

Unidade 1

Ambiente de Desenvolvimento e Teste de Software



Introdução

Iniciaremos nosso estudo na disciplina de Análise e Projeto de Algoritmos, apresentando o que iremos estudar nesta unidade.

Logo após a definição da linguagem de programação, no qual será desenvolvido nosso código, é sumariamente importante definirmos um Ambiente de Desenvolvimento. Com o surgimento e propagação de tecnologias e ferramentas no mercado de *DevOps*, os ambientes também estão se multiplicando. É comum você se deparar com no mínimo três opções de Ambiente de Desenvolvimento para cada linguagem. Outro termo bastante utilizado para nomear um ambiente é a palavra IDE - *Integrated Development Enviroment*, que significa Ambiente de Desenvolvimento Integrado. Na aula 1 (um), você terá mais detalhes e opções de alguns ambientes para desenvolvimento na linguagem de programação Java.

Um outro assunto que será abordado nessa unidade de ensino, diz respeito a Programação com testes e *debugging*. Vale lembrar que Teste ou Teste de *Software*, além de ser uma área dentro do desenvolvimento de *software*, ainda é uma etapa do ciclo de vida básico do processo. Essa etapa geralmente é realizada após algumas etapas anteriores serem concretizadas, como o levantamento de requisitos, a análise de sistemas e a etapa de implementação, etapa que acontece a codificação do sistema. No entanto, o teste que estamos nos referindo é o teste por parte do desenvolvedor.

Este tipo de teste está ligado a etapa de implementação do *software*. Se trata do teste natural que o desenvolvedor realiza várias vezes de várias formas para que ele possa dar o seu trabalho como concretizado. Só então, ele finalizará a atividade, colocando-a à disposição da equipe para que seja iniciado o ciclo de testes.

No decorrer da unidade de ensino serão abordados eventualmente outros conteúdos fundamentais para o processo de desenvolvimento de *software*.

Objetivos

- Conhecer as Ferramentas para Análise e Projeto de Algoritmos;
- Conhecer e utilizar uma IDE - Ambientes de Desenvolvimento Integrado;
- Conhecer as técnicas existentes de *debugging*;
- Compreender e Aplicar um Teste de Mesa em um fragmento ou código-fonte.

Conteúdo programático

Aula 01 – Ferramentas e Ambientes de Desenvolvimento

Aula 02 – Testes e *Debugging*.



Você poderá também **assistir às videoaulas** em seu celular! Basta apontar a câmera para os **QR Codes** distribuídos neste conteúdo.

Pode ser necessário instalar um aplicativo de leitura QRcode no celular

Aula 01 – Ferramentas e Ambientes de Desenvolvimento

Introdução



Videoaula Apresentação

Utilize o QRcode para assistir!



Videoaula Mini Currículo

Utilize o QRcode para assistir!



O objetivo dessa aula é apresentar alguns ambientes de desenvolvimento para a linguagem de programação Java. A linguagem Java será adotada nessa disciplina para facilitar o seu aprendizado, visto que na disciplina de Lógica de Programação e Algoritmos e na disciplina de Algoritmos e Estrutura de Dados essa linguagem já foi abordada.

A linguagem Java é uma das linguagens mais avançadas da atualidade. Uma linguagem de alto nível, que atua em cima do paradigma orientado a objetos que nos permite programar mais próximo de como seria representado na vida real. A sintaxe do Java é derivada do C e C++, que facilita a compreensão e o suporte devido à grande diversidade de desenvolvedores dessas linguagens.

O primeiro fator que começa a definir a IDE que será utilizada, é a linguagem de programação. Nesse momento, como já temos a linguagem, vamos abordar 3 (três) ferramentas de trabalho, ou três ambientes diferentes para o desenvolvimento do sistema. O nível do sistema desenvolvido depende do tamanho e complexidade. Por exemplo, você pode desenvolver uma aplicação comercial completa, um sistema de análise estatística, ou um simples exemplo contendo somente algumas funcionalidades.

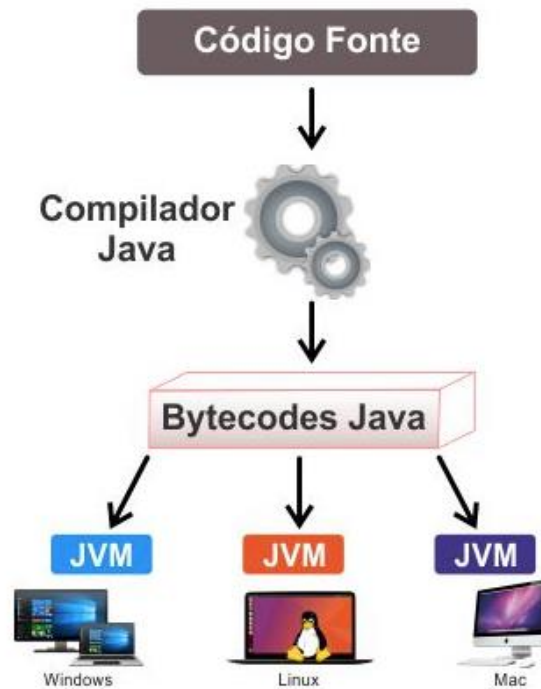
Uma das justificativas de se aprender Java e compreender a sua estrutura, é que o Java é rápido, seguro e confiável. Está presente em *datacenters*, *consoles* de *games*, supercomputadores de análise científica, *smartphones* e aparelhos eletrodomésticos como TVs entre outros.

Antes de instalar e começar a escrever os seus códigos na linguagem Java, é preciso fazer a instalação do JDK – *Java Development Kit*. O JDK é um conjunto de ferramentas que o desenvolvedor utiliza para desenvolvimento. Quando se faz o *download* de um JDK, você obtém, além do compilador e outras ferramentas, uma biblioteca de classe completa contendo utilitários, pré-construção que o ajuda a realizar tarefas de desenvolvimento de sistemas. O JDK de acordo com o seu sistema operacional está disponível no *site* da Oracle (disponível em: <https://www.oracle.com/java/technologies/javase-downloads.html>).

A *Java Virtual Machine* – JVM é responsável por ler e interpretar arquivos *.class*. Os arquivos com extensão *.class* são *bytecodes* gerados a partir de arquivos com extensão *.java*, que contêm os algoritmos escritos na sintaxe Java. Os *Bytecodes* são conjuntos de instruções destinadas a executar em uma JVM. A JVM atua como um interpretador, pois carrega o *bytecode* da aplicação compilada em Java, interpreta esse código para a arquitetura do computador que está sendo usado e depois executa a aplicação. Esse processo é realizado rapidamente como se a aplicação fosse nativa do sistema.

Por exemplo, considerando uma aplicação em Java implementada no sistema operacional Windows, essa mesma aplicação pode ser executada na plataforma MAC-OS ou no Windows. Quem torna tudo isso possível é a JVM, como pode ser observado na Figura 1.

Figura 1 – Esquema de funcionamento do Java



Fonte: adaptado de Deitel (2010)

Com o mesmo código *bytecode* compilado, é possível a execução por meio da JVM em Windows, Linux e MAC-OS.

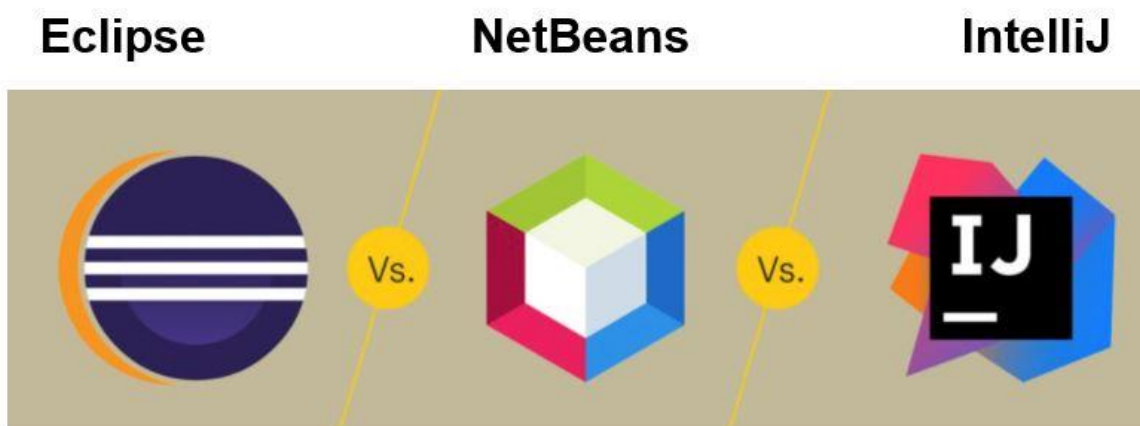
Ambientes de Desenvolvimento

O Ambiente de Desenvolvimento é a sua principal ferramenta na construção de um sistema computacional. Sendo assim, você deve levar várias características em conta para fazer a escolha certa, isso claro, se ela depende somente de você. O que acontece é que muitas vezes em empresas de *software*, o ambiente utilizado já está escolhido. Você só vai precisar se adaptar a ele.

Os 3 (três) ambientes ou IDEs que serão apresentadas nessa unidade de ensino são ferramentas de respaldo, que possuem todos os recursos necessários para o desenvolvimento de aplicações para qualquer plataforma, seja ela *desktop*, *web* ou *mobile*.

