação - como se faz em

Pascal e como se traduz para Java; (ii) por ocultamento - bibliotecas simplificadoras que tornam desnecessário, em um primeiro momento, explicar conceitos complexos; e, (iii) uso de um ambiente integrado de desenvolvimento simples e livre.

Existe uma grande preocupação dos professores, nas disciplinas introdutórias, com a adoção de uma linguagem de programação que satisfaça tanto aspectos teóricos e didáticos quanto técnicos, e que também seja de fácil aprendizado para o aluno. Com a utilização de Java, como linguagem de programação em disciplinas introdutórias, é inevitável a utilização em larga escala de classes e objetos, visto Java ser totalmente orientada a objetos. O objetivo da simplificação pretendida é unicamente esconder detalhes de implementação sem descaracterizar os princípios de programação orientada a objetos.

Existem várias abordagens [20], [26] para solucionar problemas existentes em Java que dificultam a sua aplicabilidade como linguagem introdutória. Nas próximas subseções deste artigo serão apresentadas somente as abordadas pelo ambiente JEduc.

5.1 Entrada e Saída de Dados

Para alunos iniciantes, a leitura de dados via console pode ser destacada como um ponto de complexidade. Em Java, existem inúmeras classes que possibilitam a entrada/saída de tipos de dados, sejam eles primitivos, *Strings*, invólucros ou arquivos. Assim, é necessário que os alunos possuam um conhecimento minucioso da linguagem para poder usar essa funcionalidade.

A solução proposta consiste em integrar, ao ambiente JEduc, classes "simplificadoras" que ocultem as complexidades relacionadas às classes de I/O padrão da linguagem Java e exigem a compreensão de conceitos sofisticados como *buffers, streams* e tratamento de exceções.

O pacote `ufrgs.inf.iosimple` das classes "simplificadoras", em um primeiro momento, possuía apenas uma classe – a `ConsoleP`, que foi definida e construída com a intenção de simplificar a entrada e saída de tipos primitivos, *String* e das classes invólucros. Além disso, foi realizado o tratamento local das exceções que poderiam ocorrer para cada entrada evitando, assim, a necessidade de uso do bloco `try/catch` no código do usuário. A definição dessa classe sofreu uma influência direta da classe `Keyboard` desenvolvida por Lewis e Loftus [15]. A segunda etapa consistiu no projeto e construção da classe `ConsoleO`, que visa simplificar a entrada e saída de objetos definidos pelo usuário. A interface da classe `ConsoleO` é bastante simples, possuindo apenas dois métodos, a saber: `readAll()` e `printAll()`, destinados à entrada e saída de todos os campos definidos e herdados de objetos, os quais utilizam métodos definidos na classe `Infs_Reflex` e `ConsoleP`. Os métodos dessa classe são todos reflexivos, isto é, utilizam a técnica de reflexão computacional para realizar introspecção ou intercessão sobre os campos/métodos de um objeto [2]. Na terceira etapa, foram definidas as classes simplificadoras para a leitura, escrita e pesquisa em arquivos. As classes `TextFile` e `ObjectFile` implementam a interface `FileAccess` particularizada para arquivos texto e arquivos de objetos, respectivamente.

Para exemplificar melhor, abaixo estão ilustrados programas que realizam a entrada de dados para todos os atributos da classe `Student`, a saber: `name` e `birthdate`. Em um primeiro momento, a entrada é realizada utilizando as classes existentes no pacote `java.io`, conforme mostra o exemplo 1.

