CENTRO UNIVERSITÁRIO INTERNACIONAL UNINTER
MESTRADO PROFISSIONAL EM EDUCAÇÃO E NOVAS TECNOLOGIAS

GILBERTO ANTONIO MÜLLER SOBRINHO

PRODUTO DA DISSERTAÇÃO AULA GAMIFICADA DE PROGRAMAÇÃO ORIENTADA A OBJETOS

CURITIBA
2023

5 DESENVOLVENDO O APRENDIZADO DE PROGRAMAÇÃO ORIENTADA A OBJETOS COM GAMIFICAÇÃO – PRODUTO DA DISSERTAÇÃO

Programação orientada a objetos é um paradigma de programação amplamente utilizado em várias linguagens de programação. Ensinar esse paradigma aos estudantes é crucial para prepará-los para carreiras de sucesso na área de desenvolvimento de software. No entanto, muitos alunos enfrentam desafios ao aprender esses conceitos, pois podem ser abstratos e difíceis de entender, sem exemplos concretos e práticos. Neste capítulo, apresentaremos uma roda de conversa realizada com profissionais da área para embasar o porquê da utilização da linguagem de programação Java no produto desta dissertação, uma aula de programação orientada a objetos gamificada. Em seguida, a aula foi desenvolvida para abordar os conceitos fundamentais de orientação a objetos, como objetos e classes, herança e polimorfismo, encapsulamento de forma mais envolvente, facilitando a compreensão dos alunos e com demonstrações práticas de como utilizamos e trabalhamos com a linguagem Java.

5.1 RODA DE CONVERSA

Para situar a composição da aula com as práticas realizadas por profissionais da área de engenharia de software foi realizada uma roda de conversa. Na composição do grupo foram considerados, para a participação, critérios que permitissem relatos de práticas de programadores iniciantes, experientes e seniores, observando a inclusão de diversidade de gênero masculino e feminino. A roda de conversa se assemelha a um grupo de discussão que aborda os pontos fundamentais das práticas de profissionais coletivamente. Para Weller (2010, p.247), esses tipos de coleta de dados é “um instrumento por meio do qual o pesquisador estabelece uma via de acesso que permite a reconstrução dos diferentes meios sociais e do *habitus* coletivo do grupo”. A autora ressalta que “seu objetivo principal é a análise dos epifenômenos (subproduto ocasional de outro) relacionados ao meio social, ao contexto geracional, às experiências de exclusão social (WELLER, 2010, p.247). Na obtenção dos dados provenientes das experiências se evidência a contextualização da plataforma a ser produzida cotejadas com as práticas dos profissionais programadores. Isso permite nas análises a identificação do que importante no tratamento do tema.

Além disso, a roda de conversa é “uma forma de coleta de dados em que o pesquisador se insere como sujeito da pesquisa pela participação na conversa” (MOURA; LIMA, 2014, p. 25). Esses autores ressaltam que a roda de conversa é um processo de coleta de dados que permite compartilhar experiências e o desenvolvimento de reflexões sobre um assunto, que os participantes expressam considerações sobre suas práticas. “Rodas de Conversa, quando utilizada como instrumento de pesquisa, uma conversa em um ambiente propício para o diálogo”. (MOURA; LIMA, 2014, p. 25). Trata-se de uma roda de conversa porque não segue todas as regras de um grupo focal. A base desse tipo de coleta de dados toma como pressuposto as argumentações de Weller (2010).

Ressalta Weller (2010, p. 246) que “Os grupos de discussão, como método de pesquisa, passaram a ser utilizados a partir da década de 1980, sobretudo nas pesquisas sobre juventude”. A autora destaca que nesta técnica, a organização da coleta de dados se dá em fases, conforme Weller (2010, p.251): “Seleção das passagens relevantes para a pesquisa; transcrição da passagem inicial, das passagens de foco e daqueles relevantes para a pesquisa; reconstrução da estrutura temática da passagem a ser analisada, que também poderá ser dividida em temas e subtemas”. Esta técnica exige a transcrição completa das falas realizadas durante a roda de conversa, e recomenda uma análise do tipo documentário de interpretação (WELLER, 2010, p. 251), no qual

> Esse processo compreende dois momentos: interpretação formulada e interpretação refletida. Durante a interpretação formulada, busca-se compreender o sentido imanente das discussões e decodificar o vocabulário coloquial.

Com efeito, o pesquisador faz o registro escrito do que foi dito pelos participantes evidenciando “o conteúdo dessas falas para uma linguagem que também poderá ser compreendida por aqueles que não pertencem ao meio social pesquisado”. (WELLER, 2010, p. 251).

Na finalização das análises Weller (2010, p. 252) afirma que “Toda interpretação somente ganhará forma e conteúdo quando realizada e fundamentada na comparação com outros casos empíricos” e em seguida colocada em discussão com os autores que fundamentaram o estudo, favorecendo elaborar apontamentos que se referem às situações típicas de um determinado grupo social e não restritas ao grupo participante.

Assim, para a roda de conversa foi feita a consulta da possibilidade de participação, com agenda definida em dia e hora. A roda de conversa foi gravada

e por isso foi escolhida o aplicativo Teams da Microsoft. Como os participantes são diferentes países foi necessário realizar com cuidado o agendamento pela diversidade de fusos horários.

Na organização da coleta de dados um planejamento das questões foi feito. Na definição dos itens foi feito consulta a outras pesquisas que abordam composição plataformas. Foram definidos como Tópicos orientadores da conversa:

- Introdução
- Pontos chave do aprendizado
- Dificuldades
- Qual a contribuição do Java na opinião de vocês?
- O que mais vocês (convidados) teriam para acrescentar?

Para a seleção dos participantes foram observadas as recomendações de Weller (2010) que recomenda incluir pessoas experientes com o assunto tratado, e mesmo que o próprio pesquisador deve ter alguma informação sobre a pertinência dos incluídos em relação ao assunto abordado. No caso, os integrantes pertencem aos mesmos grupos sociais de programadores. Essa pertinência favorece o levantamento de pontos pertinente durante a conversa. Moura e Lima (2014) indicam a importância da realização de convite para a participação, e que o convite deve ser feito de modo a conciliar a disponibilidade de todos em relação à agenda estabelecida. Os convites foram feitos para:

- José[12] – Engenheiro de Software, reside no exterior.
- Paula – Engenheira Elétrica de formação e está mudando de área, mais especificamente na área de Engenharia de Dados, morou no exterior.
- Rogério – Engenheiro de Software com bastante experiência, residiu no exterior.
- Laerte – Engenheiro de Dados, residiu no exterior.
- Aldair – Gerente de Arquitetura de Software em uma multinacional.

Após os convites realizados por meio de WhatsApp, os participantes expressaram seu consentimento em participar da investigação. A roda de

[12] Foram considerados nomes fictícios para preservar o anonimato respeitando as normas éticas em pesquisa com seres humanos.

conversa foi gravada e transcrita. Para a gravação os participantes foram consultados quanto à autorização.

Em relação às questões éticas em pesquisa com seres humanos estão respeitadas as indicações de anonimato dos participantes, o respeito a fidedignidade dos dados consultados. Também são observados os preceitos de confidencialidade dos dados e o consentimento expresso no Termo de Consentimento Livre e Esclarecido, conforme consta do Anexo I. Ainda, observa-se a questão de riscos na coleta de dados não incluindo questões de ordem pessoal para a realização da roda de conversas.

Para análise das falas realizadas durante a roda de conversa foi feito um agrupamento que toma como possibilidade de compreensão dos dados obtidos a análise do conteúdo foi fundamental. Conforme, Bardin (2011) examinar o conteúdo é uma técnica de pesquisa com elementos e características metodológicas que possibilitam analisar os dados com objetividade, a partir da sistematização das falas e contribuições dos participantes. Ressalta Bardin (2011, p. 42), que as técnicas de análise de dizeres ao serem examinadas com procedimentos sistemáticos favorecem fazer submergir os conteúdos presentes nas mensagens contribuindo para apontar indicadores.