Para a construção e customização de sistemas desenvolvidos na linguagem de programação Java, os ambientes de desenvolvimento integrados que vamos apresentar nessa aula são:

Figura 2 – Ambientes para desenvolvimento em Java



Fonte: disponível em: <https://fiodevida.com/eclipse-vs-netbeans-vs-intellij-idea>. Acesso em: 4 jul. 2021.
(adaptado)

Na figura 2, estão os mais conhecidos e difundidos ambientes de desenvolvimento para a linguagem Java. O IntelliJ é um pouco mais novo popularmente, possui o mesmo fabricante do Android Studio, uma ferramenta criada para desenvolvimento exclusivamente para Android. Nas últimas versões, já é possível desenvolver projeto na linguagem Kotlin.

Indicação de Leitura

Para complementar sua leitura, você poderá recorrer ao artigo publicado em Maio de 2021 pela *FiodeVida*. No artigo é possível encontrar aspectos como características, vantagens e principais recursos dos ambientes.

O material que vai encontrar aqui, com certeza vai ajudar você a formar sua ideia sobre essas ferramentas. Vale a pena conferir!

Disponível em: <https://fiodevida.com/eclipse-vs-netbeans-vs-intellij-idea/>. Acesso em: 4 jul. 2021.

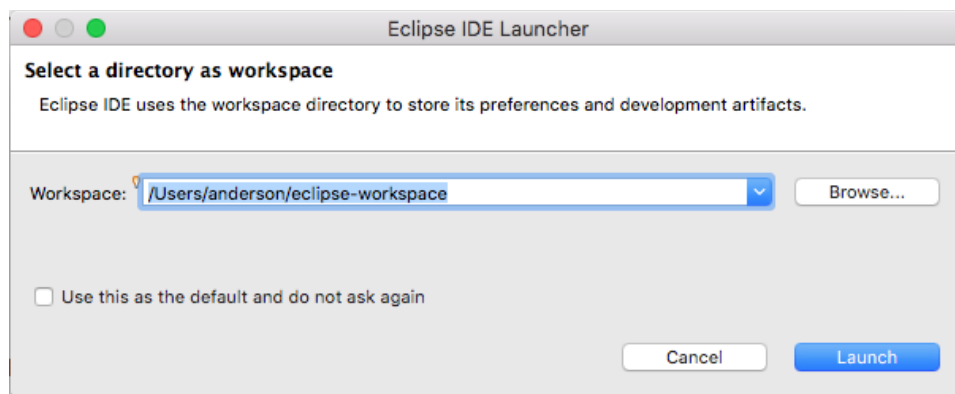
Eclipse

Para começar o desenvolvimento de programas em Java por meio da plataforma do Eclipse comece realizando o *download* e instalando a ferramenta. Acesse o site oficial (<https://www.eclipse.org/downloads/>) e obtenha a ferramenta. Não se esqueça de selecionar o S.O. correspondente ao seu computador ou dispositivo.

Criado em 2001, pela IBM, é uma plataforma de código aberto. Utilizada amplamente nas disciplinas de programação em Universidades e no desenvolvimento de sistemas em empresas, pela usabilidade e por oferecer suporte em outras linguagens de programação.

Após a instalação do Eclipse, basta abrir o executável no caminho onde foi instalado. O próximo passo é configurar o *Workspace*. Essa é a primeira configuração solicitada pelo Eclipse. Observe a figura 3.

Figura 3 – Definição do *Workspace*

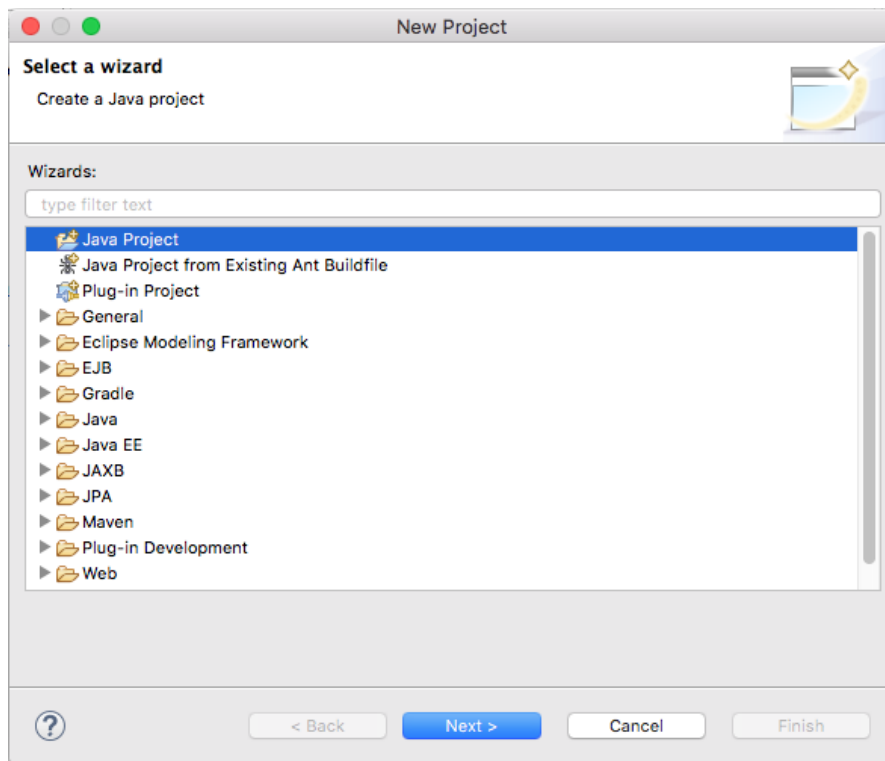


Fonte: elaborado pelo autor via IDE Eclipse (2021)

O *workspace* é o local em que suas configurações pessoais e seus projetos serão gravados. Basta defini-lo para que a tela principal do Eclipse seja carregada. Você pode também definir o *workspace* como padrão.

Vamos agora criar um projeto no Eclipse para a linguagem Java. Para isso, vá em *File / New Project* e selecione: *Java Project*, como está demonstrado na figura 4.

Figura 4 – Criando um projeto no Eclipse



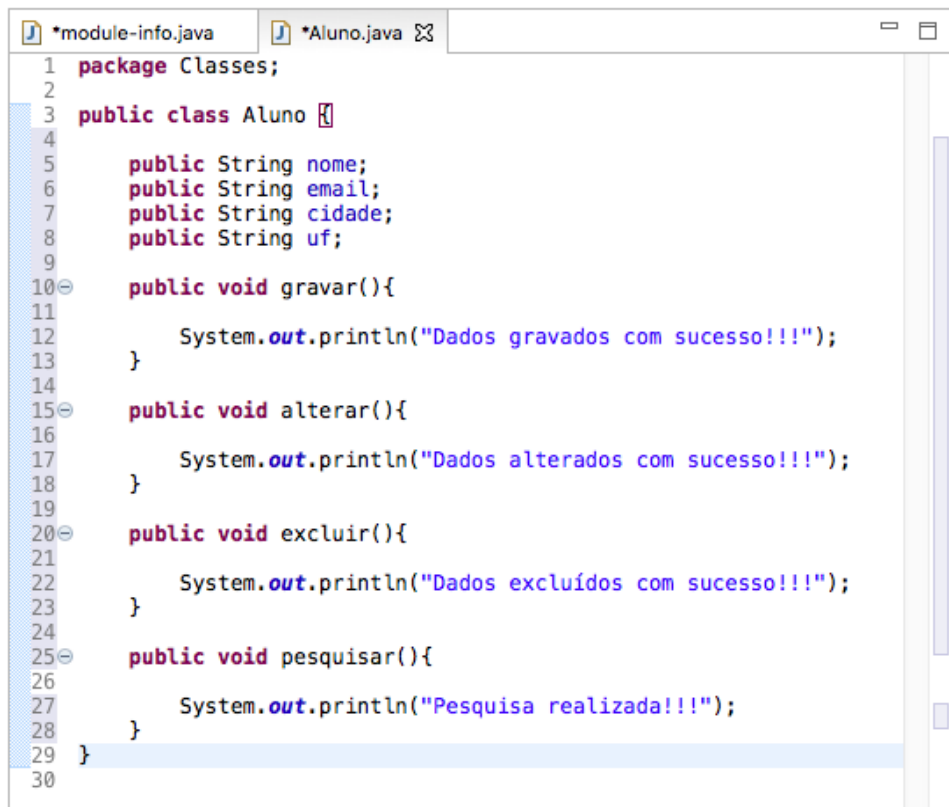
Fonte: elaborado pelo autor via IDE Eclipse (2021)

Na próxima tela, dê um nome para o projeto e clique em *Next*. Algumas configurações serão exibidas na tela seguinte, porém não é necessário alterar nada. Clique em *Finish* para o Eclipse criar o seu projeto. Agora você tem um projeto Java dentro do Eclipse. Para começar a programar nesse projeto basta criar uma classe e iniciar o desenvolvimento. Vamos fazer um exemplo simples, criando uma classe com o nome de Aluno com alguns atributos e operações. Vamos ao exemplo.

No Eclipse, crie uma classe dentro de **src**, local inicial no projeto onde geralmente são adicionadas as classes. Não se esqueça de criar um pacote para armazenar as classes.

Dessa maneira, adicione os atributos e operações (métodos), conforme figura 5.

