Análise de Ferramentas para Apoio ao Treinamento de Linguagem de Programação Java

Roberto Silva de Oliveira Júnior

[1]Curso de Licenciatura em Ciência da Computação – Universidade Universidade Federal da Paraíba (UFPB) Campus IV - Rio Tinto - PB

roberto.silva@dcx.ufpb.br

Abstract. *This work discusses tools used for Java programming language training. The mapping performed by Henrique and Rebouças (2015) was used to identify the applications studied in this work, where filtrations were performed according to the focus of each software, the simplicity of use for didactic purposes and also the availability of access. From this filtering, four tools were selected: BlueJ, Greenfoot, Alice3 and Jeliot. Subsequently, each tool analyzed considering: interface characteristics, the main features of the tool and the available feedback method. It was also observed the coverage of these tools in support of java programming language training and object orientation. For this, example programs were tested in each environment studied. Finally, we investigated, through bibliographical research, experience reports that demonstrate the use of these environments in the classroom context.*

Resumo. *O trabalho aborda ferramentas utilizadas para treinamento de linguagem de programação Java. Foram analisadas quatro ferramentas ferramentas, sendo elas: BlueJ, Greenfoot, Alice3 e Jeliot. Cada ferramenta foi analisada considerando: características de interface, as principais funcionalidades da ferramenta* e a método de *feedback* disponível. *Também foi observada a cobertura dessas ferramentas no apoio ao treinamento de de linguagem de programação Java e orientação a objetos. Para isso, programas exemplos foram testados em cada ambiente estudado. Por fim, foram investigados, por meio de pesquisa bibliográfica, relatos de experiência que demonstram o uso desses ambientes no contexto de sala de aula.*

1. Introdução

As linguagens de programação são de grande importância atualmente, sendo necessárias para controlar o funcionamento de equipamentos eletrônicos. Por meio de seu ensino é possível desenvolver habilidades como por exemplo o raciocínio lógico e resolução de problemas. Dentre as linguagens de programação utilizadas em cursos de programação e

[1] Trabalho de conclusão de curso, sob orientação da professora Thaise Kelly de Lima Costa submetido ao Curso de Licenciatura em Ciência da Computação do Centro de Ciências Aplicadas e Educação (CCAE) da Universidade Federal da Paraíba, como parte dos requisitos necessários para obtenção do grau de LICENCIADO EM CIÊNCIA DA COMPUTAÇÃO.

orientação a objetos, tem-se Java. Essa é uma linguagem ainda amplamente utilizada pois além de ser requisitada no mercado de trabalho, é considerada uma importante plataforma de desenvolvimento. A comunidade do JAVA é ativa, o que facilita a resolução de problemas e há um grande número de *frameworks na linguagem*. Além disso, uma aplicação criada em JAVA com um mesmo código pode executar em diversos sistemas operacionais [Santana 2011]. Neste contexto, faz-se necessário saber quais ferramentas estão disponíveis e sendo utilizadas no apoio ao treinamento da linguagem de programação JAVA.

O processo de treinamento de programação no modo tradicional, utilizando primeiramente ambientes profissionais para desenvolvimento de software, pode requisitar uma alta capacidade de abstração e dificultar o treinamento da linguagem em um momento inicial, pois alguns alunos necessitam de ferramentas que possuam ludicidade, contemplando por exemplo geração de animações, jogos e diagramas. Nesse contexto, é interessante o uso de ferramentas que facilitam o apoio ao treinamento, como os ambientes de desenvolvimento integrado (*Integrated Development Environment*- IDE), que também são ambientes educacionais e abordam um paradigma textual e visual, com objetivo de tornar as atividades mais didáticas.

Este trabalho tem por foco investigar diferentes ambientes de programação educacional e caracterizá-los em termos de utilização, de acordo com suas funcionalidades e formas de disponibilizar *feedback* para o usuário, a fim de ajudar professores e alunos na escolha de ferramentas para apoio ao treinamento da linguagem de programação Java. Os ambientes alvo deste estudo são Bluej, Greenfoot, Alice3 e Jeliot. Estes ambientes foram escolhidos com base no trabalho realizado por Henrique e Rebouças (2015), que identificou Objetos de Aprendizagem para auxiliar o ensino de conceitos do Paradigma de Programação Orientada a Objetos. A partir dessa listagem de objetos educacionais, foram selecionadas ferramentas com foco no treinamento de programação Java, de uso simples para fins didáticos e que tivessem disponibilidade de acesso. Assim, das quinze ferramentas citadas no mapeamento de Henrique e Rebouças (2015), quatro foram incluídas neste estudo.

O trabalho encontra-se dividido em seis seções que estão organizadas da seguinte forma: na seção 2 é demonstrada a metodologia do trabalho; na seção 3 são apontadas ferramentas para o treinamento de JAVA; na seção 4 é demonstrado um resumo de características das ferramentas encontradas; na seção 5 são apresentadas experiências de uso; e na seção 6 são apresentadas as considerações finais.

2. Metodologia

No mapeamento realizado por Henrique e Rebouças (2015) foram identificadas ferramentas para ensino de Programação Orientada a Objetos, algumas com foco na linguagem de programação JAVA. Com base nesse mapeamento e considerando a importância do apoio ao treinamento da linguagem de programação Java nos cursos de Computação, foram selecionadas para estudo as ferramentas que possuem o foco nessa linguagem, considerando simplicidade de utilização para fins didáticos e disponibilidade de acesso. Para identificar essas características nas aplicações, foi necessário realizar buscas por artigos que trabalharam com as ferramentas destacadas pelos autores,

websites das aplicações para identificar se atualmente estão disponíveis e, quando existia acesso aos *softwares*, utilizá-los de fato com o objetivo de validar a simplicidade de utilização.