Exemplo 1 – Entrada da classe Student

```
1. class StudentTestSC{
2.    private static BufferedReader in = new BufferedReader
3.                (new InputStreamReader(System.in));
4.    public static void main( String args[]){
5.       Student Mary = new Student();
6.       try{
7.          System.out.print("Enter a String: ");
8.          Mary.name=in.readLine();
9.          System.out.print("Enter an int: ");
10.         Mary.birthDate.day=Integer.parseInt(in.readLine());
11.         System.out.print("Enter an int: ");
12.         Mary.birthDate.month=Integer.parseInt(in.readLine());
13.         System.out.print("Enter an int: ");
14.         Mary.birthDate.year=Integer.parseInt(in.readLine());
15.     } catch(IOException e){System.out.println("I/O error!");}
17.   }
18. }
```

No exemplo 1, apresentado anteriormente, podem ser destacados os seguintes pontos de complexidade:

(i) declaração do *buffer* de entrada (linha 2-3);

(ii) sistema de tratamento de exceções (linhas 6 e 15);

(iii) acesso aos campos de `birthDate` (linhas 10, 12 e 14);

(iv) conversão de tipos (linhas 10, 12 e 14).

Utilizando uma das classes "simplificadoras", integrada ao ambiente JEduc, denominada `ConsoleP`, que implementa métodos de entrada/saída de tipos primitivos, invólucros e *String*, o programa pode ser codificado conforme esquematiza o exemplo 2.

Exemplo 2 – Entrada da classe Student usado a classe`ConsoleP`

```
1. import ufrgs.inf.iosimple.ConsoleP;
2. class StudentTestCC {
3.    public static void main( String args[]){
4.       Student Mary= new Student();
5.       Mary.name=ConsoleP.readString();
6.       Mary.birthDate.day=ConsoleP.readInt();
7.       Mary.birthDate.month=ConsoleP.readInt();
8.       Mary.birthDate.year=ConsoleP.readInt();
9.    }
10. }
```

Uma outra simplificação pode ser feita utilizando a classe `ConsoleO`, que oferece serviços de entrada/saída de objetos complexos. O mesmo exemplo seria assim apresentado (exemplo 3):

Exemplo 3 – Entrada da classe Student usado a classe`ConsoleO`

```
1. import ufrgs.inf.iosimple.ConsoleO;
2. class StudentTestCCO {
3.    public static void main( String args[]){
4.       Student Mary= new Student();
5.       ConsoleO.readAll(Mary);
6.    }
7. }
```

É nítida a verificação, considerando até mesmo o aspecto visual, que se obtém benefícios com o uso das classes do pacote `ufrgs.inf.iosimple`. Todo o trabalho árduo que os alunos teriam, em compreender alguns conceitos e conhecer as classes padrão da API de Java, está encapsulado nas classes "simplificadoras". Com isso, esse estudo facilita o uso de Java em disciplinas introdutórias de programação.

5.2 Tratamento de Exceções

Em Java os programadores são forçados a considerar exceções mesmo em programas muito simples. A nível introdutório pode ser complicado explicar e entender esse mecanismo.

Assim, na solução proposta, o tratamento de exceções é realizado localmente nas classes "simplificadoras" não sendo necessário, ao aluno iniciante, o conhecimento sobre exceções, poupando também o professor de ensinar os complexos mecanismos para realizar todo esse tratamento, nas primeiras aulas de um curso de introdução à programação.

5.3 Mensagens de Erro

Os erros detectados pelos ambientes de execução são exibidos de forma bastante completa, mostrando todo o estado da pilha de execução no momento da ocorrência de um erro. Por um lado essas informações são úteis aos programadores experientes, mas por outro, esse excesso de informações pode ser incompreensível aos novatos.

O ambiente JEduc, em seu módulo desenvolvido para os alunos, inclui um filtro de mensagens de erros, reduzindo o número de linhas a serem exibidas; apenas as informações pertinentes ao programa do usuário são mostradas. Já no módulo desenvolvido para os professores, é exibido todo o estado da pilha de execução, como ocorre nos outros IDEs.