Figura 5 – Classe Aluno

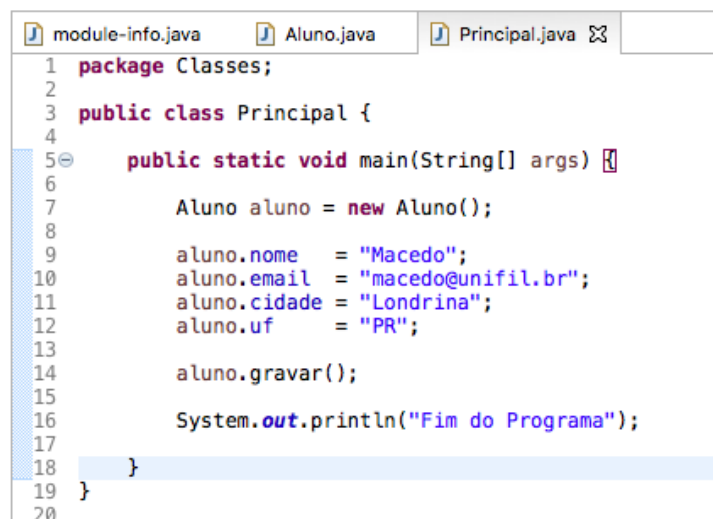


```
1 package Classes;
2
3 public class Aluno {
4
5     public String nome;
6     public String email;
7     public String cidade;
8     public String uf;
9
10    public void gravar(){
11        System.out.println("Dados gravados com sucesso!!!");
12    }
13
14    public void alterar(){
15        System.out.println("Dados alterados com sucesso!!!");
16    }
17
18    public void excluir(){
19        System.out.println("Dados excluídos com sucesso!!!");
20    }
21
22    public void pesquisar(){
23        System.out.println("Pesquisa realizada!!!");
24    }
25 }
26
27
28
29
30
```

Fonte: elaborado pelo autor via IDE Eclipse (2021)

Agora vamos criar outra classe que será responsável por instanciar a classe Aluno e executar o método gravar(). Deixe a classe idêntica à figura 6.

Figura 6 – Classe Principal



```
1 package Classes;
2
3 public class Principal {
4
5     public static void main(String[] args) {
6
7         Aluno aluno = new Aluno();
8
9         aluno.nome = "Macedo";
10        aluno.email = "macedo@unifil.br";
11        aluno.cidade = "Londrina";
12        aluno.uf = "PR";
13
14        aluno.gravar();
15
16        System.out.println("Fim do Programa");
17    }
18 }
19
20
```

Fonte: elaborado pelo autor via IDE Eclipse (2021)

Na Figura 6 foi criada a classe Principal que faz a instância da classe Aluno por meio o objeto “**aluno**”. Nesse ponto é possível acessar os atributos e métodos que foram definidos de forma pública na classe Aluno, figura 5. Desse modo, é possível executar o programa e ver as mensagens no Console como é demonstrado na figura 7.

Figura 7 – Execução no Eclipse

```
<terminated> Principal [Java Application] /Users/anderson/.g
Dados gravados com sucesso!!!
Fim do Programa
```

Fonte: elaborado pelo autor via IDE Eclipse (2021)

Videoaula 1

A primeira videoaula abordará os assuntos introdutórios e a codificação do nosso exemplo a partir do ambiente de desenvolvimento Eclipse. Neste vídeo, vamos abordar a linguagem de programação Java.



Videoaula 1

Utilize o QRcode para assistir!

A primeira videoaula abordará os assuntos introdutórios e a codificação do nosso exemplo a partir do ambiente de desenvolvimento Eclipse. Neste vídeo, vamos abordar a linguagem de programação Java.



Para demonstração dos ambientes NetBeans e IntelliJ, será utilizado o mesmo exemplo para facilitar a comparação da construção do projeto.

NetBeans

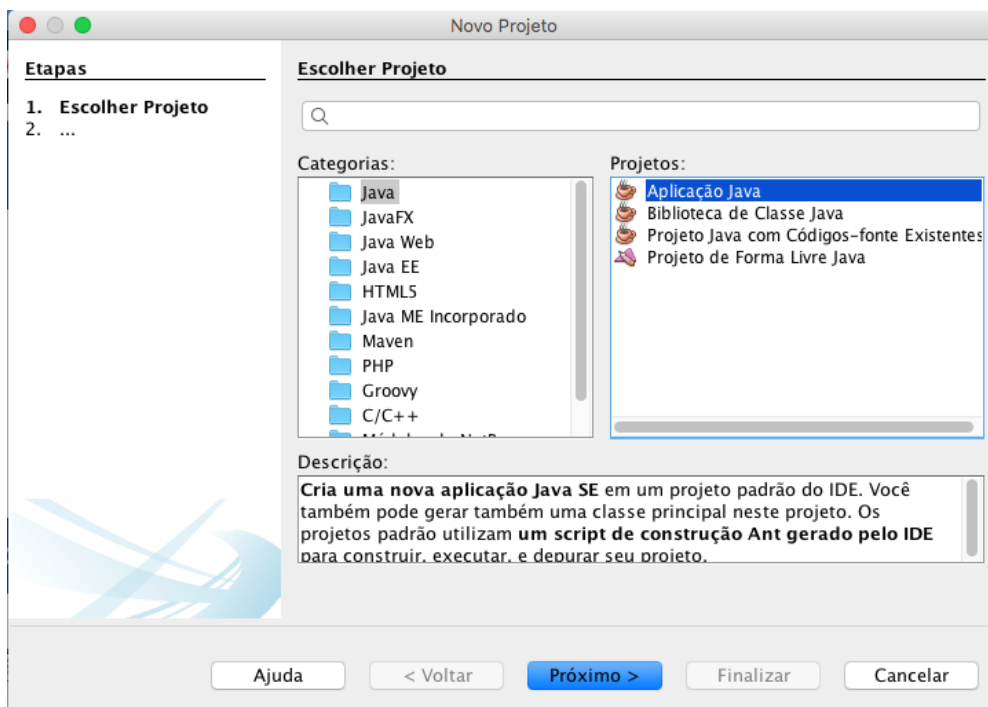
O NetBeans também é um ambiente para desenvolvimento em Java *OpenSource*, ou seja, de código aberto bem popular entre os desenvolvedores Java. Para instalação, acesse o site oficial (<https://www.oracle.com/java/technologies/javase-jdk-7-netbeans-downloads.html>) e obtenha a ferramenta. Similarmente ao Eclipse, o NetBeans também é oferecido para sistemas operacionais Windows, Linux e Mac OS. Não se esqueça de selecionar o S.O. correspondente ao seu computador ou dispositivo. Uma característica interessante, é que esse *link* já disponibiliza, em conjunto com a ferramenta, uma versão do *Java Development Kit* – JDK.

Criado na década de 1990, acabou evoluindo juntamente com a própria linguagem Java. Surgiu como uma plataforma de código aberto e depois foi adquirido pela Sun em 1999. Posteriormente foi adquirido pela empresa Oracle. Agora, ela pode ser utilizada no desenvolvimento de todas as distribuições do Java, entre elas está o Java EE. Assim como o Eclipse, o NetBeans também é utilizado na comunidade acadêmica e empresarial.

Uma das grandes diferenças entre essas duas plataformas são os *plug-ins*. Enquanto as duas plataformas possuem uma variedade enorme dessas extensões, o NetBeans já vem mais completo, isso evita a instalação e configuração a todo momento. Uma desvantagem é que por ser mais completo, acaba sendo mais pesado, exigindo maior processamento por parte do computador.

Vamos replicar o exemplo no NetBeans. Para tanto, vá em Arquivo / Novo Projeto e selecione Aplicação Java, como está demonstrado na Figura 8.

Figura 8 – Criando um projeto no NetBeans

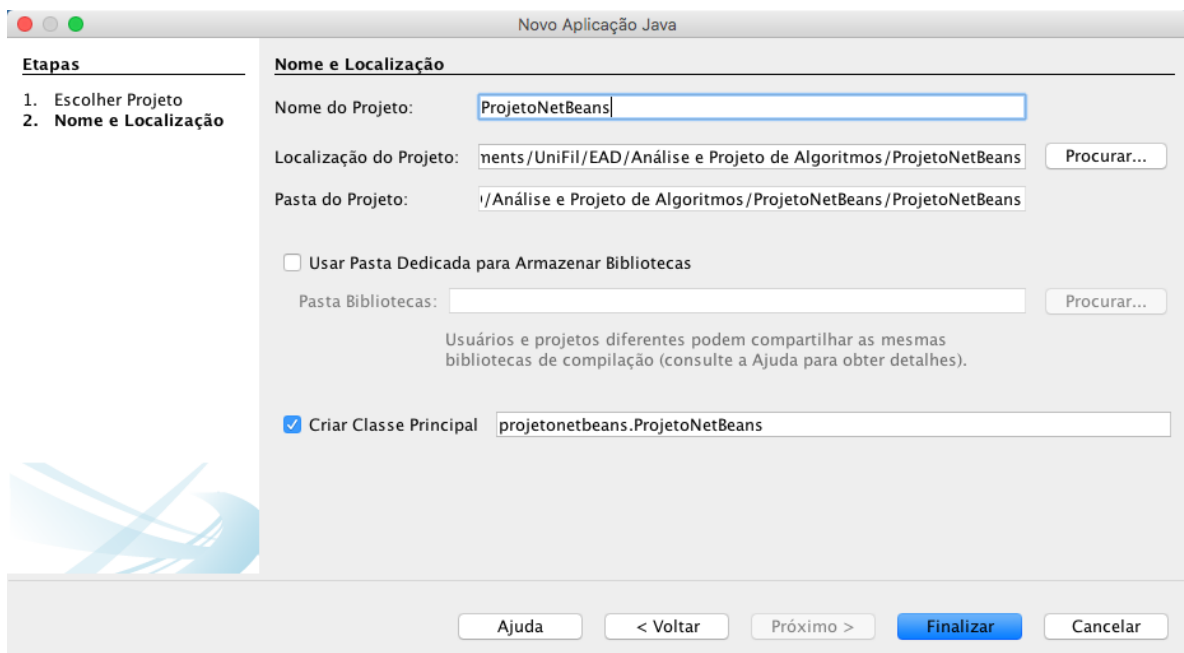


Fonte: elaborado pelo autor via IDE NetBeans (2021)

Na próxima tela, é necessário escolher o nome do projeto e sua localização. Existe também a opção de criação da classe Principal do projeto. Você pode deixar essa opção marcada. Entre outras opções e configurações, também pode-se alterar o diretório principal do projeto.