Do conjunto das leituras dessas falas, considerando os níveis da primeira leitura, flutuante, até a última leitura interpretativa, realizadas em etapas sucessivos, destacando os pontos fundamentais, se estabelece a releitura compreensiva, a leitura de sistematização destacando os conteúdos importantes para a composição do produto de pesquisa. O processo dessa leitura também toma por referência as etapas propostas por Bardin (2011), para o processo da análise de dados, quais se apresentam como: a descrição analítica, a interpretação referencial dos dados e a análise, realizada pela organização dos conteúdos das falas e pela leitura flutuante, que faculta ao pesquisador elaborar indicativos contribuem para a interpretação dos dados. São atributos de análise, conforme Bardin (2011). Os objetos das falas dos participantes são originários de sua experiência profissional. Essas experiências contêm aspectos psicológicos e comportamentais, relacionados às emoções, pensamentos e ações, como ressalta Bardin (2011). Enfim esses conteúdos incluem propriedades que expressam os espaços e tempos de atuação profissional dos participantes de modo quantitativo e qualitativo, ainda que nessa pesquisa se valorize muito mais a qualidade das proposições e não a sua quantidade.

Com esse conjunto de indicações de procedimentos metodológicos parece ser possível levar a termo essa investigação.

5.1.1 Dados da Roda de Conversa

A roda de conversa, com duração de 1 hora, 55 minutos e 33 segundos, proporcionou um ambiente propício para a troca de ideias e a construção de conhecimento entre os participantes. O grupo apresentou uma diversidade de experiências com linguagens orientadas a objetos, como por exemplo Java, além de outras linguagens de programação, incluindo profissionais que trabalham com Java desde o início dos anos 2000 e aqueles que ingressaram recentemente na área. Em muitos casos, Java foi a primeira ou segunda linguagem aprendida durante a carreira, embora tenha sido a primeira para a grande maioria. Python e NodeJS também foram mencionadas como linguagens adquiridas posteriormente pelos colegas, com alguns deles atuando principalmente com essas linguagens ou, em alguns casos, trabalhando com múltiplas linguagens, dependendo da atividade a ser realizada.

Na formação acadêmica, quase todos os participantes tiveram contato com Java como a primeira linguagem de programação aprendida em sala de aula. A riqueza da documentação, a quantidade de materiais disponíveis para aprendizado e a facilidade de encontrar esses materiais foram destacadas, assim como a dimensão e qualidade da comunidade, a quantidade e qualidade de bibliotecas e frameworks e o fato de algumas dessas bibliotecas serem incorporadas pela linguagem e se tornarem padrão. Além disso, foi mencionada a excelente capacidade de integração do Java com outros componentes, bem como a quantidade de livros traduzidos para o português. Todos os participantes, sem exceção, passaram a consumir mais conteúdo em inglês, o que facilitou bastante o aprendizado, pois aumentou a quantidade e qualidade do material disponível. No entanto, todos comentaram a dificuldade no aprendizado da língua inglesa. No início da carreira, a maioria dos participantes seguiu um aprendizado de maneira desestruturada.

A evolução do Java e a adoção de melhorias de outras linguagens de programação ao longo do tempo, como *Generics*, *Lambda* e *Closure*, também foram discutidas. A velocidade das atualizações da linguagem é um tema polêmico, pois, apesar de trazer novos recursos e capacidades, pode ser um desafio manter a última versão atualizada para algumas empresas e

colaboradores, especialmente quando se possui um grande número de servidores. A disponibilidade de múltiplos *Java Development Kits* (JDKs) e *Java Virtual Machines* (JVMs) foi vista como positiva pelos participantes.

A orientação a objetos foi discutida como a abordagem dominante no início dos anos 2000, e o Java foi lembrado como a linguagem que atraiu muitos programadores devido às inúmeras oportunidades de emprego disponíveis nas empresas. Recursos de aprendizado, como o fórum GUJ e a Java Magazine, foram mencionados com apreço como pilares do aprendizado. Os participantes concordaram que é essencial praticar gradualmente o que se aprende para solidificar o conhecimento. Alguns participantes aprenderam outras linguagens ao longo do tempo, como Scala, Elixir e Closure, e relataram dificuldades com todas elas. Ter parcerias e colaborações durante o aprendizado foi apontado como extremamente útil, assim como a importância de ter objetivos tangíveis e construir projetos passo a passo para alcançar esses objetivos. A leitura e compreensão de código, mesmo sem ter amplo domínio da linguagem, foi destacada como um exercício válido para o aprendizado.

A partir da segunda linguagem adquirida, a curva de aprendizado tende a diminuir. A flexibilidade, ou falta dela, em linguagens de programação foi discutida. Java é vista como uma linguagem mais restrita, o que pode ser desafiador no início do aprendizado, mas, eventualmente, é considerada uma qualidade. Linguagens mais permissivas, como JavaScript, requerem uma base mais sólida para escrever códigos eficientes, devido à maior flexibilidade e ao risco de encontrar armadilhas no código ou seguir caminhos menos eficientes e mais complicados.

Houve discussões sobre o equilíbrio entre o aprofundamento dos conhecimentos em linguagem, estruturas de dados, algoritmos, padrões de projeto e o pragmatismo necessário para ensinar o mínimo indispensável para que alguém possa ingressar no mercado de trabalho. Além disso, foi enfatizado que o conhecimento em linguagem de programação, por si só, não é suficiente. Hoje em dia, é preciso dominar noções de implantação de código em produção, ferramentas de suporte, computação em nuvem, escalabilidade, testes, transações, DevOps e outros aspectos relacionados à área de tecnologia.

É inegável que o contexto tecnológico e a demanda do mercado continuam evoluindo rapidamente. Por isso, os profissionais devem ser capazes de aprender e se adaptar às mudanças na indústria. Desenvolver habilidades de

aprendizagem ao longo da vida, como autodidatismo, resolução de problemas e pensamento crítico, torna-se cada vez mais importante.

A troca de experiências e conhecimentos entre profissionais da área também é fundamental, seja por meio de fóruns, redes sociais, grupos de estudo, workshops ou eventos. Esses espaços de interação proporcionam um ambiente onde os participantes podem compartilhar desafios, soluções e melhores práticas, bem como se manter atualizados sobre as últimas tendências e avanços tecnológicos.

Outra perspectiva levantada foi a importância da diversidade de conhecimentos, indo além do domínio de uma única linguagem de programação. A familiaridade com múltiplas linguagens e tecnologias é benéfica tanto para o crescimento profissional quanto para a resolução de problemas complexos e multifacetados.

Além disso, o mercado atual exige uma compreensão cada vez maior dos aspectos éticos e sociais relacionados ao desenvolvimento e implementação de soluções tecnológicas. Os profissionais de tecnologia da informação devem estar cientes das implicações e consequências de suas ações e decisões no contexto mais amplo da sociedade.

Neste cenário, a roda de conversa corrobora as evidências encontradas na literatura, destacando que o Java ainda é uma linguagem de programação altamente atrativa para o aprendizado atualmente, especialmente como primeira linguagem. Obviamente, não se trata de uma solução perfeita para todas as questões de negócios ou tecnologias. O ideal é utilizar a ferramenta mais adequada para cada tarefa específica, o que sempre representa um desafio. Entretanto, é consenso que o Java é uma linguagem respeitada e com um amplo mercado global.

O contexto geográfico e social pós-pandêmico também impulsiona a relevância do aprendizado de linguagens de programação, não apenas do Java, mas de outras linguagens igualmente, visto que a possibilidade de trabalhar remotamente de qualquer parte do mundo amplia significativamente as oportunidades profissionais. Agora, é possível trabalhar em projetos, colaborar com colegas, atuar em empresas e enfrentar desafios internacionais sem sair de casa. Embora a barreira linguística possa ser um obstáculo em alguns casos, dependendo do cliente, da empresa ou do país, essa questão poderia ser abordada em uma discussão separada.

Dessa forma, a roda de conversa serve como um importante espaço de reflexão e troca de experiências sobre o papel do Java e outras linguagens de programação, no contexto atual e futuro da indústria tecnológica. Ela também enfatiza a necessidade de se adaptar às mudanças e desafios do mercado de trabalho, buscando o aprimoramento contínuo das habilidades e competências necessárias para ter sucesso nesse ambiente dinâmico e globalizado.