5.4 Simplificações Sintáticas

Existem também alguns problemas relacionados à sintaxe de Java para os alunos iniciantes. São vários os conceitos a serem compreendidos ao mesmo tempo, começando pela declaração do método principal: `public static void main (String argv[])`. Essa construção envolve conceitos de métodos, visibilidade (`public`), métodos de classe (`static`) e instância, tipos de retorno (`void`), nomes de método (`main`), parâmetros (`argv`) e vetores (`[]`). É necessário atacar este problema por partes, inicialmente explicando o que essa declaração significa e tentando assegurar que esta assinatura de método seja sempre reproduzida de modo correto. Há também a complexidade envolvida na sintaxe das estruturas de controle de seleção e iteração, ocasionando freqüentes erros sintáticos, principalmente relacionados com os delimitadores {} de blocos de comandos e com a sintaxe do comando `switch`. Visando reduzir a ocorrência de erros e buscando facilitar o uso da linguagem, foi embutido no ambiente JEduc um menu específico para a inserção de esqueletos de código - os gabaritos sintáticos. A Figura 2 ilustra o menu Wizards do JEduc e alguns exemplos inseridos no editor e programas.

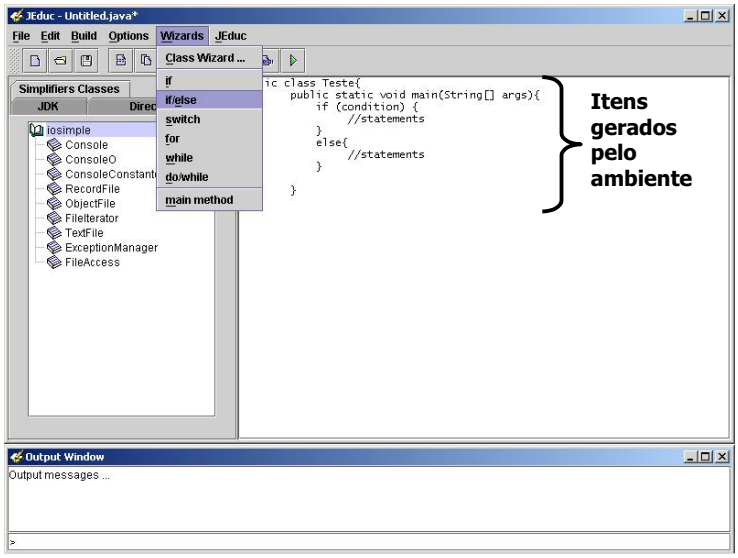

Figura 2 - Menu Wizards do JEduc

5.5 Pacotes

Devido ao grande número de pacotes disponíveis na implementação padrão da linguagem Java, alunos iniciantes sentem dificuldade para localizar informações úteis. Os novatos não compreendem claramente o limite entre a linguagem e os pacotes. A solução encontrada foi disponibilizar através do ambiente JEduc apenas os pacotes fundamentais, Figura 3, ocultando do aluno pacotes destinados a aplicações mais avançadas.