Clicando no botão “Finalizar”, o projeto é criado e você terá todas as possibilidades para criar um sistema de acordo com as regras de negócio e cenário proposto. A figura 9 apresenta a IDE com o projeto aberto.

Figura 9 – Configurações do projeto no NetBeans



Fonte: elaborado pelo autor via IDE NetBeans (2021)

Agora vamos criar a classe Aluno do nosso exemplo no projeto NetBeans.

Figura 10 – Classe Aluno

```
12 public class Aluno {
13
14     public String nome;
15     public String email;
16     public String cidade;
17     public String uf;
18
19     public void gravar() {
20
21         System.out.println("Dados gravados com sucesso!!!");
22     }
23
24     public void alterar() {
25
26         System.out.println("Dados alterados com sucesso!!!");
27     }
28
29     public void excluir() {
30
31         System.out.println("Dados excluídos com sucesso!!!");
32     }
33
34     public void pesquisar() {
35
36         System.out.println("Pesquisa realizada!!!");
37     }
38 }
```

Fonte: elaborado pelo autor via IDE NetBeans (2021)

A classe principal desse projeto já foi adicionada quando ele foi criado. Basta agora adicionar a instância para acessar as características e comportamentos da classe Aluno. Tal procedimento pode ser observado na figura 11.

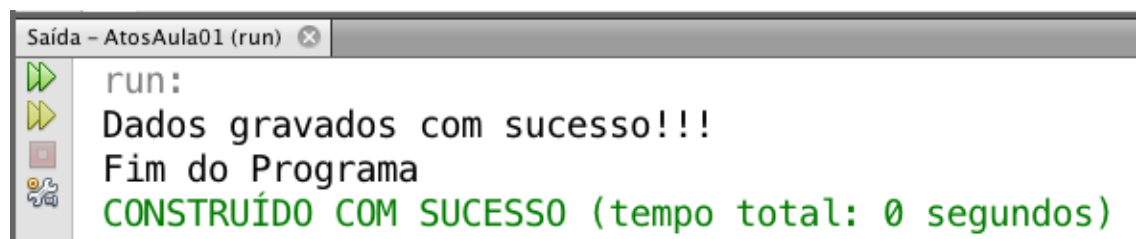
Figura 11 – Classe Principal

```
12 public class ProjetoNetBeans {
13
14     public static void main(String[] args) {
15
16         Aluno aluno = new Aluno();
17
18         aluno.nome = "Macedo";
19         aluno.email = "macedo@unifil.br";
20         aluno.cidade = "Londrina";
21         aluno.uf = "PR";
22
23         aluno.gravar();
24
25         System.out.println("Fim do Programa");
26
27     }
28
29 }
```

Fonte: elaborado pelo autor via IDE NetBeans (2021)

A exemplo do Eclipse, é possível executar o programa e ver as mensagens no Console como é demonstrado na figura 12.

Figura 12 – Execução do NetBeans



```
Saída - AtosAula01 (run) x
run:
Dados gravados com sucesso!!!
Fim do Programa
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Fonte: elaborado pelo autor via IDE NetBeans (2021)

Videoaula 2

A segunda videoaula abordará a configuração e criação de um projeto em Java. O ambiente de desenvolvimento NetBeans será utilizado para o desenvolvimento da aula. Neste vídeo, o professor irá explicar o funcionamento da aplicação.



Videoaula 2

Utilize o QRcode para assistir!

A segunda videoaula abordará a configuração e criação de um projeto em Java. O ambiente de desenvolvimento NetBeans será utilizado para o desenvolvimento da aula. Neste vídeo, o professor irá explicar o funcionamento da aplicação.



O NetBeans é um ambiente muito poderoso e repleto de recursos para criação de sistemas com toda a tecnologia oferecida por meio do Java. Agora vamos para a demonstração do terceiro ambiente, o IntelliJ.

IntelliJ

O IntelliJ oferece suporte para linguagens além do Java. Entre elas estão Scala, Groovy e Clojure. A versão comercial Ultimate, voltada para o setor empresarial, oferece suporte a SQL, ActionScript, Ruby, Python e PHP. A versão 12, acompanha um designer de UI Android para o desenvolvimento de aplicativo móveis, bem similar ao Android Studio já comentado neste material.

Um ponto forte dessa ferramenta é a quantidade de *plug-ins* em sua versão corporativa. São mais de 50 *plug-ins* disponíveis usando os componentes *Swing* integrados na plataforma.

Para instalação, acesse o site oficial (<https://www.jetbrains.com/pt-br/idea/>) e obtenha a ferramenta. Similarmente ao Eclipse e NetBeans, o IntelliJ também é oferecido para sistemas operacionais Windows, Linux e Mac OS. Não se esqueça de selecionar o S.O. correspondente ao seu computador ou dispositivo. A versão Ultimate está disponível,

mas somente para avaliação durante 30 dias. A versão *Community* pode ser baixada e utilizada normalmente. A figura 13 mostra as opções de *download*.

Figura 13 – Opções de *download* para o IntelliJ

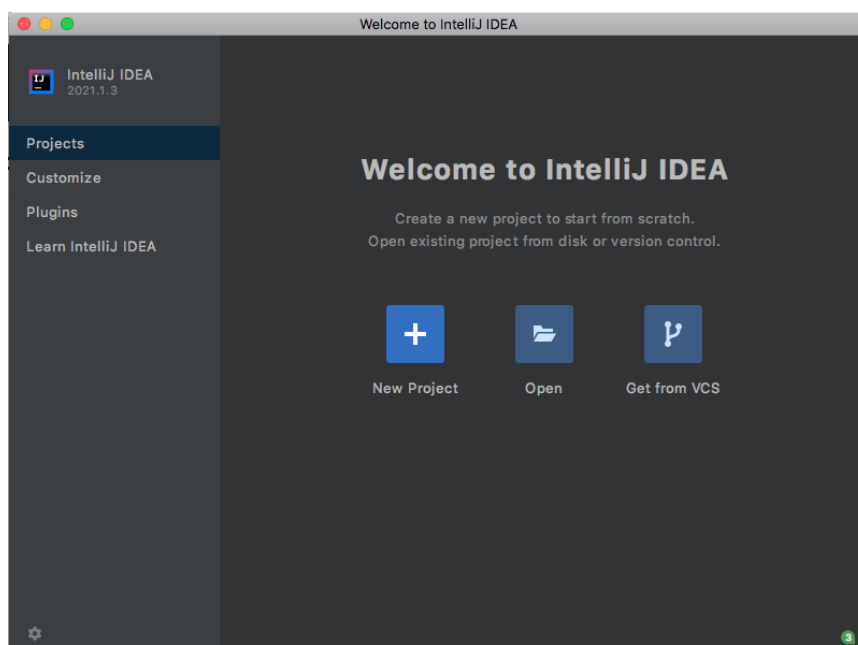


Fonte: elaborado pelo autor via IDE IntelliJ (2021)

Vamos replicar o exemplo no IntelliJ. A versão 2021.1.3, trouxe várias modificações. Consulte as principais no *site* da Jet Brains (<https://www.jetbrains.com/pt-br/idea/whatsnew/>).

Para a criação do projeto, basta clicar em *New Project*.

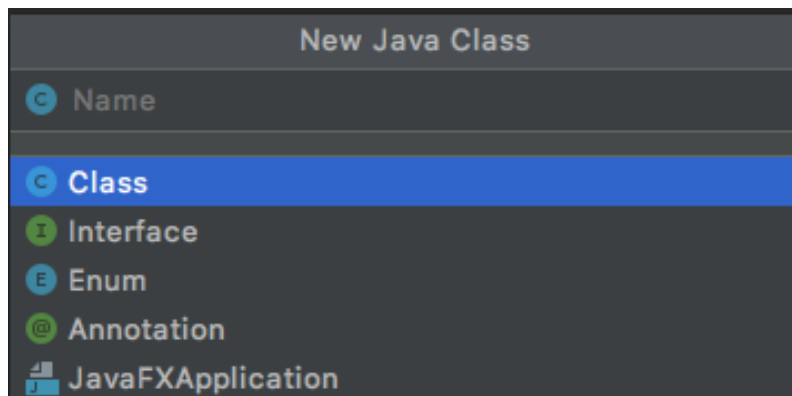
Figura 14 – Criando um projeto no IntelliJ



Fonte: elaborado pelo autor via IDE IntelliJ (2021)

Após a criação do projeto, o próximo passo é a criação da classe Aluno. No mesmo diretório em que a classe “Main” está criada, adicione uma nova classe.

Figura 15 – Criando um projeto no IntelliJ



Fonte: elaborado pelo autor via IDE IntelliJ (2021)

A classe Aluno deve ser criada no projeto e adicionado os códigos para a criação dos atributos e dos métodos do nosso exemplo como mostra a figura 16.

Figura 16 – Classe Aluno