Em suma, o debate sobre a melhor primeira linguagem de programação a ser aprendida é complexo e multifacetado, envolvendo considerações pedagógicas, mercadológicas, técnicas e pessoais. Independentemente da linguagem escolhida, é fundamental investir no desenvolvimento contínuo das habilidades e competências que permitam aos profissionais enfrentar os desafios do mercado e adaptar-se às constantes mudanças no campo da tecnologia da informação, tendo sempre em mente a responsabilidade ética e social inerente à atuação na área.

5.2 AULA GAMIFICADA DE PROGRAMAÇÃO ORIENTADA A OBJETOS

Esta dissertação propõe uma aula interativa de 140 minutos (recomendável com um intervalo de 10 minutos), com o objetivo de promover o ensino para a aprendizagem dos conceitos fundamentais de programação orientada a objetos, empregando a gamificação como meio de aumentar o engajamento e a compreensão dos estudantes. A linguagem de programação Java será utilizada como base para as atividades práticas. A aula proposta toma por referência os fundamentos teóricos da programação orientada a objetos desenvolvidos nessa pesquisa e dos dados coletados na roda de conversa. Portanto, cinge-se a uma proposta, não aplicada. A sua aplicação poderá gerar um novo estudo.

A aula pode ser amparada por recursos auxiliares como slides, quadro branco, códigos-exemplo, um quadro de líderes e um sistema de recompensas. Essa estrutura pode ser facilmente adaptada para um ambiente online, utilizando soluções digitais como alternativas ao quadro branco ou lousa.

Troféu e medalhas podem ser de ouro, prata e bronze, sendo 50 pontos para a o troféu que somente será presenteada em ocasiões especiais, 10 pontos para a medalha de ouro, 6 para a medalha de prata e 3 para a medalha de bronze.

O troféu e medalhas serão confeccionados em uma imagem JPG e será distribuído formalmente pelo professor, com menção a todos os participantes da aula, destacando que todos conseguiram e que, independente da dificuldade, todos estão aptos.

Fonte: Canva[13]

O quadro de liderança ficará exposto para todos os participantes da aula, mas sempre buscando incentivar os que estão abaixo na tabela, mostrando como a diferença em relação aos vencedores não é inatingível.

A fim de equalizar os participantes da aula, recompensar devem ser dadas observando não apenas os resultados práticos, mas também os esforços. Por isso, o professor poderá distribuir recompensas não aos líderes, mas sim aos que mais se esforçaram, como incentivo:

- Ganhará 10 pontos e uma medalha de ouro, o participante da aula que concluir a atividade ou desafio sem ajuda, 6 pontos e uma medalha de prata os que conseguirem com a ajuda de algum colega e 3 pontos e uma medalha de bronze aquele que conseguir com a ajuda do professor.
- Conclusão de todas as tarefas sem ajuda dos colegas ou professor: Medalha de ouro e 10 pontos
- Alunos que auxiliem um colega recebem uma medalha de bronze adicional e mais 3 pontos. Alunos que auxiliem três colegas recebem uma medalha adicional de prata e mais 6 pontos. Alunos que auxiliarem cinco colegas ou mais recebem uma medalha adicional de ouro e mais 10 pontos para cada cinco colegas auxiliados

[13] http://www.canva.com

- O participante da aula que tiver auxiliado o maior número de colegas recebe um troféu e 50 pontos

Este esquema se aplica para cada tópico ou módulo da aula, sendo que cada tópico ou módulo pode possuir desafios adicionais.

O plano de aula a seguir apresenta detalhes sobre o conteúdo a ser ministrado e a aplicação da gamificação como metodologia de ensino, além de sugestões práticas para o uso da linguagem Java durante a atividade.

Agenda da aula:

- Introdução
- Configuração do ambiente de desenvolvimento Java
- Classes e objetos
- Herança e polimorfismo
- Encapsulamento
- Conclusão e recompensas

Este projeto Java será escrito com o auxílio da biblioteca Maven (para vias de simplificação na criação do projeto Java, detalhes não serão cobertos pela aula) e ferramenta IDE NetBeans, ambos são projetos da fundação Apache.

Apache Maven é uma ferramenta de automação de compilação e gerenciamento de projetos de software, amplamente utilizada em projetos Java. Ele ajuda a padronizar o processo de construção, facilitando a compilação, teste, empacotamento, implantação e distribuição de projetos de software.

NetBeans é um ambiente de desenvolvimento integrado (IDE - *Integrated Development Environment*) de código aberto e gratuito, desenvolvido principalmente para a linguagem de programação Java, mas também suporta outras linguagens. Foi inicialmente lançado em 1997.

5.2.1 Introdução

Tempo: 15 minutos

Apresentar brevemente o tema da aula que é a programação orientada a objetos explicando conceitos como classes, objetos, herança, polimorfismo e encapsulamento.

Introduzir a gamificação como uma abordagem para melhorar o envolvimento e a aprendizagem, discutindo elementos como pontos, troféus e medalhas, quadros de liderança e recompensas.

5.2.2 Configuração do Ambiente de Desenvolvimento Java

Tempo: 25 minutos

Orientar os estudantes através da configuração de um ambiente de desenvolvimento Java ou *Java Development Kit*, instalação da IDE Apache NetBeans e criação de um projeto Java.

Instalar o OpenJDK:

a. Acesse o site do Adoptium: https://adoptium.net/.

b. Selecione a versão mais recente do OpenJDK adequada ao seu sistema operacional (Windows, macOS ou Linux).

c. Faça o download do instalador e execute-o.

d. Siga as instruções na tela para concluir a instalação.

Quem instalar o OpenJDK primeiro ganha uma medalha de bronze e 3 pontos.

Instalar o Apache NetBeans:

a. Acesse o site oficial do Apache NetBeans através do seguinte link: https://netbeans.apache.org/download/index.html.

b. Localize e selecione a versão apropriada do NetBeans para o seu sistema operacional (Windows, macOS ou Linux).

c. Faça o download do arquivo de instalação.

d. Execute o arquivo baixado e siga as instruções na tela para concluir a instalação, isso pode ser geralmente feito através de duplo clique no ícone do instalador.

Após a instalação bem-sucedida do Apache NetBeans, inicie o programa clicando no ícone do NetBeans no seu computador.

Quem instalar o Apache NetBeans primeiro ganha uma medalha de bronze e 3 pontos.

Para confirmar se o Java está instalado no seu computador, siga estas instruções para Windows e macOS:

Windows: Pressione as teclas *Win* + *R* no seu teclado para abrir a janela “Executar”. Digite *cmd* e pressione *Enter* para abrir o “Prompt de Comando”. Na

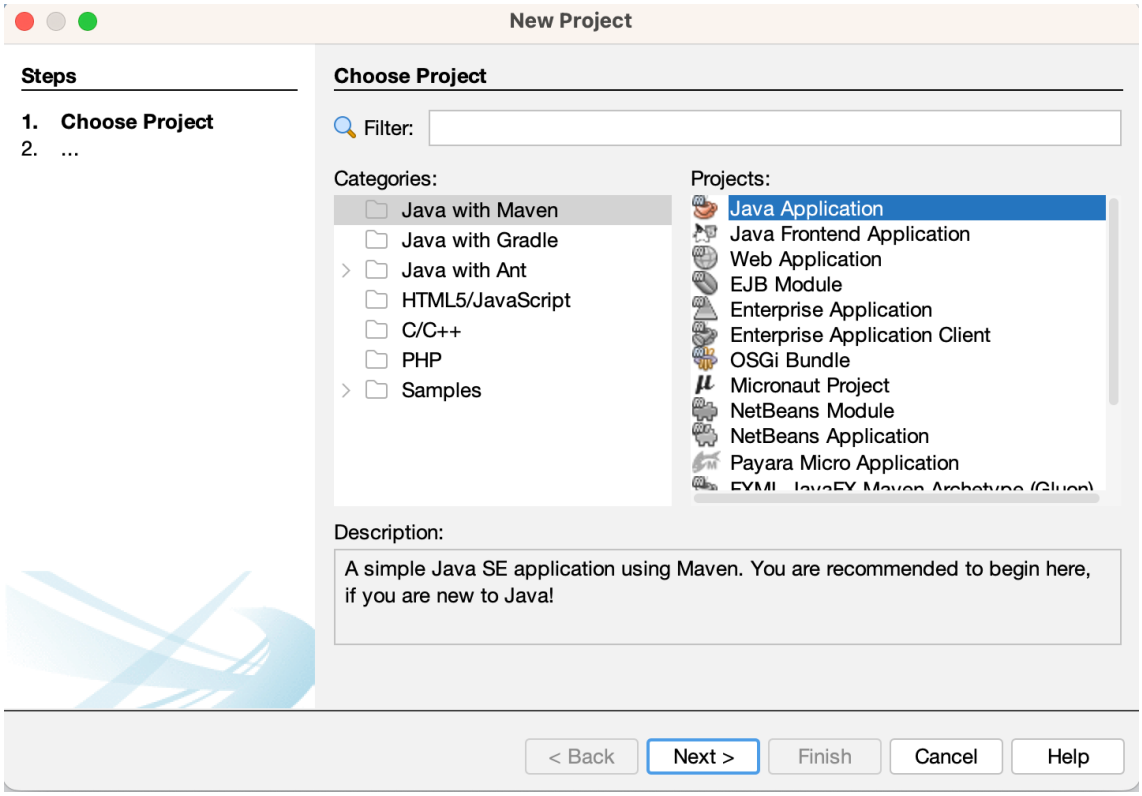
janela do “Prompt de Comando”, digite o seguinte comando e pressione Enter: “*java -version*”. Se o Java estiver instalado, você verá informações sobre a versão do Java, como “openjdk version” ou “java version”. Se não estiver instalado, você verá uma mensagem indicando que o comando não foi encontrado.