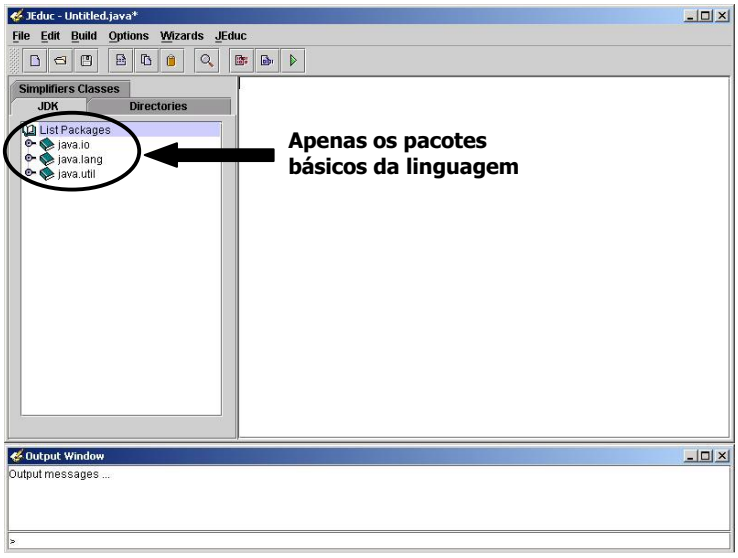

Figura 3 – Pacotes JEduc : Módulo Aluno

5.6 Auxílio aos Professores

O ambiente JEduc disponibiliza um catálogo de exemplos desenvolvido em formato HTML (Figura 4). Esse catálogo contém codificação de problemas em Pascal e em Java, apontando diferenças conceituais. Os exemplos do catálogo foram extraídos de programas de ensino de disciplinas introdutórias em diversas universidades. A intenção desse catálogo é permitir a apresentação rápida de exemplos, exercícios propostos e soluções.

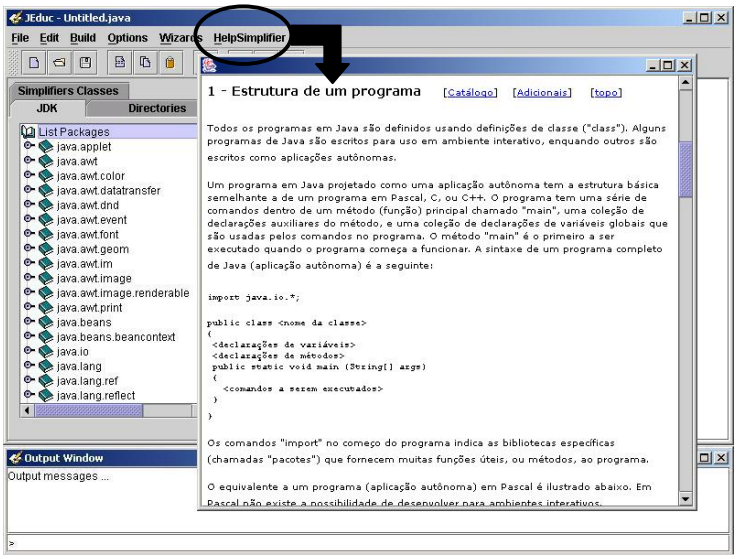

Figura 4 - Módulo Professor: Item Help

5.7 Simplificações Visuais

Alguns cuidados foram tomados com relação à excessiva quantidade de ícones e elevado número de menus, pois a grande quantidade de elementos visuais sobrecarrega a interface com elementos desnecessários.

As simplificações, Figura 5, consistem em reduzir o número de elementos visuais, existentes na maioria dos ambientes de programação, visando maximizar a usabilidade do ambiente. Dessa forma, alunos e professores podem preocupar-se apenas com o ensino/aprendizagem dos conceitos básicos de programação. Não é desejável, neste momento, perder tempo aprendendo o funcionamento do IDE.

Outro item retirado da especificação do ambiente refere-se aos projetos, permitindo que o aluno iniciante visualize apenas unidades simples de implementação.