A partir dessa filtragem, foram selecionadas quatro ferramentas para apoio ao treinamento da linguagem de programação JAVA, sendo elas: BlueJ, Greenfoot, Alice3 e Jeliot. Cada ferramenta foi testada considerando:

- Análise das características de interface: área de visualização do programa em execução, área de implementação do código, área de visualização do diagrama gerado, controles para execução de animação
- As principais funcionalidades da ferramenta;
- Formato de feedback;

Após o uso e identificação dessas características, também foi observada a cobertura dessas ferramentas no apoio ao treinamento de assuntos relacionados ao treinamento de linguagem de programação e orientação a objetos. Para isso, programas exemplos foram testados em cada ambiente estudado.

Além disso, foram investigados, por meio de uma pesquisa bibliográfica, relatos de experiência que demonstram o uso desses ambientes no contexto de sala de aula, a fim de que, por meio de exemplos, outros docentes possam direcionar possíveis aplicações didáticas de utilização.

3. Ferramentas para apoio ao treinamento de JAVA

3.1 BlueJ

É um IDE que possui funcionalidades básicas para o desenvolvimento de *software*. Ele apresenta uma interface simples que tenta facilitar a compreensão do código escrito por meio de uma abordagem visual com digrama de classes, como forma de organizar e facilitar o entendimento das interações entre cada componente do projeto.

De acordo com Vahldick (2007), apesar do Bluej não possuir recursos de autocompletar ou assistentes que facilitam a codificação, é presente o recurso de criar objetos usando código JAVA e depois instanciá-los sem a necessidade de escrever mais código, utilizando para isso funcionalidades visuais em forma de item de menu ao clicar na classe desejada no diagrama. Na Figura 1 é possível visualizar dentro da área 4 um objeto criado sem a necessidade de escrita de código, sendo ele demonstrado em forma de cartão.

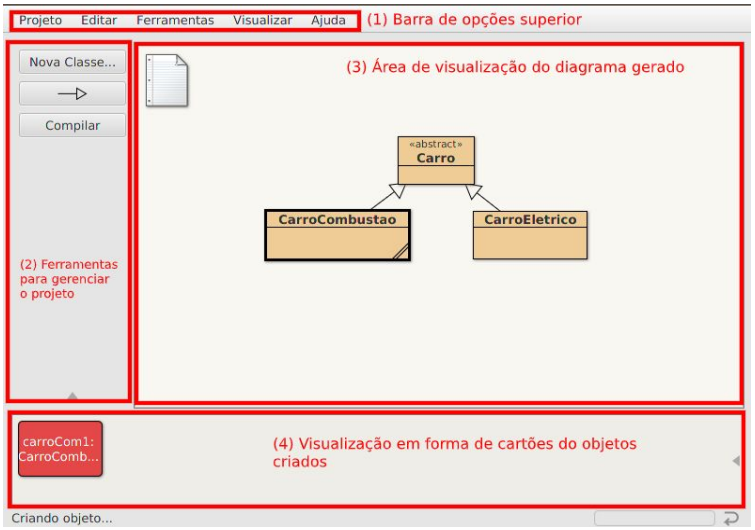

Figura 1. Ambiente do BlueJ[2] ao selecionar um objeto criado

O BlueJ está disponível atualmente para Windows, Linux, MacOs ou qualquer outro sistema com máquina virtual Java instalada. Esse fato o torna acessível a vários usuários. O site da aplicação[2] dispõe de outros materiais como livros sobre JAVA e área dedicada para professores, na qual é possível ter acesso a recursos para ensino e se comunicar com outros educadores. Além disso, o site também dispõe de uma documentação da ferramenta na qual é possível encontrar tutoriais em vídeo e em texto, material para estudo de testes unitários, trabalho em equipe usando o Git, tutoriais para o Raspberry Pi, extensões e suporte técnico.

3.1.1 Análise das principais funcionalidades

O BlueJ possui a funcionalidade de geração automática de comentários no código JAVA. Por exemplo, quando se cria uma classe dentro do Bluej, são gerados vários comentários que indicam o que se deve realizar em cada parte do arquivo: a área de descrição da classe, onde as variáveis devem ser criadas (Figura 2). Esses comentários podem ser de grande ajuda para estudantes que possuem dificuldade para se adaptar à linguagem de programação JAVA, servindo como lembrete para conceitos trabalhados ou como guia em um primeiro contato.

```
/**
 * Escreva a descrição da classe CarroCombustao aqui.
 *
 * @author (seu nome)
 * @version (número de versão ou data)
 */
public class CarroCombustao extends Carro
{
    // variáveis de instância - substitua o exemplo
    //abaixo pelo seu próprio
```

Figura 2. Ambiente do BlueJ ao selecionar uma classe

[2] "BlueJ " (https://www.bluej.org/) Acessado em 10 de mar. de 2020.

Também é presente a funcionalidade de destaque de blocos de código, onde é possível identificar, por exemplo, onde um método começa e onde termina, a partir da cor em destaque. Além disso, é possível identificar o tipo do bloco de código através das cores usadas, onde as classes são destacadas em verde claro e os métodos em amarelo, o que pode ser um fator facilitador para estudantes iniciantes (Figura 3).

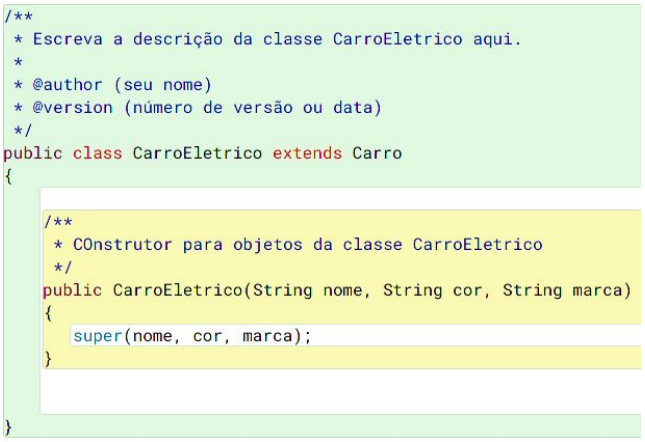

Figura 3. Ambiente do BlueJ com destaque diferente para classe e método

Outro recurso, já comentado anteriormente, é a produção de diagramas de classes de forma automática, a partir dos arquivos criados em um projeto, o que pode ser um fator para melhor entendimento do relacionamento entre elementos do projeto.