macOS: Abra o aplicativo “Terminal” (você pode encontrá-lo no Launchpad ou usando a busca no Spotlight). Na janela do “Terminal”, digite o seguinte comando e pressione Enter: “java -version”. Se o Java estiver instalado, você verá informações sobre a versão do Java, como “openjdk version” ou “Java version”. Se não estiver instalado, você verá uma mensagem indicando que o comando não foi encontrado.

Exemplo de instalação bem sucedida do JDK:

```
$ java -version
openjdk version "20" 2023-03-21
OpenJDK Runtime Environment Homebrew (build 20)
OpenJDK 64-Bit Server VM Homebrew (build 20, mixed mode, sharing)
```

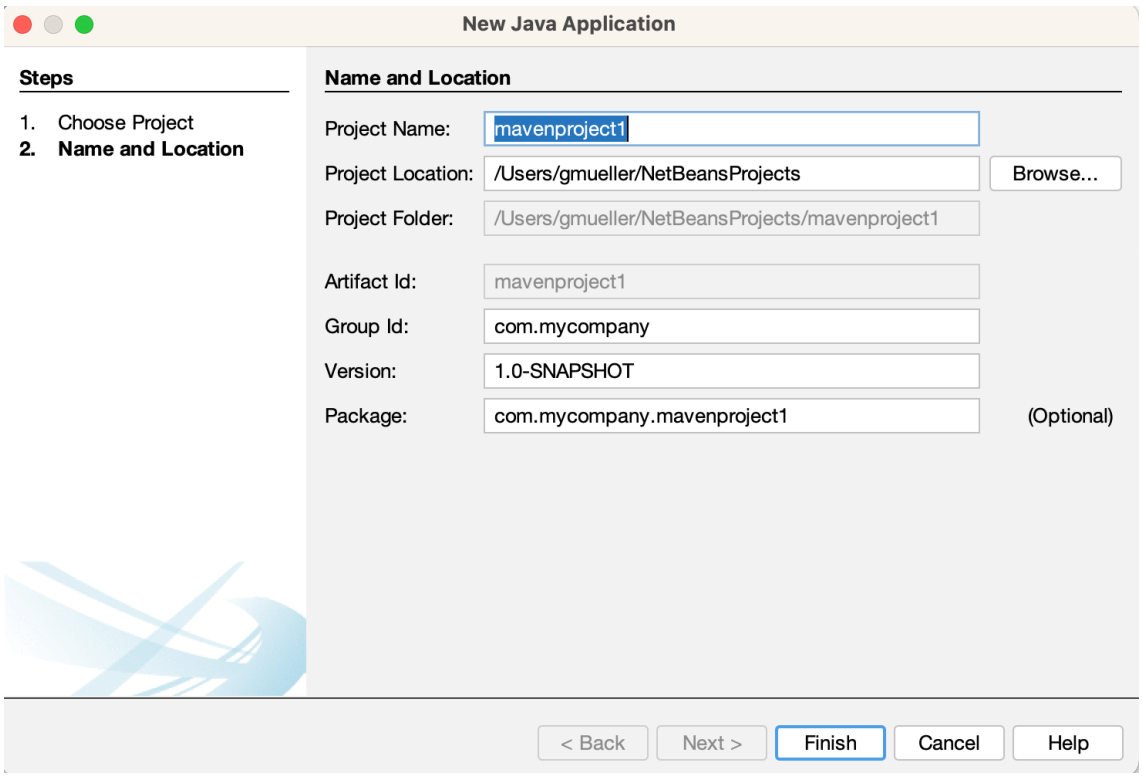
Para criar um projeto Maven no Apache NetBeans, siga as instruções: Abra o Apache NetBeans. No menu superior, clique em Arquivo (ou “*File”*, caso esteja em inglês) e selecione “Novo Projeto...” (ou “*New Project*...”). Na janela “Novo Projeto” que se abre, expanda a categoria Java com Maven (ou “*Java with Maven”*, caso esteja em inglês). Selecione Aplicação Java (ou “*Java Application”*) e clique em Próximo (ou *Next*).



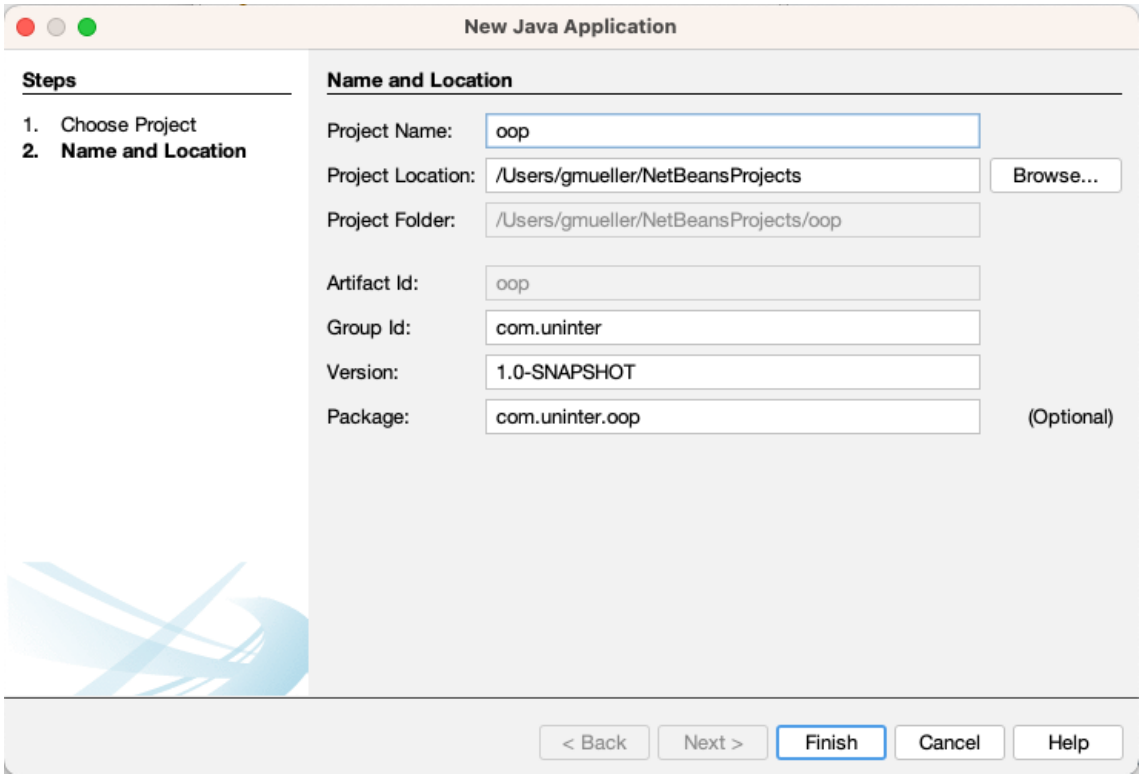
Agora você verá a tela “Nome e Localização do Projeto” (ou “*Project Name and Location”*). Aqui, você pode configurar o nome do projeto, o local onde ele será salvo e o pacote base. Preencha os campos conforme desejado e clique em Finalizar (ou “*Finish”*). Neste caso iremos digitar no campo *Project Name* ou Nome do Projeto “oop” e no campo Group Id: “com.uninter.oop”

O Apache NetBeans criará o projeto Maven e abrirá a estrutura de diretórios na janela “Projetos” (ou “*Projects”*). Você pode começar a escrever seu código no arquivo, neste caso Oop.java, Mavenproject1.java caso você utilize a opção padrão, ou o nome que você tenha escolhido. O arquivo está localizado dentro do pacote (*package*) especificado anteriormente.