```
3 public class Aluno {  
4  
5     public String nome;  
6     public String email;  
7     public String cidade;  
8     public String uf;  
9  
10    public void gravar(){  
11        System.out.println("Dados gravados com sucesso!!!");  
12    }  
13    public void alterar(){  
14        System.out.println("Dados alterados com sucesso!!!");  
15    }  
16    public void excluir(){  
17        System.out.println("Dados excluídos com sucesso!!!");  
18    }  
19    public void pesquisar(){  
20        System.out.println("Pesquisa realizada!!!");  
21    }  
22 }  
23  
24  
25  
26 }
```

Fonte: elaborado pelo autor via IDE IntelliJ (2021)

A classe principal desse projeto já foi adicionada quando ele foi criado. Basta agora adicionar a instância para que a classe Principal tenha acesso às características e comportamentos da classe Aluno. Tal procedimento pode ser observado na figura 17.

Figura 17 – Classe Principal

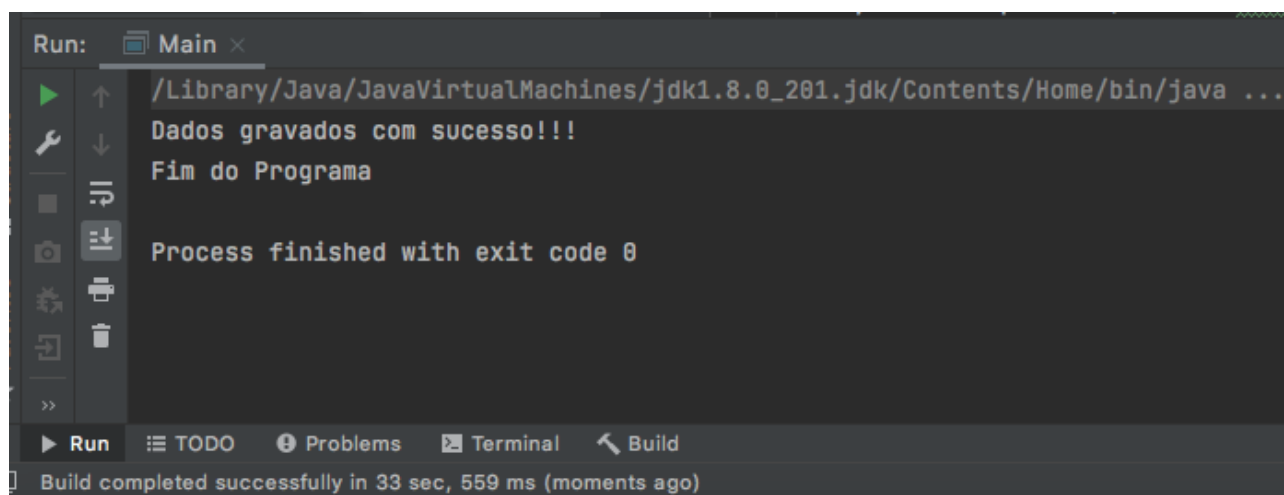


```
public class Main {  
  
    public static void main(String[] args) {  
        // write your code here  
        Aluno aluno = new Aluno();  
  
        aluno.nome = "Macedo";  
        aluno.email = "macedo@unifil.br";  
        aluno.cidade = "Londrina";  
        aluno.uf = "PR";  
  
        aluno.gravar();  
  
        System.out.println("Fim do Programa");  
    }  
}
```

Fonte: elaborado pelo autor via IDE IntelliJ (2021)

A exemplo do Eclipse e do NetBeans, é possível executar o programa e ver as mensagens no Console como é demonstrado na figura 18.

Figura 18 – Execução do IntelliJ



Fonte: elaborado pelo autor via IDE IntelliJ (2021)

Videoaula 3

A terceira videoaula abordará o projeto sendo apresentado por meio da ferramenta IntelliJ. Na videoaula, o professor vai explicar o funcionamento da aplicação.



Videoaula 3

Utilize o QRcode para assistir!

A terceira videoaula abordará o projeto sendo apresentado por meio da ferramenta IntelliJ. Na videoaula, o professor vai explicar o funcionamento da aplicação.



O IntelliJ é uma poderosa ferramenta. Graças a sua usabilidade e facilidade de parametrizações, foi adotada pela comunidade desenvolvedora e vem conquistando espaço no mercado em desenvolvimento de aplicações Java.

Aula 02 – Testes e Debugging

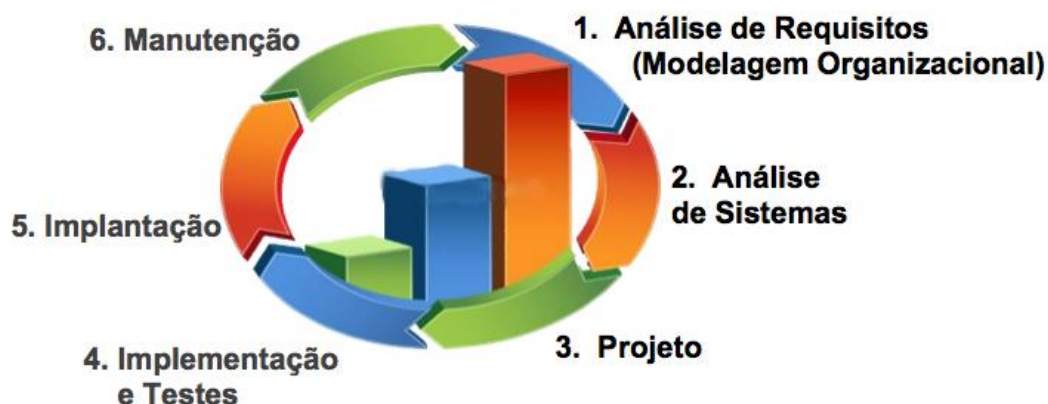
Introdução

Nessa aula, iniciaremos conceituando um pouco essa área de Teste, porém abordar sobre Teste do Desenvolvedor e não é a mesma coisa que a etapa de Teste de *Software*, como já diferenciado no início da Unidade de Ensino.

O teste do desenvolvedor faz parte da etapa de implementação do código, ou seja, a codificação do sistema. Contextualizando na forma de uma atividade prática, que um desenvolvedor desempenha todos os dias, o teste do desenvolvedor ou de desenvolvimento, seria garantir que o seu trabalho foi realizado corretamente. Portanto, tem que estar de acordo com o requisito levantado.

No processo de desenvolvimento de *software* figura 1, é possível visualizar algumas etapas para que a funcionalidade seja concluída, por exemplo: Digamos que um relatório de clientes foi solicitado pelo usuário. Seguindo a metodologia de “Desenvolvimento Iterativo”, é preciso passar por todas as etapas do processo de desenvolvimento para que a atividade seja concluída.

Figura 1 – Processo de Desenvolvimento



Fonte: o autor (2021)

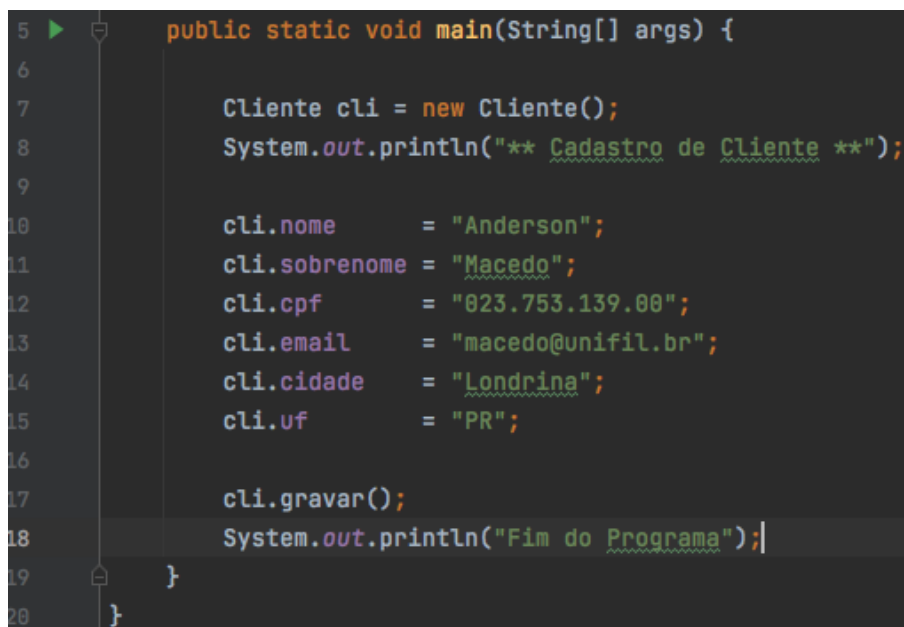
Isso significa que primeiramente vem a etapa de “Análise de Requisitos”, essa etapa é onde se faz a coleta de todas as informações para criação do relatório. Quais os campos

serão exibidos, quem são os clientes que entraram nesse relatório ou ainda se isso será decidido pelo usuário no momento em que ele estiver gerando ele no sistema. Em outras palavras, se o relatório será parametrizado ou não.

Como podemos observar, todo o trabalho de assertividade entre o que está sendo construído e o que foi desejado, começa exatamente no levantamento de requisitos. Após essa etapa concluída segue-se para as próximas até que se chegue na etapa de “Implementação”. Dentro dessa etapa será realizado o “teste do desenvolvedor”, etapa esta, em que estamos abordando nesse momento.

Na figura 2, será apresentado um código em Java simples para que o desenvolvedor teste o resultado do seu código. Para que isso funcione, temos que ter o requisito em mãos. O requisito é algo analisado e documentado em forma de diagramas pela *Unified Modeling Language* – UML. Tudo isso, auxilia a equipe técnica no controle e gerenciamento das atividades do sistema.

Figura 2 – Código do Cadastro de Cliente

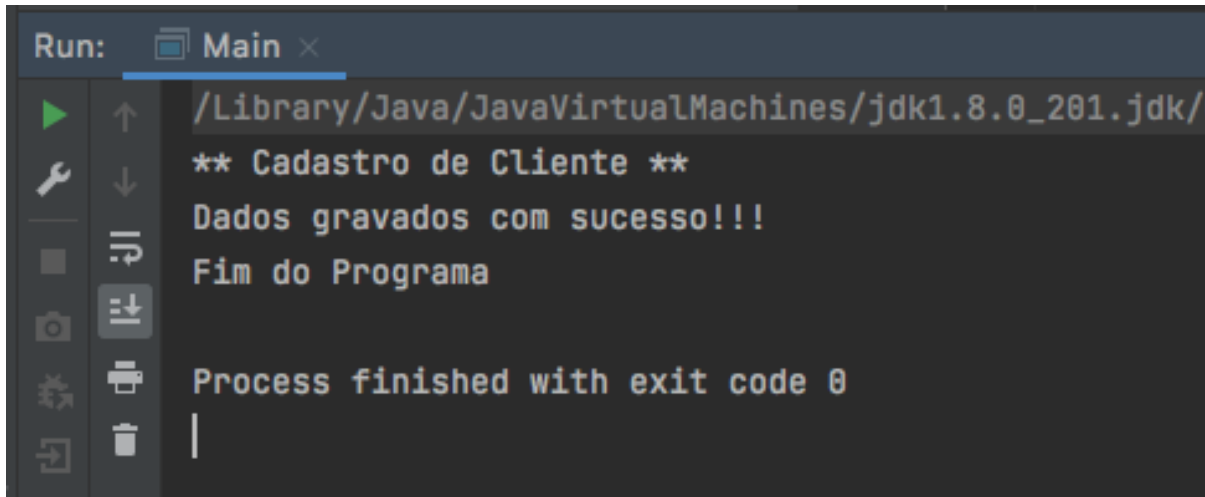