3.1.2 Análise de *Feedback*

O BlueJ oferece *feedback* a partir da declaração de erros de execução aos seus usuários. Esse *feedback* ocorre quando são detectados problemas nas classes compiladas, demonstrado por meio de destaque com a cor vermelha na linha onde o problema foi encontrado. Esse formato de *feedback* é semelhante ao que ocorre com IDEs profissionais, não apresentando formas alternativas para identificação e visualização dos erros de execução por usuários iniciantes, como por exemplo execução do código a partir de animações para auxiliar na depuração do erro. No entanto, por meio da funcionalidade de geração automática de diagramas de classes é identificado outro tipo de *feedback*, no qual é possível visualizar erros de relacionamento entre os elementos do projeto através das ligações existentes no diagrama.

3.1.3 Conteúdos suportados pelo BlueJ

O BlueJ é um ambiente bem completo no sentido do suporte aos conteúdos. A partir dele é possível trabalhar com vários tipos de assuntos com linguagem de programação JAVA, sendo identificados: classes abstratas, objetos, polimorfismo, herança, criação e invocação de métodos.

3.1.3.1 Classes Abstratas

O BlueJ possibilita a criação de classes abstratas, onde é possível criar subclasses a partir da classe mãe, e posteriormente utilizar seus métodos e atributos através da

interface visual. Na Figura 4, é demonstrado um exemplo de implementação de uma classe abstrata chamada de Carro, que possui três atributos e um método. As classes que forem filhas dela, poderão ter acesso ao seu conteúdo.

```
public abstract class Carro
{
    // variáveis de instância - substitua o exemplo abaixo pelo seu próprio
    protected String nome;
    protected String cor;
    protected String marca;
    /**
     * COnstrutor para objetos da classe Carro
     */
    public Carro(String nome, String cor, String marca)
    {
        this.nome = nome;
        this.cor=cor;
        this.marca=marca;
    }

    public String getStatus(){
        return "Nome: "+this.nome+" Cor: "+this.cor+" Marca: "+this.marca;
    }
```

Figura 4. Editor do BlueJ criando uma classe abstrata

3.1.3.2 Polimorfismo

A partir do diagrama gerado é possível criar objetos da classe selecionada com o botão direito e clicando no item de menu “new NomeDaClasse”, assim um novo objeto é criado e representado em forma de cartão, como é demonstrado na Figura 5. A classe CarroCombustao é uma classe filha da superclasse Carro, pois ela estendeu a classe Carro. Assim, pode-se perceber o suporte ao polimorfismo, pois a classe referente ao objeto representado na Figura 5 redefiniu o método declarado na sua classe mãe.

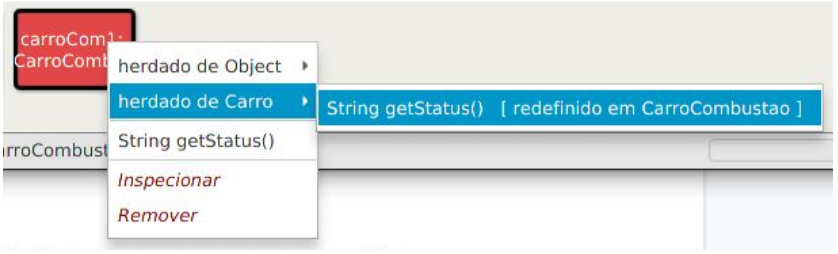

Figura 5. Gerenciando objetos de forma visual com o BlueJ

3.1.3.3 Construtores e Instanciação

O BlueJ disponibiliza a funcionalidade de instanciação de objetos sem a necessidade da criação de código JAVA, de forma a facilitar a abstração de como são utilizados os construtores e criadas instâncias dos objetos, utilizando para isso recursos visuais disponíveis na interface do usuário. Em momentos iniciais do processo de treinamento, este método disponibilizado pelo BlueJ se demonstra mais didático. Após a adaptação dos estudantes o professor pode comparar em sala de aula a forma como objeto é instanciado com código em JAVA e como era feito anteriormente. A Figura 6 é um exemplo de como esta opção pode ser utilizada através da instanciação de um objeto do tipo CarroCombustao.

Figura 6. Instanciando uma classe de forma lúdica

3.1.3.4 Classes, objetos e herança

A partir do BlueJ, usuários podem visualizar classes e entender o conceito de herança por meio dos diagramas. Por exemplo, por meio das Figuras 7 e 8 é possível verificar que as classes *CarroCombustao* e *CarroEletrico* herdam os atributos (nome, cor e marca) e o método (getStatus) presente dentro da classe abstrata *Carro*. Ambas as classes foram geradas no editor do BlueJ e estão representadas no diagrama abaixo.