Opção padrão da tela.



Após a customização desejada.



Atribuir pontos para a conclusão bem sucedida da configuração e atualizar o quadro de liderança com a inclusão de um desafio extra: o participante da aula que for mais criativo e conseguir explorar novas classes e objetos no fim da tarefa receberá adicionalmente uma medalha de prata e 6 pontos.

Quem criar o projeto com o nome mais criativo ganha uma medalha de bronze e 3 pontos. Quem criar o projeto com o nome menos criativo também ganha uma medalha de bronze e 3 pontos.

Quem criar o pacote com o nome mais criativo ganha uma medalha de bronze e 3 pontos. Quem criar o pacote com o nome menos criativo também ganha uma medalha de bronze e 3 pontos.

Os participantes da aula vão escolher os vencedores com a ajuda do professor.

5.2.3 Classes e Objetos Java

Tempo: 25 minutos

Ensinar os participantes da aula a criar classes e objetos em Java.

Uma classe em Java é um bloco de construção fundamental na programação orientada a objetos (OOP) usando a linguagem Java. Uma classe é um modelo ou um plano que define a estrutura e o comportamento dos objetos que serão criados com base nessa classe. Em outras palavras, uma classe é um conjunto de especificações que descrevem as propriedades (também chamadas de atributos) e os métodos (ações ou comportamentos) que os objetos dessa classe possuem.

A classe serve como um modelo a partir do qual os objetos são criados ou instanciados. Cada objeto criado a partir de uma classe é chamado de “instância” da classe e possui seu próprio conjunto de valores para as propriedades e pode executar os métodos definidos na classe.

Em Java, as classes são definidas usando a palavra-chave *class*, seguida pelo nome da classe e um bloco de código que contém os atributos e métodos da classe.

Um *package* ou pacote é uma forma de organizar classes e interfaces relacionadas em um único *container* ou diretório. Pacotes facilitam a modularização, reutilização e manutenção do código, além de evitar conflitos de nomes entre classes com o mesmo nome em diferentes partes do projeto. Eles também fornecem um mecanismo para controlar o acesso a membros de classes e interfaces por meio de modificadores de acesso.

Um pacote é criado declarando-se uma instrução de pacote no início de um arquivo de código-fonte Java, antes de qualquer declaração de classe ou

interface. A instrução do pacote é seguida pelo nome do pacote, que geralmente segue uma convenção de nomenclatura baseada no domínio reverso da organização ou desenvolvedor responsável pelo código, garantindo a unicidade dos nomes de pacotes.

Objetos são a representação de entidades do mundo real, que possuem características (atributos) e comportamentos (métodos). Um objeto é uma instância de uma classe, que é a estrutura ou modelo que define os atributos e métodos que um objeto pode ter.

Na prática para criar uma nova classe Java no Apache NetBeans, siga as etapas abaixo:

Abra o Apache NetBeans e certifique-se de que o seu projeto está aberto na janela “Projetos” (ou “Projects”).

Expanda os diretórios do seu projeto até encontrar o pacote onde deseja adicionar a nova classe. Os pacotes geralmente estão localizados dentro do diretório `src/main/java`. Ou dentro da estrutura de pacote customizada, neste caso `src/main/java/com/uninter/oop`. Esta estrutura se encontra dentro do diretório do projeto criado.

Clique com o botão direito do mouse no pacote onde deseja criar a nova classe e selecione `Novo Arquivo` (ou `New File`) > `Classe Java...` (ou `Java Class...`).

Na janela “Nova Classe Java” (ou “New Java Class”) que aparece, insira o nome da classe no campo “Nome da Classe” (ou “Class Name”). É importante seguir as convenções de nomenclatura, ou seja, começar com uma letra maiúscula e usar o formato CamelCase.

(Opcional) Se desejar alterar o pacote, você pode fazê-lo no campo “Pacote” (ou “Package”).

Clique em `Finalizar` (ou `Finish`) para criar a nova classe.

Quem criar a classe com o nome mais criativo ganha uma medalha de bronze e 3 pontos. Quem criar a classe com o nome menos criativo também ganha uma medalha de bronze e 3 pontos.

Neste exemplo criamos uma classe Carro da seguinte forma (caso queira copiar e colar):

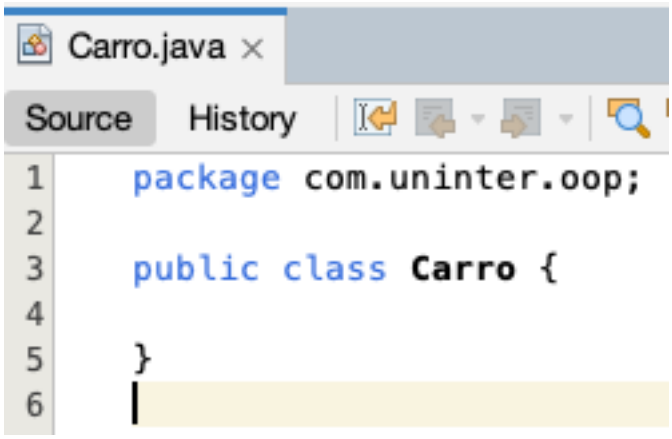
Classe Carro.java e seu conteúdo

```
package com.uninter.oop;

public class Carro {

}
```

Imagem da classe Carro no NetBeans.



Quem criar outra classe com o nome mais criativo ganha uma medalha de bronze e 3 pontos. Quem criar outra classe com o nome menos criativo também ganha uma medalha de bronze e 3 pontos.

Neste exemplo editamos a classe Oop.java (ou outro nome caso tenha escolhido anteriormente) e instanciamos a classe Carro armazenando-a no objeto *carro*. Não vamos nos ater aos detalhes do “public static void main” agora em Java, por hora basta saber que devemos colocar as instruções de código dentro deste bloco para sua execução.

Classe Oop.java e seu conteúdo

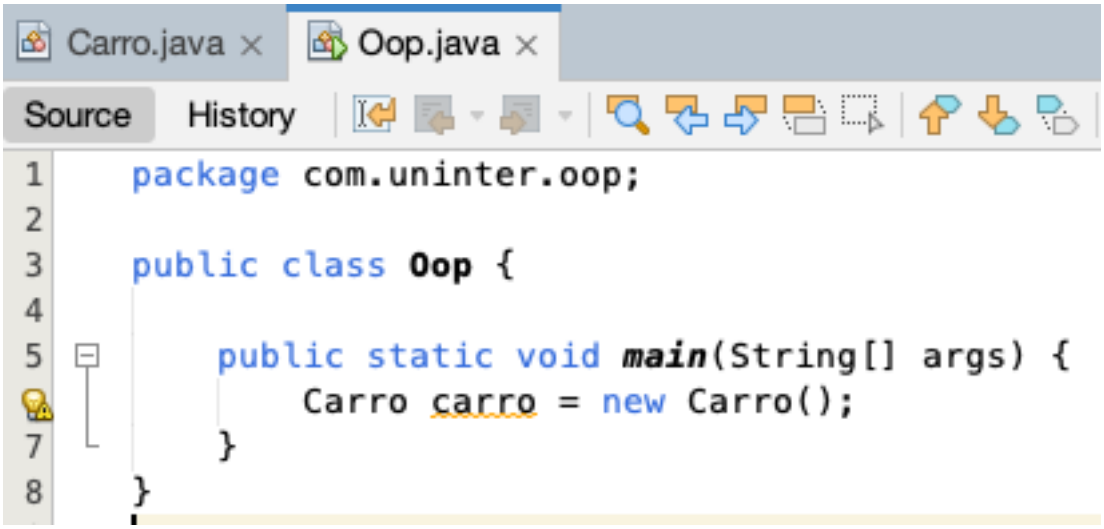
```
package com.uninter.oop;

public class Oop {

  public static void main(String[] args) {
    Carro carro = new Carro();
  }
```

```
}
```

Imagem da classe Oop no NetBeans.