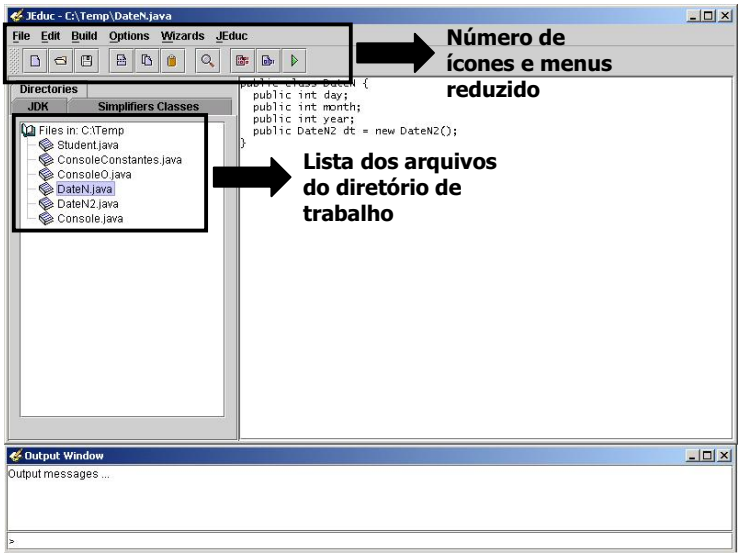

Figura 5 – Simplificações Visuais

6 Conclusões

O ambiente JEduc foi definido seguindo a filosofia de software livre, porque "... o software livre oferece um benefício mais profundo para a educação: o conhecimento embutido no software livre é conhecimento público, não secreto. A caixa preta selada de um sistema de software proprietário é projetada para manter as pessoas no escuro" [13]. Este trabalho também pretende propiciar aos alunos mais avançados na linguagem Java a possibilidade de estudar o ambiente que eles utilizam no desenvolvimento de seus programas, para aprender seu funcionamento. E assim, se necessário, escrever melhorias o integrar novas funcionalidades, gerando um software cada vez mais funcional e de melhor qualidade [4].

A principal vantagem identificada na geração do ambiente JEduc a partir do uso da IDECoRe, é o uso de um código que já foi testado, validado e está documentado, diminuindo assim a taxa de erros, falhas e o tempo para implementar o novo IDE. Esse núcleo pode ser usado pelos alunos mais avançados para experimentar novas derivações de ambientes. O ambiente JEduc encontra-se em fase de testes e está disponibilizado, juntamente com o pacote das classes "simplificadoras" e o catálogo de exemplos, em [8].

Pretende-se continuar trabalhando no IDE do JEduc, realizando várias melhorias identificadas ao longo de todo este estudo, como:

(i) um reconhecedor que identifique as palavras-chave da linguagem Java, de forma que a leitura e escrita de programas se torne mais clara e fácil.

(ii) execução parametrizada de programas Java, visto que atualmente o JEduc só permite a execução de programas que não utilizem os parâmetros passados pelo vetor `argv[]`.

(iii) implementação de um depurador.

(iv) disponibilizar a tradução de pseudocódigo em um algoritmo em Java [27].

7 Referências

[1] Bertagnolli, S. C. *Ambiente Visual para o Desenvolvimento de Aplicações Java Reflexivas*: dissertação de mestrado. Porto Alegre: PPGC da UFRGS, (Maio 2000).

[2] Bertagnolli, Silvia C., Lisbôa M. L. B., Perego C. A., Brugnara T. *Reflexão Computacional como Mecanismo de Simplificação Sintática*. Submetido ao Simpósio Brasileiro de Linguagens de Programação, SBLP, (Março 2002).

[3] Bertagnolli, Silvia C., Lisbôa M. L. B., Perego C. A. I*DECoRe: Reutilizando Componentes e Ferramentas*. Submetido ao Simpósio Brasileiro de Engenharia de Software, SBES, (Outubro 2002).

[4] Brugnara, T., Lisboa, M. L., Perego, C. A., Bertagnolli, S. C. *JEduc : uma ferramenta livre para auxiliar o ensino da linguagem de programação Java*. In: Workshop Software Livre, WSL, Porto Alegre, (Maio 2002), pp. 106-109.

[5] Deitel, H.M., Deitel P. J. *Java: como programar*. 3. ed., Porto Alegre: Bookman. 2001.

[6] GRASP. *Graphical Representations of Algorithms, Structures and Processes*, http://www.eng.auburn.edu/grasp, (Novembro 2001).

[7] Grundy, J.C. *Multi-perspective specification, design and implementation of components using aspects*, International Journal of Software Engineering and Knowledge Engineering, Vol. 10, No. 6, (December 2000), World Scientific.

[8] JEduc. *JEduc*, http://www.inf.ufrgs.br/~silviacb/JEduc, (Fevereiro 2002).