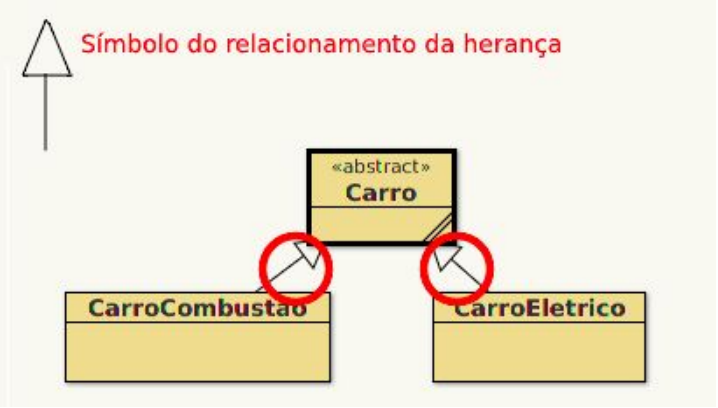

Figura 7. Diagrama de classes gerado pelo BlueJ a partir de código JAVA

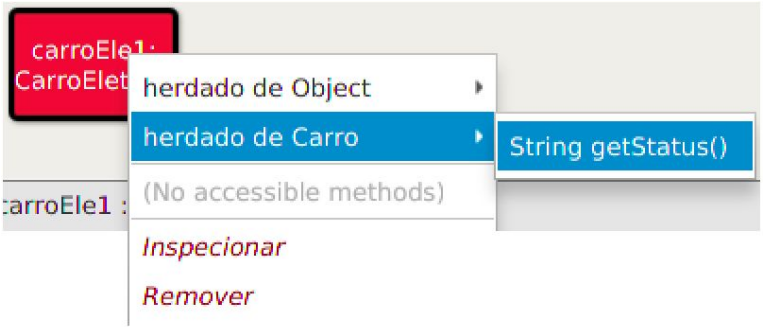

Figura 8. Visualizando métodos herdados da classe Carro

3.1.3.5 Criação e Invocação de métodos

A criação de métodos é feita de forma textual pelo editor (Figura 9), já a invocação é realizada de forma visual quando se clica com o botão direito do mouse no cartão do objeto criado, como demonstrado na Figura 10. Na invocação, os métodos são listados para seleção, indicando o tipo do retorno e o nome do método.

```
public String getStatus(){
    return "Nome: "+this.nome+" Cor: "+this.cor+
    " Marca: "+this.marca
    +"Tipo Combustivel: "+this.tipoCombustivel;
}
```

Figura 9. Criando um método com o BlueJ

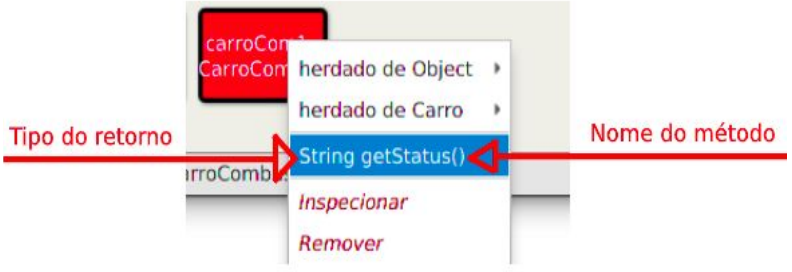

Figura 10. Invocando um método com o BlueJ

3.2 Greenfoot

De acordo com Henriksen e Kölling (2004 apud Vahldick 2007) o Greenfoot é um framework e um ambiente para gerar aplicações interativas e simulações em um plano bidimensional (2D). Ele possui como proposta abordar orientação a objetos e ao mesmo tempo a linguagem de programação JAVA, usando o apelo visual e interativo a fim de tornar o aprendizado mais lúdico. O ambiente do Greenfoot pode ser observado na Figura 11.

Figura 11. Ambiente do Greenfoot[3]

A aplicação dispõe de ferramentas para visualizar e interagir com ambientes e atores pré-inseridos e, assim, gerar animações utilizando código Java escrito no editor da ferramenta. O *website* desta ferramenta possui tutoriais em forma de vídeo, texto e uma documentação online. É possível publicar projetos no site oficial da ferramenta, para que outros usuários possam utilizá-los direto do navegador de internet, além de poder participar de fóruns de dúvidas. Ela está disponível para dispositivos com Windows, Mac OS X, Ubuntu e dispositivos que rodam arquivos do tipo ".JAR". De acordo com Vahldick (2010), o Greenfoot utiliza o BlueJ como ferramenta de edição de código.

3.2.1 Análise das principais funcionalidades

O salvamento automático é uma das principais funcionalidades do Greenfoot, isto impede por exemplo que alterações sejam perdidas ao fechar a janela da aplicação. Além disso a aplicação possibilita a criação de animações e dispõe da funcionalidade de controlá-las usando funções parecidas com as encontradas em *players* de vídeo, podendo pausar, reiniciar e aumentar a velocidade de execução da animação. Ainda existe a funcionalidade de compartilhar os projetos criados, permitindo o uso por outros usuários, que poderão executar os trabalhos pelo site da aplicação ou em seus computadores pessoais.

O Greenfoot possui o editor do BlueJ incorporado à ferramenta, incorporando características que facilitam a interação com os objetos visuais que compõem a animação, como por exemplo, miniatura do objeto trabalhado ao lado do nome da classe, além do autoimport da biblioteca do Greenfoot. Esta biblioteca possui vários métodos para trabalhar com os objetos que compõem a animação, controlando movimento, posição e tamanho.

3.2.2 Análise de Feedback

O Greenfoot, assim como o BlueJ, também possui como um dos meios de *feedback* a declaração de erros de execução quando as classes são compiladas, e assim são

[3] "Greenfoot " (https://www.greenfoot.org/door) Acessado em 10 de mar. de 2020.

demonstrados *warnings* e *erros* explicando os problemas encontrados. O diferencial do Greenfoot é sua área de representação visual do código, enquanto no BlueJ são mostrados diagramas de classes, o Greenfoot cria uma pré-visualização da animação que será criada. Esta visualização permite ao aluno ir do abstrato ao concreto, fazendo com que ele visualize imagens representativas das classes criadas, em vez de cartões encontrados nos diagramas de classes.

3.2.3 Conteúdos suportados pelo Greenfoot

3.2.3.1 Classes, objetos e herança

O Greenfoot permite a criação e instanciação de classes de forma visual, sendo possível visualizar o código JAVA escrito e o objeto declarado na forma de imagem. Na Figura 12, é mostrada uma imagem representando a declaração de um objeto do tipo Peixe e a representação em código pode ser observada à direita da imagem (2 - código fonte do ator). Já o objeto de forma visual é apresentado à esquerda (1- representação do ator).