Logo neste caso a classe é Carro, e o objeto é carro também. Para objetivo de clareza. Poderíamos instanciar várias vezes a classe Carro, cada uma em um objeto distinto, como por exemplo:

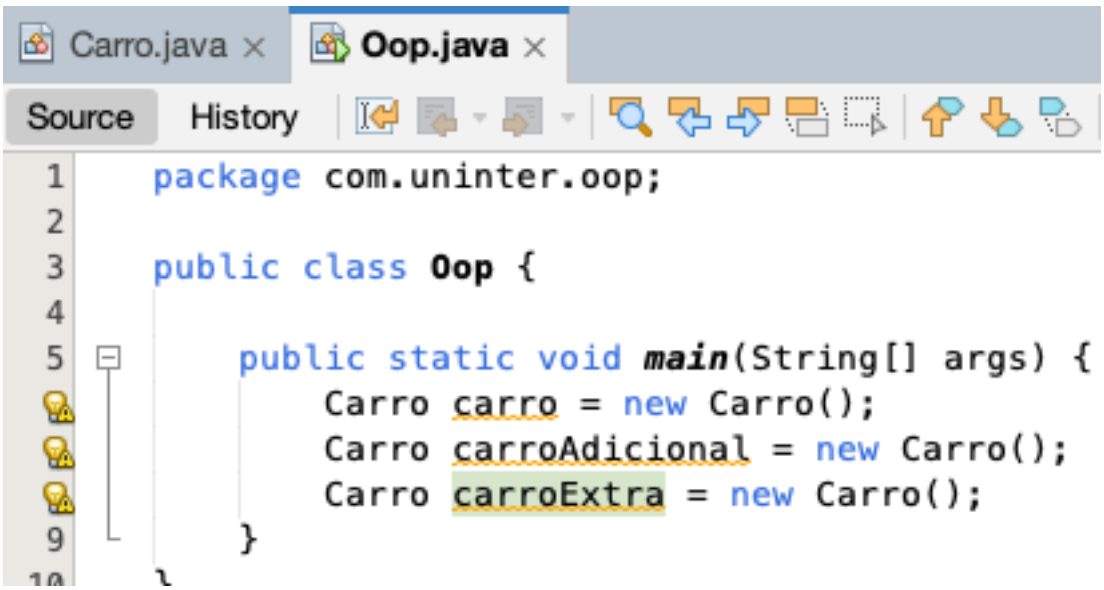
Classe Oop.java e seu conteúdo

```
package com.uninter.oop;

public class Oop {

  public static void main(String[] args) {
    Carro carro = new Carro();
    Carro carroAdicional = new Carro();
    Carro carroExtra = new Carro();
  }
}
```

Imagem da classe Oop no NetBeans após as novas instancias de Carro.



Atribuir pontos para a conclusão bem sucedida da tarefa e atualizar o quadro de liderança com a inclusão de um desafio extra: o aluno que for mais criativo e conseguir explorar novas classes e objetos (como tipos de combustíveis, veículos etc.) no fim da tarefa receberá adicionalmente uma medalha de prata e 6 pontos. Os participantes da aula vão escolher o vencedor com a ajuda do professor.

5.2.4 Herança e Polimorfismo

Tempo: 25 minutos

Ensinar aos participantes da aula sobre herança e polimorfismo em Java, usando exemplos do desafio de codificação anterior.

Herança é um mecanismo que permite a criação de uma nova classe a partir de uma classe existente, herdando seus atributos e métodos. A nova classe é chamada de subclasse e a classe existente de superclasse. Uma superclasse é uma classe da qual outras classes podem herdar atributos e métodos. Em outras palavras, é uma classe pai ou classe base de uma ou mais subclasses.

Em Java, a palavra-chave *extends* é usada para implementar a herança. A herança promove a reutilização de código e a organização hierárquica das classes, ou que o mesmo bloco de código seja reutilizado de forma que com o

aumento da quantidade de código escrito não necessariamente o código cresce de maneira diretamente proporcional.

Polimorfismo é a capacidade de uma classe assumir várias formas, permitindo que objetos de diferentes classes sejam tratados como objetos de uma classe comum. O termo “polimorfismo” deriva das palavras gregas “*poli*” (muitos) e “*morf*” (formas), sugerindo que uma única classe ou interface pode assumir múltiplas formas. Em Java, o polimorfismo é implementado através de herança e interfaces. O polimorfismo permite a reutilização de código e flexibilidade no design do software. Em Java pode ser implementado de duas maneiras: polimorfismo de subtipos (também conhecido como polimorfismo de tempo de execução) e polimorfismo paramétrico (também conhecido como polimorfismo de tempo de compilação).

Vamos demonstrar na prática este conceito criando a classe Combustivel (sem acento mesmo pois por padrão classes Java devem ser simplificadas, ou sem caracteres especiais). Crie também a classe Etanol seguindo os passos anteriormente demonstrados para a criação da classe Carro, só mude o nome para Combustivel e Etanol respectivamente.

Classe Combustivel.java e seu conteúdo

```
package com.uninter.oop;

public class Combustivel {

}
```

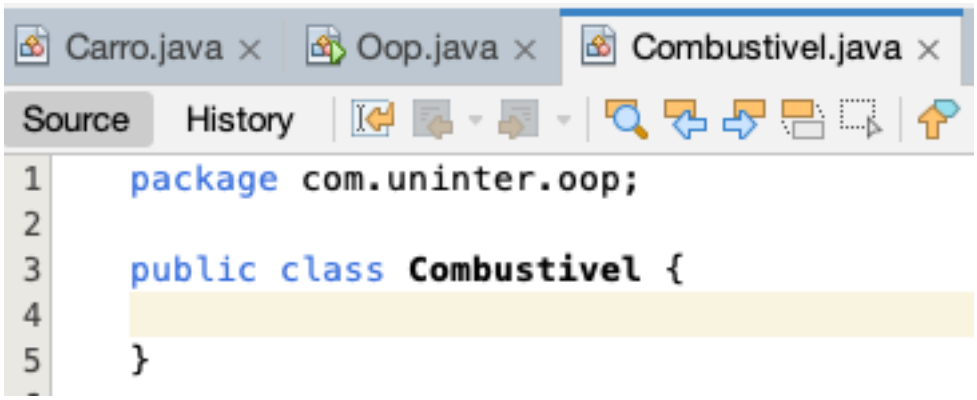
Classe Etanol.java e seu conteúdo

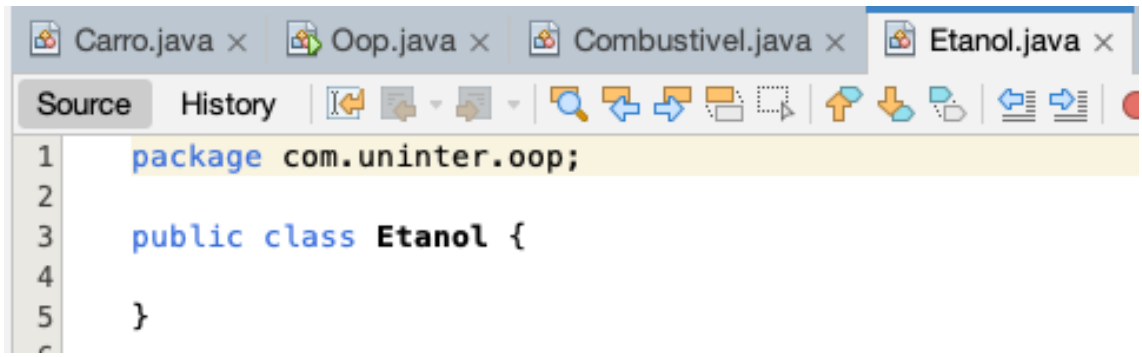
```
package com.uninter.oop;

public class Etanol {

}
```

Imagem das classes Combustivel e Etanol no NetBeans.