```
5 ▶ public static void main(String[] args) {  
6  
7     Cliente cli = new Cliente();  
8     System.out.println("** Cadastro de Cliente **");  
9  
10    cli.nome      = "Anderson";  
11    cli.sobrenome = "Macedo";  
12    cli.cpf       = "023.753.139.00";  
13    cli.email     = "macedo@unifil.br";  
14    cli.cidade    = "Londrina";  
15    cli.uf        = "PR";  
16  
17    cli.gravar();  
18    System.out.println("Fim do Programa");  
19 }  
20 }
```

Fonte: elaborado pelo autor via IDE IntelliJ (2021)

Vamos pensar em um requisito para um cadastro simples de um cliente. Nesse cadastro será solicitado: o nome, sobrenome, CPF, *e-mail*, cidade e estado. Após a inserção dos dados pelo cliente o cadastro será realizado. Vamos a execução do código, apresentado na figura 3.

Figura 3 – Execução do Algoritmo



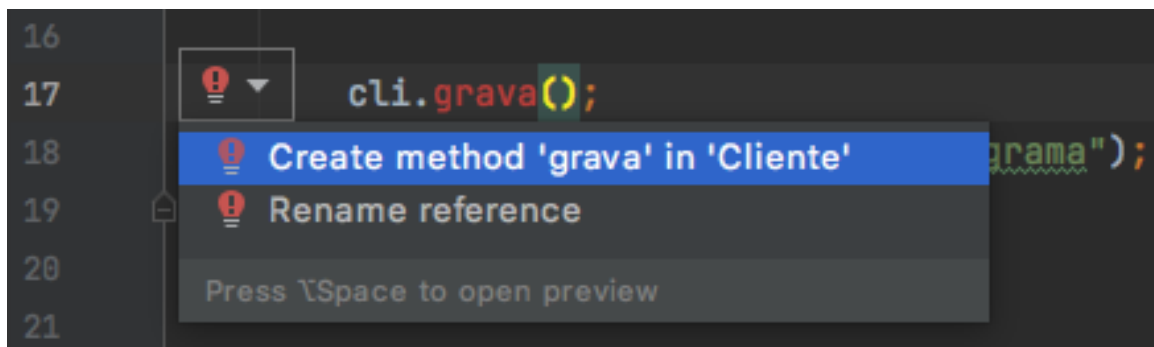
```
Run: Main x
/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/
** Cadastro de Cliente **
Dados gravados com sucesso!!!
Fim do Programa
Process finished with exit code 0
```

Fonte: elaborado pelo autor via IDE IntelliJ (2021)

É a partir desse teste que o desenvolvedor realiza, que podemos validar se o algoritmo construído por ele atende aos requisitos ou não.

Outro fator que também é identificado, é se, ou quando o código apresentar alguma falha na execução. Quando isso ocorre, chamamos de “erro de programação”. Na figura 4 um exemplo é apresentado.

Figura 4 – Erro de Programação



Fonte: elaborado pelo autor via IDE IntelliJ (2021)

Na figura 4, linha 17, a palavra “gravar” foi digitada faltando um caractere. Quando o código tenta ser compilado o erro é identificado e o ambiente pede para que seja corrigido sugerindo algumas ações por parte do desenvolvedor. O código desse exemplo foi gerado pelo ambiente IntelliJ.

Para maiores informações sobre as etapas do processo de desenvolvimento de um *software*, como funcionada e o que contêm dentro delas, acompanhe a indicação de leitura a seguir.

Indicação de Leitura

Na disciplina de Gerência de Requisitos, você vai conhecer o Processo Unificado para modelagem e documentação do sistema, o RUP. Para maiores informações confira o *site* abaixo:

Disponível em:
http://walderson.com/IBM/RUP7/LargeProjects/#core.base_rup/guidances/supportingmaterials/welcome_2BC5187F.html. Acesso em: 4 jul. 2021.

Aqui você vai encontrar maiores detalhes do funcionamento desse processo e a documentação completa para aprimorar seu conhecimento.

© Copyright IBM Corp. 1987, 2006. Todos os Direitos Reservados.

Videoaula 1

Nessa videoaula, será abordado o teste do desenvolvedor apresentado até aqui. Por meio de um exemplo prático será demonstrado algumas particularidades desse tipo de teste. O professor irá implementar um código em Java e abordará características e funcionalidades desse tipo de teste.

Acompanhe com atenção e não deixe de realizar no seu ambiente de desenvolvimento, todas as implementações, alterações no código e orientações de execução e o próprio teste do algoritmo.



Videoaula 1

Utilize o QRcode para assistir!

Nessa videoaula, será abordado o teste do desenvolvedor apresentado até aqui.



Testes e *Debugging*

O *debugging* ou o processo de depuração do código-fonte, consiste em: encontrar e reduzir defeitos em um aplicativo de *software* ou mesmo em *hardware* (MYERS, 1976).

O *debugging* no *software* é quando estamos em busca dos erros de programação para resolvê-los e aumentar a confiabilidade do código desenvolvido. Já o Teste, envolve vários procedimentos para que se tenha o sucesso almejado. Nesse sentido, uma ferramenta muito utilizada para auxiliar o desenvolvedor é o Teste de Mesa, por onde vamos continuar nosso estudo.

O Teste de Mesa no desenvolvimento de *software*, é o que chamamos na Matemática de “Prova Real”, ou seja, provar se realmente um cálculo está correto. Definindo essa prática, o teste de mesa é a execução passo a passo de qualquer algoritmo, linha após linha, verificando o que acontece com as variáveis e com o que está sendo apresentado pelo programa.

Por exemplo, em um algoritmo que se propõe a realizar o cálculo do quadrado de um número qualquer, fornecido pelo usuário, o teste de mesa vai tentar provar se realmente o algoritmo está realizando esse processo com todos os requisitos previamente levantados.

Na figura 5, o código desse exemplo é apresentado e na sequência falaremos de como proceder o teste de mesa.

Figura 5 – Quadrado de um número

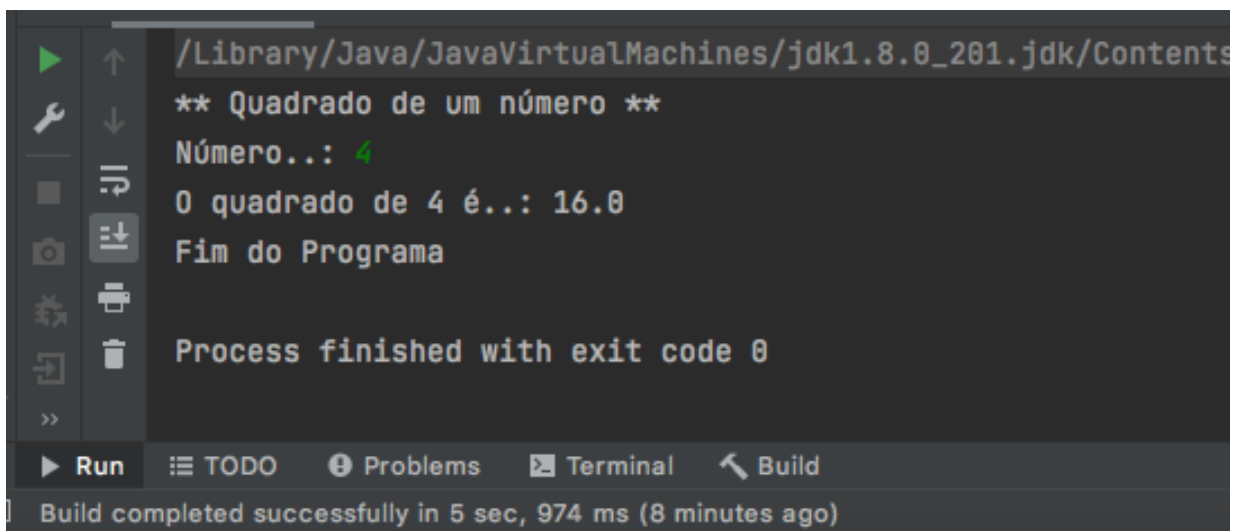
```
public static void main(String[] args) {  
  