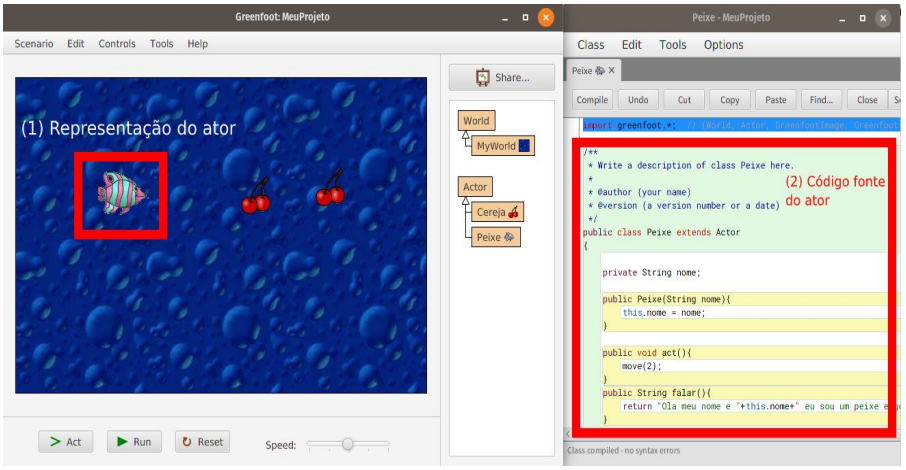

Figura 12.Ilustração da associação entre objeto visual e em código no ambiente Greenfoot

A interface do usuário do Greenfoot, permite realizar várias ações, sendo uma delas é a criação de subclasses. Na Figura 13 é possível observar como o recurso de criação de subclasse está disponível. Ao clicar na classe que deseja ser a superclasse (com o botão direito) são dispostas várias opções, sendo uma delas a "Nova subclasse...". Ao selecionar esta opção, o usuário é encaminhado para área de criação de classes padrão, mas com a diferença que o Greenfoot irá estender automaticamente a classe mãe na subclasse criada. O usuário pode acompanhar o código Java para verificar o que esta mudança representa no código e verificar que a nova subclasse herda todos os atributos e métodos da classe mãe.

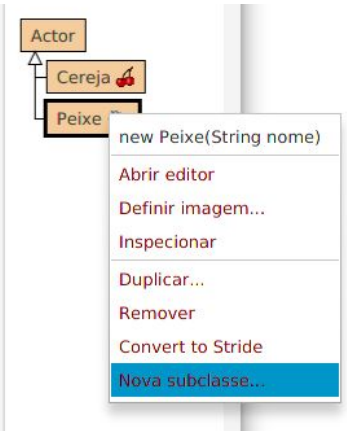

Figura 13. Criando subclasse no Greenfoot

Para instanciar novos objetos é preciso selecionar a classe desejada no diagrama, e ao clicar sobre a ela são disponibilizadas várias opções, sendo uma delas a de instanciação de novos objetos da classe selecionada. Na Figura 14 é possível visualizar um exemplo de instanciação de objetos no Greenfoot. Logo após clicar nesta opção do menu, um novo objeto em forma visual é criado na tela e o usuário deve posicioná-lo no ambiente declarado.

Figura 14. Instanciando um objeto de forma lúdica

3.2.3.2 Criação e Invocação de métodos

Como o Greenfoot utiliza o BlueJ como editor de código, ele tem suporte para criação de métodos, mas possui um diferencial: a possibilidade de invocar os métodos de forma visual. Para invocar um método de forma visual no Greenfoot, é necessário ir à área de visualização da animação e clicar com o botão direito no objeto visual que deseja invocar o método. O Greenfoot apresentará a lista de métodos que o objeto possui. Na Figura 15 é possível observar como os métodos podem ser invocados de forma visual para o exemplo de um objeto do tipo Peixe, podendo ter acesso ao método falar.

Figura 15. Invocando método no Greenfoot

3.3 Alice 3

Nesta ferramenta é possível criar animações e jogos 3D, com o objetivo de serem facilitadores da aprendizagem. Ela aborda a linguagem e lógica de programação podendo ser utilizada como ferramenta para o processo de treinamento de JAVA. Nela, a programação pode ser realizada por meio de blocos e visualizada automaticamente em JAVA (Figura 16).

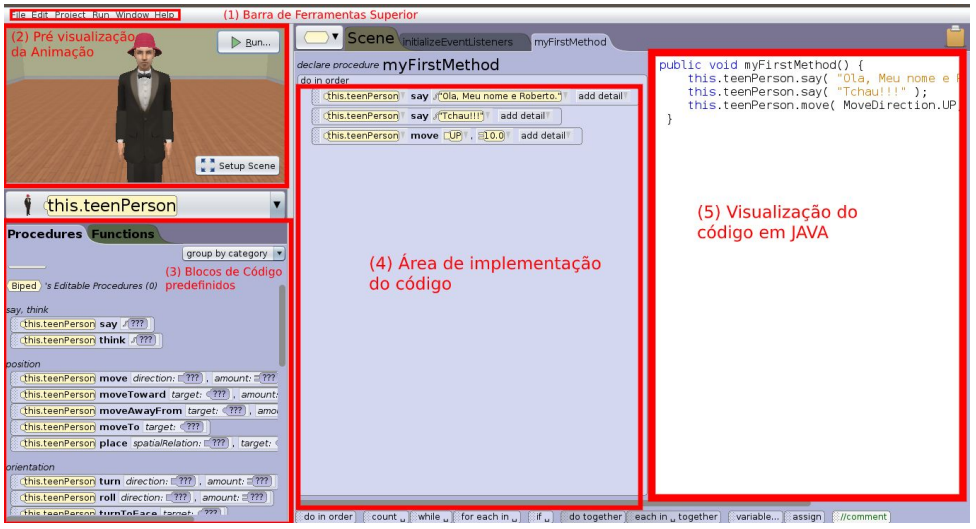

Figura 16. Ambiente do Alice[4]

[4] "Alice " (http://www.alice.org/get-alice/alice-3/) Acessado em 10 de mar. de 2020.

No trabalho de Valaski e Pairaiso (2012), foi avaliado o uso do Alice durante a disciplina intitulada de “Oficina de Programação” que é ministrada para uma turma do primeiro período do curso de Sistemas de Informação. Como forma de avaliar os efeitos gerados pelo uso do Alice, foram realizadas coletas de opinião com os alunos. E a partir dessa pesquisa o autor afirma que a aplicação é melhor indicada para estudantes com dificuldades em conteúdos teóricos.