Para demonstrar herança, iremos alterar a classe Etanol de forma que ela agora estenda a classe Combustivel com a adição da palavra *extends* e a classe que desejamos, da seguinte forma:

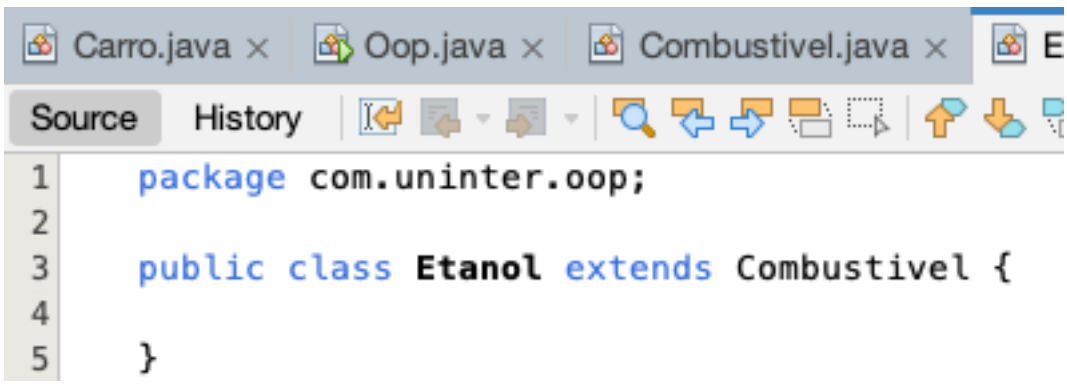
Classe Etanol.java e seu conteúdo agora com herança

```
package com.uninter.oop;

public class Etanol extends Combustivel {

}
```

Exemplo da classe Etanol estendendo a classe Combustível como demonstração de herança.



Iremos agora trabalhar o polimorfismo demonstrando a Classe Etanol sendo instanciada em um objeto Combustivel.

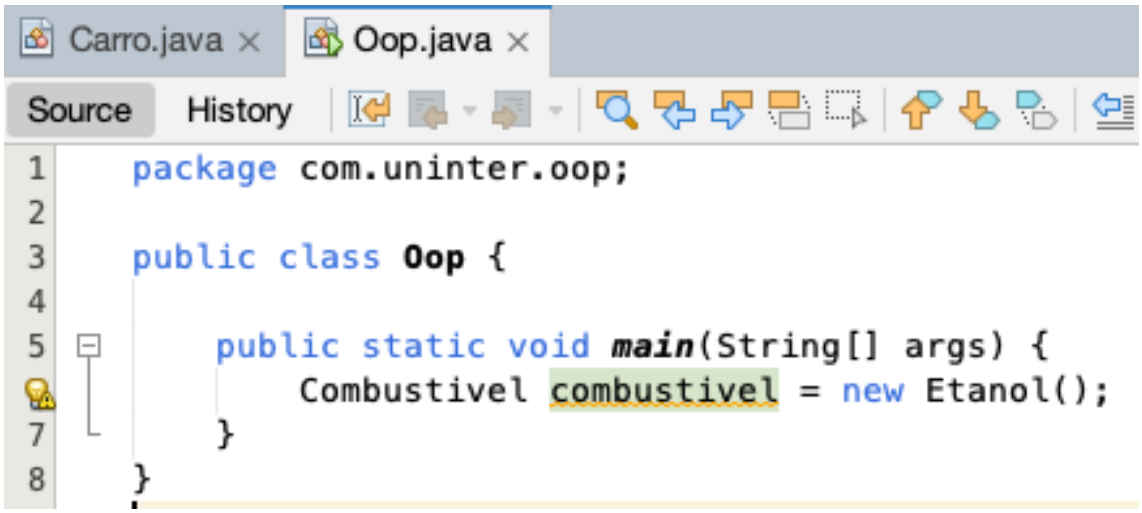
Classe Oop.java demonstrando polimorfismo

```
package com.uninter.oop;

public class Oop {

  public static void main(String[] args) {
    Combustivel combustivel = new Etanol();
  }
}
```

Imagem da class Oop com exemplo de polimorfismo instanciando a classe Etanol no objeto Combustivel.



Este conceito nos permite tratar diversas classes quando existem similaridades, por exemplo poderíamos criar classes de Gasolina. E estender Gasolina em classes de GasolinaComum ou GasolinaAditivada. Poderíamos utilizar este conceito para se trabalhar com um motor Total Flex por exemplo, o qual seria abastecido com um combustível. E este combustível poderia ser gasolina comum, gasolina aditivada, etanol etc.

Atribuir pontos para a conclusão bem sucedida da tarefa e atualizar o quadro de liderança com a inclusão de um desafio extra, o desafio de

implementações elegantes e uso extensivo de herança e polimorfismo: o aluno que for mais criativo e conseguir explorar outras classes e herança no fim da tarefa receberá adicionalmente uma medalha de prata e 6 pontos. Os participantes da aula vão escolher o vencedor com a ajuda do professor.

5.2.5 Encapsulamento

Tempo: 25 minutos

Ensinar aos participantes da aula sobre encapsulamento e a importância de esconder dados em Java.

Atribuir pontos para a conclusão bem sucedida da tarefa e atualizar o quadro de liderança com a inclusão de um desafio extra, o desafio de criatividade: o aluno que for mais criativo e conseguir explorar o encapsulamento no fim da tarefa receberá adicionalmente uma medalha de prata e 6 pontos. Os participantes da aula vão escolher o vencedor com a ajuda do professor.

Encapsulamento é o mecanismo que protege os atributos e métodos de uma classe, restringindo o acesso a eles. Em Java, "*private"*, "*protected"* e "*public*" são palavras-chave que definem os diferentes níveis de acesso aos membros de uma classe:

- *"private"*: membros de uma classe definidos como "*private"* só podem ser acessados por métodos dentro da mesma classe. Isso significa que esses membros são encapsulados e não podem ser acessados diretamente de fora da classe.
- "*protected"*: membros de uma classe definidos como "*protected"* podem ser acessados por métodos dentro da mesma classe, bem como por subclasses da classe e por outras classes no mesmo pacote.
- "*public*": membros de uma classe definidos como "*public*" podem ser acessados por qualquer classe em qualquer pacote, tornando-os completamente visíveis e acessíveis.

Esses níveis de acesso são importantes para garantir a segurança e a integridade do código em um sistema. Usar "private" para membros que não devem ser acessados de fora da classe ajuda a evitar erros de programação e limita o acesso ao código que deve ser protegido. Usar "protected" para membros que precisam ser acessados por subclasses ou outras classes no mesmo pacote

permite uma maior flexibilidade de uso, enquanto ainda protege o código contra acesso não autorizado. Finalmente, usar “public” para membros que precisam ser acessados por outras classes permite a interoperabilidade do código entre diferentes partes de um sistema.

O encapsulamento garante a integridade dos dados e a separação de responsabilidades entre as classes.