[9] JReflex. *JReflex – Informações, Links e Download*, http://www.inf.ufrgs.br/~silviacb/JReflex, (Janeiro 2001).

[10] Koffman, E., Wolz, U. *CS1 Using Java Language Features Gently,* ACM SIGCSE Technical Symposium on Computer Science Education, Vol. 31, No. 3, (Setembro 1999), pp. 40-43.

[11] Koffman, E. *Input/output for a CS1 course in Java.* In: Informatics Curricula Teaching Methods and best practice, ICTEM, Florianópolis, (Julho 2002).

[12] Kölling, M., Rosenberg, J. *Guidelines for teaching object orientation with Java*. In: Annual Conference on Innovation and Technology in Computer Science Education, Proceedings..., 6, pp. 70-74. 2001

[13] Kuhn, Bradley M. *Richard Stallman Inaugura a Fundação para o Software Livre-Índia, A Primeira Organização na Ásia Afiliada com a FSF*, http://www.gnu.org/press/2001-07-20-FSF-India.pt.html, (Julho 2001).

[14] Lea, Douglas. *Some questions and answers about using Java in Computer Science Curricula*, http://gee.cs.oswego.edu/dl/html/javaInCS.html, (Abril 2001).

[15] Lewis, J., Loftus, W. *Java Software Solutions: Foundations of Program Design*. New York:Adisson-Wesley, 2000.

[16] Lisboa, M. L., Perego, C. A., Bertagnolli, S. C. *Introductory programming: moving from Pascal to Java*. In: Informatics Curricula Teaching Methods and best practice, ICTEM, Florianópolis (Julho 2002).

[17] Mendes, A. J. N. *Software educativo para apoio à aprendizagem de programação*, http://www.c5.cl/ieinvestiga/actas/tise01/pags/charlas/charla_mendes.htm, (Dezembro 2001).

[18] Ossher, X. O. *Software Enginnering Tools and Environments: Roadmap*. New Trends in Animation and Visualization, Edited by Nadia Magnenat-Thalmann and Daniel Thalmann, John Wiley & Sons ltd., England. 2000.

[19] Pattis, R. E. *Karel the Robot*. 2. Ed. [s. l.]: John Wiley and Sons. 1995.

[20] Paula, Valéria de C. *MiCrOO: Um micro-mundo para o ensino-aprendizagem de programação orientada a objetos*, http://www.dcc.ufmg.br/pos/html/spg99/anais/valeria/valeria.html, (Março 1999).

[21] Perego, C. A, Lisboa, M. L. B. *Linguagens de Programação Adotadas nos Cursos de Computação*, http://apec.unoeste.br/~cassia/pesquisa.html, (Novembro 2001).

[22] Perego, C. A., Lisboa, M. L., Bertagnolli, S. C. *A Migração de Pascal para Java: Problemas e Propostas de Solução*. In: Workshop sobre Educação em Computação, WEI, 10, Florianópolis. (Julho 2002).

[23] Pressman, R. S. *Engenharia de Software*. São Paulo: Makron Books, 1995.

[24] Ready. *Ready to Program*. http://www.holtsoft.com/ready/home.html, (Julho 2001).

[25] Reges, Stuart. *Conservatively Radical Java in CS1*. ACM SIGCSE Technical Symposium on Computer Science Education, Vol. 32, No. 1, (March 2000), pp. 85-89.

[26] Roberts, E. *An Overview of MiniJava.* ACM SIGCSE Bulletin Conference Proceedings, Vol. 33, No. 1, (Março 2001), pp. 1-5.

[27] Souza, G. H., Pinto, M. R., Brito, M. A. S. *Uma contribuição à disciplina de introdução à programação empregando Java*, Caderno do IME - UERJ, Série Informática Nº 10, (Junho de 2001).

[28] Sziperski, C. *Component Software Beyond Object-Oriented Programming*. New York: Addison-Wesley, 1998.