3.3.1 Análise das principais funcionalidades

A pré-visualização da animação é uma das principais funcionalidades do Alice3, pois o foco aqui é criar animações e jogos tridimensionais. Dessa forma, o apelo visual se torna muito importante. A codificação nesta ferramenta utiliza uma forma visual de se programar, utilizando blocos. A partir dos algoritmos criados de forma visual, são gerados códigos em Java. Na ferramenta, é possível adicionar ou editar objetos e ambientes de forma visual através da interface do usuário. A ferramenta dispõe de métodos pré-inseridos, mas é possível criar novos.

3.3.2 Análise de Feedback

O *feedback* no Alice3, é observado por meio do jogo ou animação criada, ou seja, é visualizado através do resultado da codificação. Pela programação ser realizada através de blocos, erros de sintaxe são mais difíceis de acontecer, o que possibilita focar na lógica do programa e posteriormente visualizar como seria escrever o mesmo código na linguagem de programação JAVA.

3.3.3 Conteúdos suportados pelo Alice 3

3.3.3.1 Criação e invocação de métodos

A criação de métodos é realizada através de blocos de código, que são adicionados e instanciados automaticamente no código JAVA existente na classe “Scene”. Na Figura 17, é possível visualizar como um método é criado no Alice3.

Figura 17. Criando um método em Alice3

3.3.3.2 Objetos e Classes

Assim como os métodos são criados de forma visual, os objetos e classes também são, e logo após a sua criação elas são geradas na classe “Scene” na linguagem de programação JAVA. Na figura 18, é possível visualizar como o ambiente, a câmera e um ator é instanciado no Alice3.

```
/* Scene fields */
private final SRoom room = new SRoom();
private final SCamera camera = new SCamera();
private final TeenPerson teenPerson = new TeenPerson(
```

Figura 18. Instanciando objetos em Alice3

3.4 Jeliot

É uma ferramenta que utiliza animações para demonstrar a execução do programa escrito, onde uma mão em forma de desenho move cartões que simbolizam os elementos declarados no programa. As variáveis e seus valores são movimentadas entre quatro áreas na tela que tem como objetivo simular a execução de métodos, avaliação de expressões, constantes, instâncias e arrays.

As animações são geradas de forma automática e, de acordo com Tori e Mainente (2001), com uso do Jeliot não é preciso escrever linhas adicionais de código para gerar a animação, pois essas rotinas da animação estão encapsuladas nas operações dos tipos de dados. O editor padrão é limitado, considerando o uso didático, pois não apresenta meios de destaque do código, mas é possível utilizá-lo a partir do BlueJ.

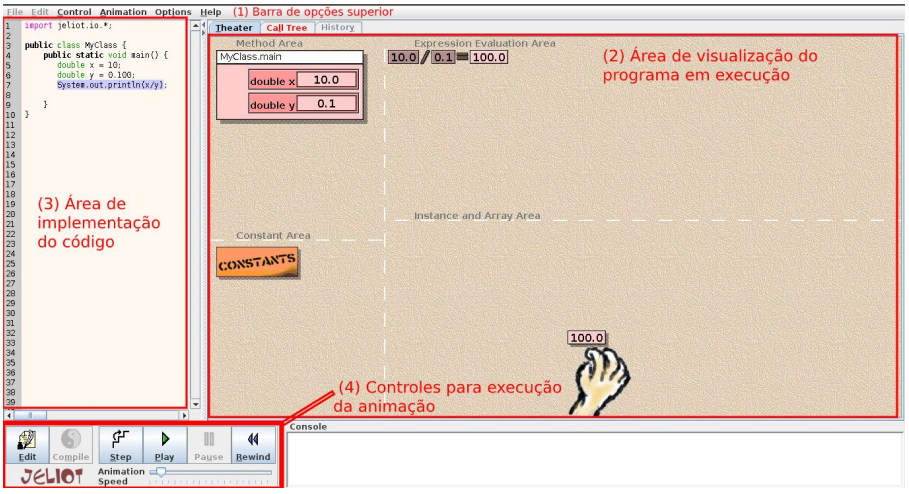

Figura 19. Ambiente do Jeliot[5]

Em Silva et al (2011) são demonstradas práticas realizadas com o objetivo de apoiar o processo de ensino e aprendizado de algoritmos, e para isso foram utilizados vários instrumentos para auxiliar na execução da atividades. Um desses instrumentos foi o Jeliot, buscando reduzir a abstração computacional dos algoritmos criados pelos alunos. Assim, o Jeliot foi apresentado para os alunos como o seu primeiro editor de código e todas as atividades práticas o utilizaram. Foi relatado que no início os alunos sentiram dificuldade pois o *software é de língua inglesa,* mas que com o passar do tempo foram se adaptando. O trabalho também demonstra que o nível de detalhes da

[5] "Jeliot " (http://cs.joensuu.fi/jeliot/) Acessado em 10 de mar. de 2020.

simulação da execução do código se mostrava tão elevado que em alguns casos gerava confusão para os alunos.

Para analisar o uso da aplicação tomando como referência os estudantes, Silva et al (2011) aplicaram um questionário para um grupo de 22 alunos. O resultado apontou que 95% dos alunos consideram o Jeliot útil para visualizar o que acontece no algoritmo, que 59% dos estudantes preferem utilizar o Jeliot no lugar de outros editores, 81% da turma considera a simulação criada pela ferramenta suficientemente clara para entender o seu algoritmo, por fim aponta que 68% do participantes afirmam que o JEliot ajuda a entender melhor o funcionamento de algoritmos sempre ou quase sempre . De acordo com Silva et al (2011), o fato de 31% do participantes preferirem utilizar outros editores aconteceu pois se acredita que as facilidades disponibilizadas por outros editores como recursos de autocompletar e identificação de erros durante a codificação influenciaram a escolha.

3.4.1 Análise das principais funcionalidades

Uma das principais funcionalidades do Jeliot é a execução da animação gerada pelo código escrito em JAVA. Para que ocorra a execução das animações, é necessário a importação da biblioteca "jeliot.io*", e ela é importada de forma automática pelo editor ao compilar as classes. Este editor possui menos funcionalidades que o editor encontrado nas ferramentas BlueJ e Greenfoot, pois não possui visualização dos blocos de código criados utilizando o tratamento por cores para destacar o tipo do bloco, nem possibilidade de trabalhar com mais de um arquivo por vez.