    Scanner ler = new Scanner(System.in);  
    int num = 0;  
    double qua = 0;  
    System.out.println("** Quadrado de um número **");  
    System.out.print("Número..: ");  
    num = ler.nextInt();  
    qua = num * num;  
    System.out.println("O quadrado de " + num + " é..: " + qua);  
    System.out.println("Fim do Programa");  
}  
}
```

Fonte: elaborado pelo autor via IDE IntelliJ (2021)

Como podemos observar é um código simples que solicita do usuário um número, faz o cálculo do seu quadrado e após o processamento exibe na tela o resultado e uma mensagem de Fim do Programa.

A figura 6, apresenta a exibição da tela, do algoritmo sendo executado. Nessa execução, o usuário, que ao mesmo tempo é o próprio desenvolvedor, pois ele está testando o código que ele fez, forneceu o valor 4(quatro), em seguida o algoritmo apresentou o resultado 16 (dezesesseis).

Figura 6 – Execução do Algoritmo quadrado de um número



```
/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents  
** Quadrado de um número **  
Número..: 4  
O quadrado de 4 é..: 16.0  
Fim do Programa  
  
Process finished with exit code 0  
  
Run | TODO | Problems | Terminal | Build  
Build completed successfully in 5 sec, 974 ms (8 minutes ago)
```

Fonte: elaborado pelo autor via IDE IntelliJ (2021)

Para se fazer o teste de mesa desse exemplo, temos que saber quais e quantas variáveis eles contêm, depois disso anotá-las em um papel ou em um editor de textos e realizar as linhas de código uma após a outra e vendo o que acontece.

Nesse código existem somente duas variáveis, portanto não é tão difícil assim. Partindo para a execução passo a passo, vamos para a primeira linha que interfira realmente no teste de mesa. Ela acontece na linha 10 do código, veja a figura 7.

Figura 7 – Teste de mesa – parte 1

```
10      int    num = 0;  
11      double qua = 0;
```

Fonte: elaborado pelo autor via IDE IntelliJ (2021)

Na linha 10 e 11 as variáveis são inicializadas. A inicialização de variáveis é um procedimento que garante que as variáveis não estão ocupando em memória um valor de um outro aplicativo qualquer, também conhecido como sujeira de dados. Continuando, o valor 0 (zero) é inserido nas variáveis “num” e “qua”. Nas linhas 13 e 14 são realizadas as leituras das informações fornecidas pelo usuário para gravação na variável “num”, isto é, detectou-se neste trecho a mudança ou alteração nesta variável. Toda alteração nos valores das variáveis deve ser aplicada também no teste de mesa. Se o valor anterior da variável “num” era 0, agora ele passou a ser 4, considerando o nosso exemplo, figura 6. Na linha 15, o processamento é realizado gravando na variável “qua” o quadrado do número 4 e posteriormente na linha 16 esse valor é exibido em tela como mostra a figura 8.

Figura 8 – Teste de mesa – parte 2

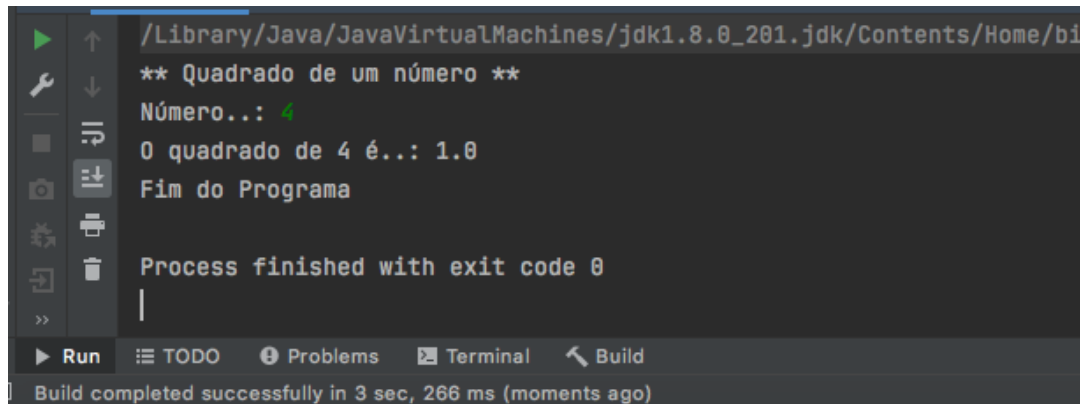
```
15      qua = num * num;  
16      System.out.println("O quadrado de " + num + " é.: " + qua);
```

Fonte: elaborado pelo autor via IDE IntelliJ (2021)

Nesse exemplo foi realizado o teste de mesa em um código que não apresentava nenhum erro de lógica de programação. Isso quer dizer que o algoritmo realmente fez o que estava planejado. Mas isso nem sempre acontece. Vamos alterar uma linha de código nesse mesmo algoritmo, porém sem alterar a ideia do seu funcionamento.

Dessa maneira, vamos continuar com o cenário de um algoritmo que solicita um valor numérico ao usuário e após isso apresenta na tela o seu quadrado. Ao executarmos o algoritmo, nos deparamos com a situação exibida na figura 9.

Figura 9 – Execução do algoritmo alterado

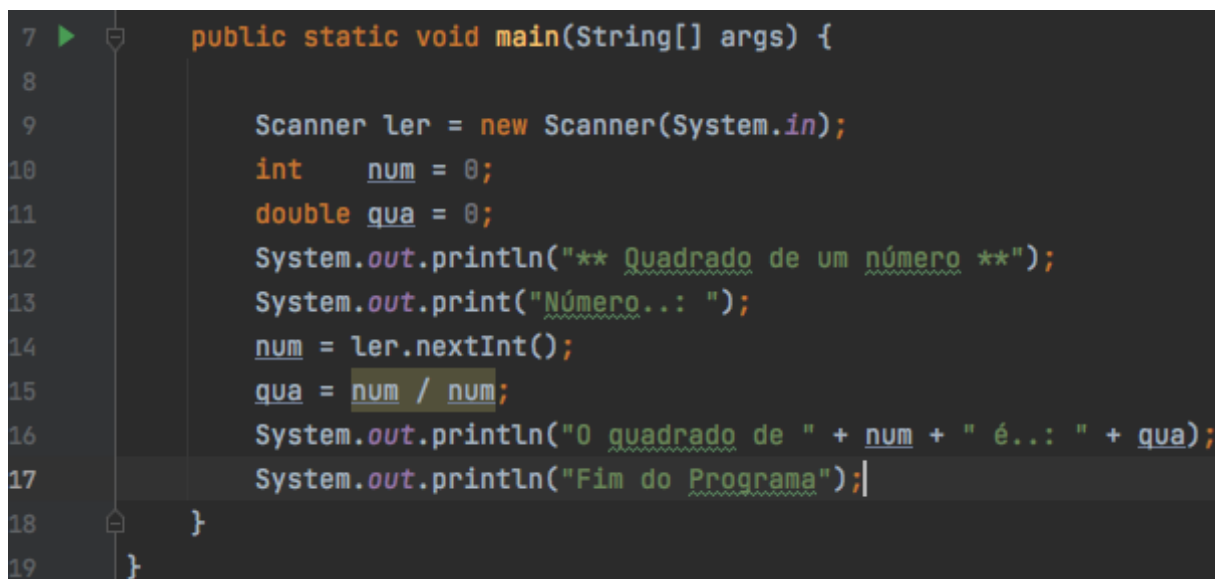


```
/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin
** Quadrado de um número **
Número..: 4
O quadrado de 4 é..: 1.0
Fim do Programa
Process finished with exit code 0
|
Run  TODO  Problems  Terminal  Build
Build completed successfully in 3 sec, 266 ms (moments ago)
```

Fonte: elaborado pelo autor via IDE IntelliJ (2021)

Vimos que o valor apresentado difere do valor 16 que realmente é o quadrado do número 4. O que aconteceu? O que deu errado? Essas questões podem ser respondidas no teste de mesa. Vamos realizá-lo e assim entender onde está o erro. Vejamos a Figura 10, que contém o código do Algoritmo em questão.

Figura 10 – Quadrado de um número

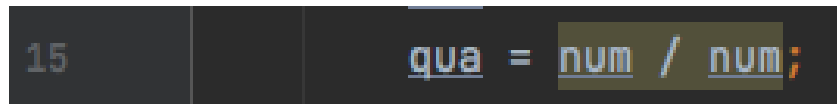