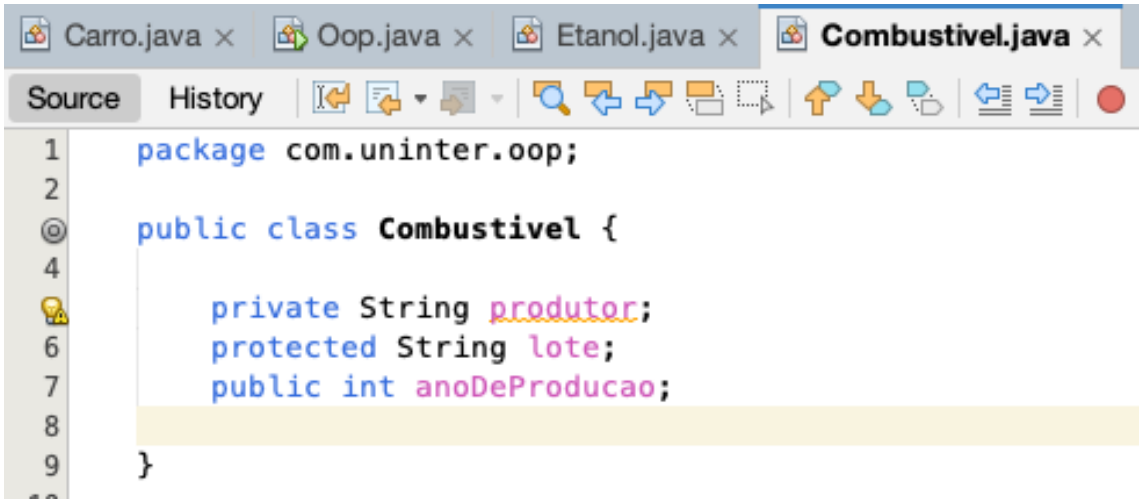
Classe Combustivel.java com demonstração de “private”, “protected” e “public”.

```
package com.uninter.oop;

public class Combustivel {

  private String produtor;
  protected String lote;
  public int anoDeProducao;

}
```



“Getters” e “*setters”* são métodos em uma classe que permitem o acesso e a manipulação dos atributos de um objeto, garantindo o princípio do encapsulamento na Programação Orientada a Objetos (POO). Eles servem como uma interface para acessar e modificar os atributos de um objeto de maneira controlada, sem expor diretamente os atributos, protegendo assim a integridade dos dados.

“Getters:” Um “*getter”* é um método que retorna o valor de um atributo específico do objeto. Geralmente, o nome do método começa com “*get”* seguido pelo nome do atributo com a primeira letra em maiúsculo. Os métodos “*getters"* não recebem parâmetros e retornam o valor do atributo associado.

“Setters”: Um “*setter”* é um método que define o valor de um atributo específico do objeto. Geralmente, o nome do método começa com “*set”* seguido pelo nome do atributo com a primeira letra em maiúsculo. Os métodos “*setters”* recebem um parâmetro do mesmo tipo do atributo e não retornam nenhum valor.

No exemplo abaixo, produtor é “*private”* mas será exposto através de “*set”* e “*get”*.

Classe Combustivel.java com exemplo de “*set”* e “*get”*.

```
package com.uninter.oop;

public class Combustivel {

  private String produtor;
  protected String lote;
  public int anoDeProducao;

  public String getProdutor() {
    return produtor;
  }

  public void setProdutor(String produtor) {
    this.produtor = produtor;
  }

}
```

Imagem da classe Combustivel com exemplos de “*set”* e “*get”*.

Atribuir pontos para a conclusão bem sucedida da tarefa e atualizar o quadro de liderança com a inclusão de um desafio extra, o desafio de encapsulamento: o aluno que for mais criativo e conseguir criar outras classes de forma a explorar o encapsulamento no fim da tarefa receberá adicionalmente uma medalha de prata e 6 pontos. Os alunos vão escolher o vencedor com a ajuda do professor.

5.2.6 Conclusão e Recompensas

Tempo: 25 minutos

Recapitular os principais pontos de aprendizagem e conceitos da lição.

Classe Oop.java demonstrando herança, polimorfismo e encapsulamento.

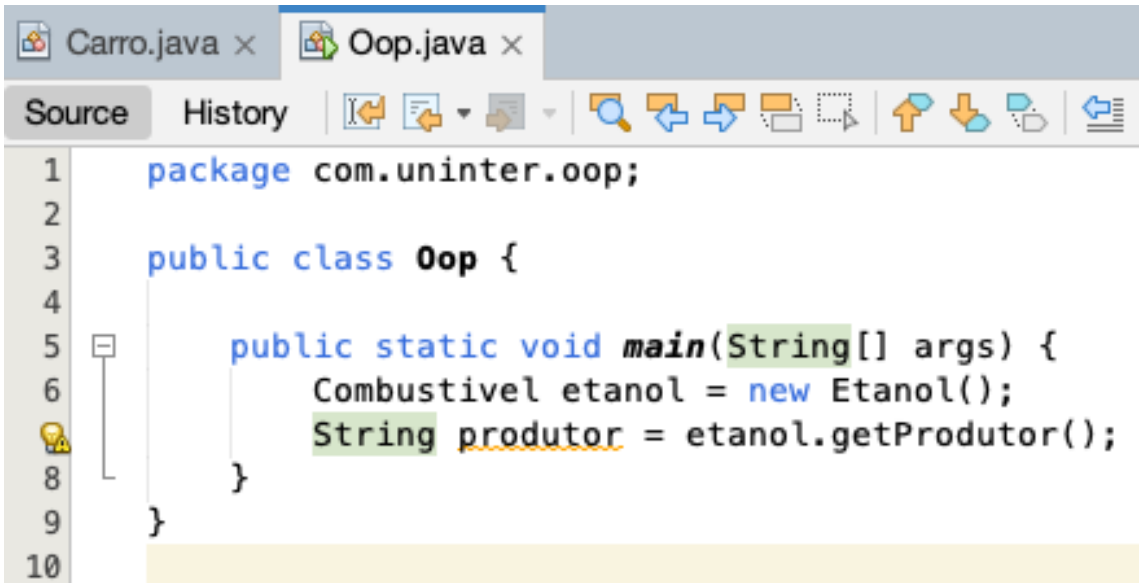
```
package com.uninter.oop;

public class Oop {

  public static void main(String[] args) {
    Combustivel etanol = new Etanol();
```

```
        String produtor = etanol.getProdutor();
    }
}
```

Imagem da classe Oop demonstrando herança, polimorfismo e encapsulamento no NetBeans.



Lembre que a classe Etanol não possui propriedade nenhuma, mas ao estender a classe Combustivel, passa a ter acesso a todas as propriedades da classe. Ao instanciar a classe Etanol num objeto Combustivel demonstramos polimorfismo e ao utilizar o método *getProdutor()* demonstramos encapsulamento e herança.

Ilustra-se assim a eficácia da programação orientada a objetos, especialmente em softwares compostos por dezenas, centenas ou até mesmo milhares de classes. A manutenção e a legibilidade do código são significativamente aprimoradas por meio dessa abordagem.

Indagar os participantes da aula com relação à qualidade da aula e eventuais conhecimentos que algum aluno possa ter que seja relevante ao tema, oferecendo recompensa as menções, como por exemplo UML.

Anunciar a classificação final do quadro de líderes e distribuir troféus e medalhas de sucesso aos melhores desempenhos.

O aluno com menor pontuação ganha um troféu e 50 pontos como recompensa pela perseverança.

Oferecer recompensas a estudantes de alto nível, tais como recursos adicionais, pequenos prêmios ou pontos de bônus para futuras lições.

Os elementos de gamificação incluídos neste plano de aula são pontos, troféus e medalhas, tabelas de liderança e recompensas. Incentivar a competição amigável e a colaboração entre os estudantes, o que pode melhorar ainda mais sua experiência de aprendizagem.

Para encerrar, o esquema de pontuação padrão aplicado para todos os temas ou módulos da aula podem ser representados como orientação a objetos. A gamificação seria o primeiro objeto. A aula estende gamificação. O módulo ou tema estende a aula.

Guia rápido de recompensas da aula:

Medalha	**Pontuação**	**Descrição da atividade**
Troféu	50 pontos	No término da aula auxiliou o maior número de colegas
Troféu	50 pontos	No término da aula com menor pontuação: perseverança
Ouro	10 pontos	Concluiu todas as atividades sem ajuda
Ouro	10 pontos	Concluiu a atividade ou desafio sem ajuda
Ouro	10 pontos	No término da aula auxiliou 5 ou mais colegas
Prata	6 pontos	Concluiu a atividade ou desafio com ajuda de um colega
Prata	6 pontos	No término da aula auxiliou 3 colegas
Bronze	3 pontos	Concluiu a atividade ou desafio com ajuda do professor
Bronze	3 pontos	No término da aula auxiliou 1 colega
Bronze	3 pontos	Criou uma classe, objeto, pacote mais criativo
Bronze	3 pontos	Criou uma classe, objeto, pacote menos criativo
Bronze	3 pontos	Instalou OpenJDK primeiro
Bronze	3 pontos	Instalou Apache NetBeans primeiro