3.4.2 Análise de Feedback

Assim como o Bluej e o Greenfoot, o Jeliot demonstra erros de execução quando as classes são compiladas, e assim são apresentados *warnings* e *erros* encontrados no programa desenvolvido, mas com o diferencial da forma como o erro é apresentado. No Jeliot, ocorre a interrupção da animação e então o erro é apresentado, deixando o seu *feedback* de erros bem diferente dos encontrados em IDEs profissionais. Assim como encontrado no Greenfoot, o Jeliot também possui uma representação visual do código gerado, mas essa é mais indicada para estudantes que possuem um nível mais básico da linguagem de Programa JAVA, já que é demonstrado o processo de execução do código, ou seja, é importante para a compreensão do algoritmo criado, enquanto por exemplo o Greenfoot demonstra uma ilustração que aborda orientação a objetos, e este é um tema mais avançado. Essa forma de observar os resultados obtidos trabalha a habilidade de tirar erros do código devido a visualização de execuções inesperadas pelo usuário.

3.4.3 Conteúdos suportados pelo Jeliot

3.4.3.1 Criação e invocação de métodos

O Jeliot possui suporte a Orientação a Objetos, no entanto possui a limitação de trabalhar somente com um único arquivo, o que pode dificultar a realização de atividades devido ao excesso de linhas de código. Na figura 20, é possível visualizar um método do tipo estático sendo criado no jeliot.

```
public class MyClass {
    public static void main() {
        System.out.println(somar(2,5));

    }
    public static double somar(double x,
    double y){
    return x+y;
    }
}
```

Figura 20. Método estático criado no Jeliot

4. Resumo de características

A partir do estudo realizado com as ferramentas BlueJ, Greenfoot, Alice3 e Jeliot foi possível realizar um comparativo em termos de funcionalidade, *feedback*, pontos positivos e negativos encontrados em cada ferramenta para apoio ao treinamento de linguagem de programação JAVA. O Quadro 1 apresenta o resultado da análise comparativa das ferramentas.

Quadro 1. Resumo de características encontradas nas ferramentas para apoio ao treinamento de JAVA

Nome	Funcionalidades	Feedback	Pontos Positivos	Pontos Negativos
BlueJ	• Geração automática de comentários; • Destaque nos blocos de código; • Geração automática de diagrama de classes a partir de código JAVA.	• Declaração de erros de execução após compilar classes; • Identificar erros através do relacionamento entre os elementos representados no diagrama de classes.	• Mais simples que IDEs profissionais; • Utiliza elementos visuais ao instanciar objetos;	• Alunos iniciantes podem não possuir conhecimento sobre diagrama de classes, o que pode diminuir a eficácia da ferramenta em relação a outras com

				diferentes abordagens.
Greenfoot	• Utiliza da ludicidade ao instanciar objetos; • Criação de animações através de código JAVA; • Controle da execução das animações criadas; • Compartilhar projetos na internet para outros usuários; • Autoimport da biblioteca do Greenfoot, onde são disponíveis métodos para operar com ambientes e atores no projeto;.	• Declaração de erros de execução após compilar classes; • Pré visualização da animação que está sendo criada, aborda a ludicidade de forma mais eficaz que digrama de classes.	• Mais simples que IDEs profissionais; • Utiliza da ludicidade ao criar e instanciar objetos; • Feedback visual e com mensagens textuais. • Aborda assuntos de Orientação a Objetos.	• Limitação na criação de classes, pois só é possível instanciar objetos das classes World e Actor.
Alice3	• Programação realizada com blocos e posteriormente traduzido para JAVA; • Geração de animações ou jogos tridimensionais, utilizando para isso mecanismos visuais e programação em blocos;	• Observado por meio do jogo ou animação criada;	• Pela programação ser realizada através de blocos os erros de sintaxe são evitados, o que o difere das demais ferramentas; • Aborda assuntos de Orientação a Objetos.	• Só ser possível codificar com programação em blocos pode ser um fator limitante no aprendizado, seria interessante possuir a funcionalidade de programar em JAVA e posteriormente ocorrer a tradução para blocos.
Jeliot	• Criação de animações demonstrando a execução do código JAVA que foi criado; • Controle de execução da animação criada a partir do código escrito; • Possibilidade de utilizar a IDE, BlueJ,	• Declaração de erros de execução após compilar classes; • A partir da animação gerada é possível identificar erros.	• Declaração de erros de execução de forma mais lúdica do que é encontrado no BlueJ e Greenfoot, parando a execução da animação para	• Editor padrão com poucas funcionalidade s.

	como editor de código.		demonstrar o erro.	

5. Experiências de uso

Neste trabalho, foi feita uma revisão bibliográfica em busca de relatos de experiência sobre o uso do BlueJ e Greenfoot, pois após os estudos realizados, elas demonstraram possuir maior quantidade de conteúdos e funcionalidades. Além disso demonstram o *feedback* para o usuário de forma lúdica sem perder o foco na linguagem de programação Java.

5.1 BlueJ

O trabalho relatado por Vahldick (2007), demonstra a condução de uma disciplina de Programação Orientada a Objetos com Java. Ele utilizou como proposta didática aliar aulas com objetos didáticos em sala de aula com o uso do BlueJ e Greenfoot no laboratório. Em sala de aula por exemplo, utilizou caixas de sapato confeccionadas com tiras de papel para anotação das características dos objetos criados, onde na tampa são declarados os nomes de cada objeto e na lateral a lista de métodos. Além disso, dentro da caixa existiam outros papéis onde foram declarados os atributos dos objetos. Assim, nas primeiras atividades os estudantes tiveram a missão de ler o código fonte em JAVA e depois representar os objetos encontrados nas caixas disponibilizadas anteriormente pelo professor.