```
7 public static void main(String[] args) {\n8\n9     Scanner ler = new Scanner(System.in);\n10    int num = 0;\n11    double qua = 0;\n12    System.out.println("** Quadrado de um número **");\n13    System.out.print("Número..: ");\n14    num = ler.nextInt();\n15    qua = num / num;\n16    System.out.println("O quadrado de " + num + " é..: " + qua);\n17    System.out.println("Fim do Programa");\n18 }\n19 }
```

Fonte: elaborado pelo autor via IDE IntelliJ (2021)

Se realizarmos o teste de mesa, ou seja, a execução passo a passo do nosso algoritmo chegamos até a linha 15, no qual o operador de multiplicação foi trocado pelo de divisão, fazendo assim com que o valor final fosse alterado.

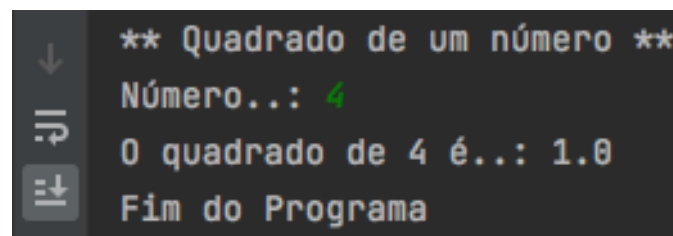
Figura 11 – Teste de mesa – parte 3



Fonte: elaborado pelo autor via IDE IntelliJ (2021)

O algoritmo encontrando essa operação fez o que estava programado para ser feito. O que estamos querendo dizer é que o algoritmo não está incorreto. O desenvolvedor é que trocou o operador, sendo assim, ele supôs que o resultado seria 16, mas na verdade ele acabou exibindo o valor 1, que é o resultado de 4 dividido por 4, figura 12.

Figura 12 – Teste de mesa – parte 4



Fonte: elaborado pelo autor via IDE IntelliJ (2021)

Videoaula 2

Agora, assista ao vídeo que aborda a demonstração do teste de mesa desse Algoritmo, a inicialização das variáveis, como iniciar o teste de mesa e fornecer algumas dicas de quando e como você deve realizá-lo. Vale a pena conferir!



Videoaula 2

Utilize o QRcode para assistir!

Agora, assista ao vídeo que aborda a demonstração do teste de mesa desse Algoritmo, a inicialização das variáveis, como iniciar o teste de mesa e fornecer algumas dicas de quando e como você deve realizá-lo. Vale a pena conferir!



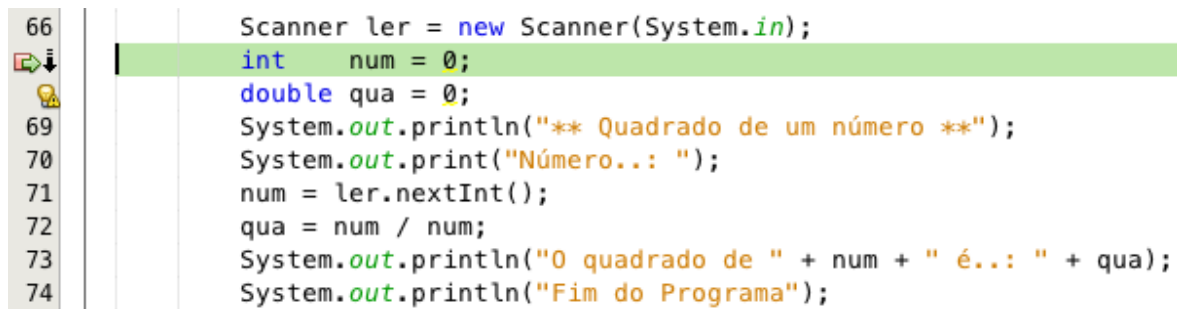
Testes de mesa no Ambiente

Todos os ambientes apresentados nessa unidade de ensino, Eclipse, NetBeans e IntelliJ, possuem processos de teste de mesa automatizado pela ferramenta. Isso significa que você pode realizar o teste de mesa dentro do próprio ambiente. Vamos apresentar esse processo por meio do NetBeans.

Partindo do mesmo exemplo do quadrado de um número vamos colocar um ***break point*** no código. Isso vai determinar onde o processo de execução irá parar para ser conduzido pelo desenvolvedor linha após linha. Basta dar um clique na barra a direita do código na linha de comando em que se deseja inserir o *break point*.

Após esse procedimento você deve iniciar a execução em modo *debugging*. Acesse o menu Depurar / Depurar Arquivo. A partir desse momento o compilador dará a condição de continuar a execução ao desenvolvedor. Observa a figura 13.

Figura 13 – Teste de mesa automatizado – parte 1

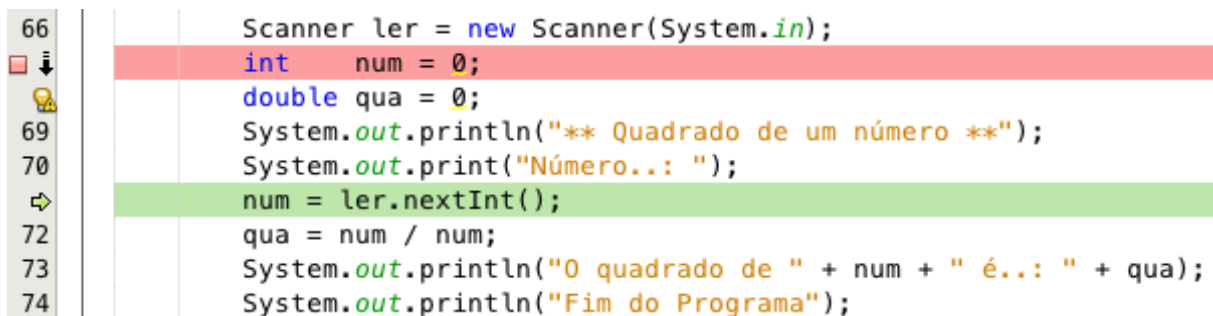


```
66 Scanner ler = new Scanner(System.in);
67 int num = 0;
68 double qua = 0;
69 System.out.println("** Quadrado de um número **");
70 System.out.print("Número..: ");
71 num = ler.nextInt();
72 qua = num / num;
73 System.out.println("O quadrado de " + num + " é..: " + qua);
74 System.out.println("Fim do Programa");
```

Fonte: elaborado pelo autor via IDE NetBeans (2021)

Veja que uma faixa na cor verde e uma seta no canto esquerdo indicam que o processo de execução foi pausado na linha 67. No NetBeans, para prosseguir a execução para a próxima linha basta ir pressionando a tecla F8. Dessa maneira a indicação da linha sendo executada vai avançando, como mostra a figura 14.

Figura 14 – Teste de mesa automatizado – parte 2

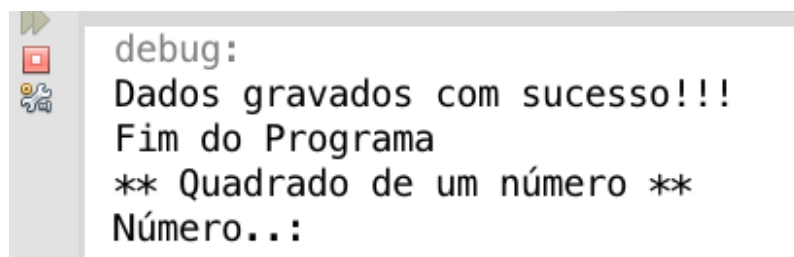


```
66 Scanner ler = new Scanner(System.in);
67 int num = 0;
68 double qua = 0;
69 System.out.println("** Quadrado de um número **");
70 System.out.print("Número..: ");
71 num = ler.nextInt();
72 qua = num / num;
73 System.out.println("O quadrado de " + num + " é..: " + qua);
74 System.out.println("Fim do Programa");
```

Fonte: elaborado pelo autor via IDE NetBeans (2021)

Da mesma maneira a execução do programa vai avançando. Veja que na figura 15 a execução está parada exatamente na mesma linha que na figura 14.

Figura 15 – Teste de mesa automatizado – parte 3



```
debug:
Dados gravados com sucesso!!!
Fim do Programa
** Quadrado de um número **
Número..:
```

Fonte: elaborado pelo autor via IDE NetBeans (2021)

Desse modo, basta ir seguindo e informando os dados de entrada, conferindo se os cálculos estão corretos até a última linha de código. Nesse exemplo de teste de mesa automatizado, replicamos o mesmo exemplo em que o operador estava trocado, sendo assim, o resultado exibido não será o desejado. É preciso ajustar o algoritmo trocando novamente o sinal de divisão pelo da multiplicação.

Para maiores detalhes, assista à videoaula!

Videoaula 3

Agora, assista ao vídeo que aborda o teste de mesa por meio dos ambientes de desenvolvimento. É possível iniciar o teste, finaliza-lo e começar novamente de acordo com a necessidade. Vale lembrar que em todos os ambientes é possível realizar esse tipo de teste. Assista o vídeo no qual o professor demonstra o teste de mesa na prática!

Bons estudos!



Videoaula 3

Utilize o QRcode para assistir!

Agora, assista ao vídeo que aborda o teste de mesa por meio dos ambientes de desenvolvimento.



Espero, que até este ponto do conteúdo, tenha ficado claro algumas noções básicas sobre os ambientes e os procedimentos de testes existentes.

Encerramento da Unidade

Chegamos ao fim desta Unidade de Ensino, após caminhar por conteúdos importantes para o seu desenvolvimento acadêmico até aqui. A Unidade 1 apresentou alguns ambientes de desenvolvimento que serão necessários para a construção de sistemas na linguagem de programação Java. Foram detalhados processos como a criação de projetos, classes e execução de exemplos no Eclipse, NetBeans e IntelliJ.

Após a assimilação dos conteúdos abordados nesta Unidade, você estará apto a prosseguir estudando um pouco mais sobre a Análise e Projeto de Algoritmos nas próximas Unidades de Ensino.

Tenha uma ótima leitura e não se esqueça de ler e estudar a Bibliografia e as aulas gravadas pelo professor.

Obrigado pela oportunidade de ser o mediador entre os conteúdos específicos abordados e o conhecimento adquirido por você por meio da didática e abordagem desta Unidade.

Referências

CORMEN, Thomas H. **Desmistificando algoritmos**. Rio de Janeiro: Campus, 2012. 926 p.

DEITEL, Harvey M.; DEITEL, Paul J. **Java: como programar**. 8. ed. São Paulo: Pearson, 2010.

FORBELLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. **Lógica de programação**. 2. ed. São Paulo: Makron Books, 2000. 195 p.

JANDL JUNIOR, Peter. **Java: guia do programador**. São Paulo: Novatec, 2007. 681 p.

PUGA, Sandra.; RISSETTI, Gerson. **Lógica de programação e estruturas de dados: com aplicações em Java**. São Paulo: Pearson, 2016.



UNIFIL.BR