Após esse período foi utilizado o BlueJ como extensão do que foi ensinado em sala de aula, pois foi necessário um ambiente que mostrasse como e quando os objetos foram instanciados, métodos utilizados e atributos acessados. Durante este processo foi constatado que os alunos que obtiveram melhores resultados iniciaram a implementação de novos métodos somente quando o atual não exibisse erros de compilação. Além disso, de acordo com Vahldick (2007) o fato do BlueJ não possuir funcionalidade de automatização na edição do código, como por exemplo autocompletar, exige mais atenção e compreensão na produção de classes.

5.2 Greenfoot

Nas atividades descritas em Vahldick (2007), além do BlueJ também foi utilizado o Greenfoot. Para a realização das atividades foram disponibilizados arquivos compactados com projetos não concluídos para os alunos. Cada projeto continha um enunciado e mais figuras para serem associadas aos atores. Além disso, em alguns haviam classes de atores e de mundo prontas.

Com o objetivo de ambientação com a ferramenta e com o framework, as primeiras atividades utilizando Greenfoot para adaptar exercícios criados na primeira unidade que foram criadas utilizando o BlueJ. Assim os alunos transformaram as classes criadas anteriormente, em atores. A medida que as aulas foram passando o nível de

dificuldade foi aumentando tendo com métrica a diminuição de recurso prontos. Em certa etapa os projetos só tinham o enunciado para a execução do projeto. Como forma de avaliação da aprendizagem, foram realizadas duas verificações e um trabalho final onde os alunos tiveram a missão de implementar o jogo Pacman.

Em outro trabalho, Santos et. al (2017) trata do processo de ensino e aprendizado de orientação a objetos utilizando ferramentas com abordagens lúdicas. Uma das ferramentas trabalhadas por este estudo foi o Greenfoot. A experiência apresentada por Santos et al. (2017) ocorreu nos moldes de oficina.

Essa oficina foi planejada para que alunos escrevessem pequenos programas orientados a objetos. Ela foi dividida em três momentos e em cada momento eram trabalhados conteúdos técnicos disponíveis na linguagem de programação JAVA como por exemplo objetos, classes, atributos, métodos, mensagens, herança e polimorfismo. Durante a oficina, foi possível verificar o comportamento dos estudantes ao realizar a transição entre o Greenfoot, que é uma IDE visual, para o NetBeans, que é uma IDE profissional. Assim foi verificado que dúvidas frequentes existiram ao utilizar o ambiente profissional, mas que essa experiência foi válida pois os alunos tiveram a oportunidade de visualizar os conceitos aprendidos de forma visual no Greenfoot aplicados de forma prática e usual por profissionais. O trabalho afirma como alguns dos resultados encontrados que o uso de competições deixam as atividades mais excitantes para os estudantes, além de que a oficina realizada foi um instrumento motivador e introdutório para o aprendizado de POO.

6. Considerações Finais

O presente trabalho apresentou uma análise de ferramentas para treinamento de linguagem de programação JAVA. As aplicações analisadas surgiram a partir de um filtro dos Objetos de Aprendizagem citados por Henrique e Rebouças (2015). Para que ocorresse a filtragem dos resultados obtidos foi necessário selecionar as ferramentas têm foco no treinamento de programação Java, são de uso simples para fins didáticos e que tivessem disponibilidade de acesso. Para identificar essas características nas aplicações foi necessário realizar buscas por artigos que trabalharam com as ferramentas destacadas, websites das aplicações para identificar se atualmente estão disponíveis e, quando existia acesso aos softwares, utilizá-los de fato com o objetivo de validar a simplicidade de utilização.

Os resultados da análise foram satisfatórios, demonstrando que as ferramentas BlueJ e Greenfoot demonstram possuir maior quantidade de conteúdos e funcionalidades. Já no caso do Alice3 e Jeliot, é presente um feedback mais elaborado e lúdico, contando o primeiro com a possibilidade de criação de animações e Jogos tridimensionais e o segundo com a animação de como o código JAVA é compreendido. As quatro ferramentas se mostraram interessantes para uso introdutório de linguagem de programação e orientação a objetos. Elas, podem ser usadas até mesmo para treinamento de lógica de programação, antes mesmo do estudo da linguagem Java propriamente dita, o que as tornar boas ferramentas para abordar durante monitorias de disciplinas de programação.

Agradecimentos

Para desenvolver este trabalho de conclusão de curso foi necessária a ajuda de diversas pessoas, às quais expresso uma imensa gratidão por cada uma delas, são elas:

A banca avaliadora, composta pelas professoras Ayla Débora Dantas de Souza Rebouças e Izabelly Soares de Morais e principalmente à minha orientadora Thaíse Kelly de Lima Costa, por sua paciência e sugestões para elaboração deste trabalho.

Referências

HENRIQUE, Mychelline Souto; REBOUÇAS, Ayla Débora Dantas Souza. (2015). "Objetos de Aprendizagem para auxiliar o ensino de conceitos do Paradigma de Programação Orientada a Objetos". RENOTE-Revista Novas Tecnologias na Educação, v. 13, n. 2, 2015.

SANTANA, Otávio Gonçalves.(2011)." Por que java?". Disponível em: <https://www.devmedia.com.br/por-que-java/20384>. Acesso em: 22 nov. 2019.

SANTOS, Cleison Simoes, et al. (2017). "Aprendendo Programação Orientada a Objetos com uma abordagem lúdica baseada em Greenfoot e Robocode". arXiv preprint arXiv:1710.04132.

SILVA, Gabriel Costa et al. (2011). "Uma experiência na aplicação de práticas de apoio no ensino-aprendizado de algoritmos". In: Anais do Workshop de Informática na Escola. 2011. p. 1378-1381.

TORI, Romero; MAINENTE, Cilene Aparecida. (2001). "Aprendendo lógica e programação via web".

VAHLDICK, Adilson. (2007). "Uma experiência lúdica no ensino de programação orientada a objetos". In: I Workshop de Ambientes de Apoio à Aprendizagem de Algoritmos e Programação–Simpósio Brasileiro de Informática na Educação.

VALASKI, Joselaine; PARAISO, Emerson Cabrera. (2012). "Limitações da utilização do Alice no ensino de programação para alunos de graduação". In: Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE).