

**Competências, Habilidades e Bases Tecnológicas da disciplina de Técnicas de Programação.**

<b>I.8 TÉCNICAS DE PROGRAMAÇÃO</b>	
<b>Função: Programação de baixa complexidade para computadores</b>	
<b>Classificação: Execução</b>	
<b>Atribuições e Responsabilidades</b>	
<ul style="list-style-type: none"> <li>Desenvolver programas de computador, utilizando princípios de boas práticas.</li> <li>Realizar versionamento no desenvolvimento de programas.</li> <li>Verificar usabilidade no desenvolvimento de programas.</li> </ul>	
<b>Valores e Atitudes</b>	
<ul style="list-style-type: none"> <li>Incentivar a criatividade.</li> <li>Incentivar comportamentos éticos.</li> <li>Fortalecer a persistência e o interesse na resolução de situações-problema.</li> </ul>	
<b>Competências</b>	<b>Habilidades</b>
1. Elaborar sistemas aplicando princípios e paradigmas de programação.	1.1 Codificar programas, utilizando técnica de programação estruturada. 1.2 Depurar e versionar programas, utilizando ambiente de desenvolvimento integrado.
<b>Bases Tecnológicas</b>	
<p><b>Princípios de programação</b></p> <ul style="list-style-type: none"> <li>Linguagens de programação e códigos fonte, objeto e executável;</li> <li>Paradigmas de programação;</li> <li>Conceitos de usabilidade de sistemas.</li> </ul> <p><b>Ferramentas para o desenvolvimento</b></p> <ul style="list-style-type: none"> <li>Ambiente integrado de desenvolvimento (IDE);</li> <li>Editor de código               <ul style="list-style-type: none"> <li>✓ navegação;</li> <li>✓ completar comandos;</li> <li>✓ coloração de sintaxe;</li> <li>✓ marcas de erro.</li> </ul> </li> <li>Compilação, empacotamento e distribuição (<i>build and deploy</i>);</li> <li>Bibliotecas, <i>frameworks</i> e gestão de dependências;</li> <li>Modularização e organização em projetos de programas e sistemas.</li> </ul> <p><b>Verificação e depuração de código</b></p> <ul style="list-style-type: none"> <li>Execução passo a passo;</li> <li>Criação de pontos de interrupção (<i>breakpoints</i>);</li> <li>Visualização de valores de variáveis em tempo de execução;</li> <li>Pilha de chamadas (<i>call stack</i>);</li> <li>Interpretação de informações detalhadas sobre exceções.</li> </ul> <p><b>Versionamento e colaboração</b></p> <ul style="list-style-type: none"> <li>Conceitos de controle de versão e gestão de código fonte;</li> <li>Software livre e colaboração com repositórios remotos;</li> <li>Criação de repositórios locais e remotos;</li> <li>Envio (<i>commit</i>) e resgate de versões, <i>checkin</i> e <i>checkout</i>;</li> <li>Controle de usuários para o desenvolvimento colaborativo;</li> <li>Ramificação (<i>branch</i>), comparação (<i>diff</i>) e mesclagem (<i>merge</i>).</li> </ul>	

<b>Práticas de programação</b> <ul style="list-style-type: none"><li>• Estilo de codificação, indentação, legibilidade, comentários;</li><li>• Refatoração;</li><li>• Programação em par;</li><li>• Testes unitários.</li></ul>					
<b>Carga horária (horas-aula)</b>					
Teórica	00	Prática em Laboratório*	40	Total	40 Horas-aula
Teórica (2,5)	00	Prática em Laboratório* (2,5)	50	Total (2,5)	50 Horas-aula
* Possibilidade de divisão de classes em turmas, conforme o item 4.8 do Plano de Curso.					
* Todos os componentes curriculares preveem prática, expressa nas habilidades, relacionadas às competências. Para este componente curricular está prevista divisão de classes em turmas.					
Para ter acesso às titulações dos Profissionais habilitados a ministrarem aulas neste componente curricular, consultar o site: <a href="http://www.cpscetec.com.br/crt/">http://www.cpscetec.com.br/crt/</a>					

**1-) Comunicação de alunos com alunos e professores:**

- Um e-mail para a sala é de grande valia para divulgação de material, notícias e etc.
- Criação de grupo nas redes sociais também é interessante.

**2-) Uso de celulares:**

Para o bom andamento das aulas, recomendo que utilizem os celulares em vibracall, para não atrapalhar o andamento da aula.

**3-) Material das aulas:**

A disciplina trabalha com NOTAS DE AULA que são disponibilizadas ao final de cada aula, e calendários de PTCC estarão disponíveis no site: [www.marcoscosta.eti.br](http://www.marcoscosta.eti.br), na página principal.

**4-) E-mail da disciplina:**

A disciplina terá um e-mail de envio das atividades, dúvidas e assim por diante, assim sendo todas as atividades deverão ser enviadas para tal e-mail, se forem enviadas atividades de tal disciplina para e-mail não mencionado, a atividade será desconsiderada.

O e-mail desta disciplina é: [1\\_dsn\\_tp@outlook.com](mailto:1_dsn_tp@outlook.com)

**5-) Prazos de trabalhos e atividades:**

Toda atividade solicitada terá uma data limite de entrega, de forma alguma tal data será postergada, ou seja, se não for entregue até a data limite a mesma receberá menção I, isso tanto para atividades entregues de forma impressa ou enviadas ao e-mail da disciplina. No caso de PTCC o calendário será seguido, e no dia marcado de determinada atividade só receberão menção os alunos **presentes**, caso não estejam a menção para aquela atividade será I.

**6-) Qualidade do material de atividades:**

- Impressas ou manuscritas:

Muita atenção na qualidade do que será entregue, atividades sem grampear, faltando nome e número de componentes, rasgadas, amassadas, com rebarba de folha de caderno e etc. serão desconsiderados por mim.

- Digitais:

Ao enviarem atividades para o e-mail da disciplina, SEMPRE no assunto deverá ter o nome da atividade que está sendo enviada, e no corpo do e-mail deverá ter o(s) nome(s) do(s) integrante(s) da atividade, sem estar desta forma a atividade será DESCONSIDERADA.

## 7-) Menções e critérios de avaliação:

Na ETEC os senhores serão avaliados por MENÇÃO, onde temos:

- MB - Muito Bom;
- B - Bom;
- R - Regular;
- I - Insatisfatório.

Cada trimestres teremos as seguintes formas de avaliação:

- 1 avaliação teórica;
- 1 avaliação prática (a partir do 2º trimestre, e as turmas do 2º módulo em diante);
- Seminários;
- Trabalhos teóricos e/ou práticos;
- Assiduidade;
- Outras que se fizerem necessário.

## 1. SISTEMA COMPUTACIONAL

Um sistema computacional pode ser visto como uma associação entre dois conceitos utilizados na terminologia de informática:

O hardware, que está associado à parte física do sistema (os circuitos e dispositivos) que suporta o processamento da informação;

O software, que corresponde ao conjunto de programas responsáveis pela execução das tarefas.

Software de sistema (ou sistema operacional) : facilita o acesso aos recursos do computador, através de comandos ou serviços especiais ativados a nível de programa. O sistema operacional administra os arquivos, controla os periféricos e executa utilitários;

- Software utilitário: são programas desenvolvidos para facilitar a realização de certas atividades como detecção de vírus, programas de comunicação em redes, compressão de arquivos, etc...)
- Software aplicativo: são programas desenvolvidos ou adquiridos pelos usuários para algum fim específico (de natureza profissional, educacional ou de lazer – jogos).

Sendo mais específicos, podemos definir que um programa de computador, ou software, é uma sequência de instruções que são enviadas para o computador (hardware). Cada tipo de microprocessador (cérebro) entende um conjunto de instruções diferente, ou seja, o seu próprio "idioma". Também chamamos esse idioma de linguagem de máquina. As linguagens de máquina são, no fundo, as únicas linguagens que os computadores conseguem entender, só que elas são muito difíceis para os seres humanos entenderem. É por isso nós usamos uma coisa chamada linguagem de programação.

## 2. LINGUAGEM DE PROGRAMAÇÃO

Nós seres humanos precisamos converter as nossas ideias para uma forma que os computadores consigam processar, ou seja, a linguagem de máquina. Os computadores de hoje (ainda) não conseguem entender a linguagem natural que nós usamos no dia a dia, então precisamos de um outro "idioma" especial para instruir o computador a fazer as tarefas que desejamos. Esse "idioma" é uma linguagem de programação, e na verdade existem muitas delas no mercado, como exemplos podemos citar o Arduino, Java, C#, PHP, Python etc.

### 2.1 NÍVEIS DE LINGUAGENS DE PROGRAMAÇÃO

Uma linguagem de programação é um vocabulário e um conjunto de regras gramaticais usadas para escrever programas de computador. Esses programas instruem o computador a realizar determinadas tarefas específicas. Cada linguagem possui um conjunto único de palavras-chaves (palavras que ela reconhece) e uma sintaxe (regras) específica para organizar as instruções dos programas. Os programas de computador podem ser escritos em várias linguagens de programação, algumas diretamente compreensíveis pelo computador e outras que exigem passos de tradução intermediária. As linguagens de programação podem ser divididas em três tipos, com relação à sua similaridade com a linguagem humana:

- Linguagem de máquina;
- Linguagem simbólica;

- Linguagem de alto nível.

### 2.1.1 LINGUAGEM DE MÁQUINA (MACHINE LANGUAGE)

É a linguagem de mais baixo nível de entendimento pelo ser humano e a única, na verdade, entendida pelo processador (UCP). É constituída inteiramente de números, o que torna praticamente impossível entendê-la diretamente. Cada UCP tem seu conjunto único de instruções que definem sua linguagem de máquina, estabelecido pelo fabricante do chip. Uma instrução típica em linguagem de máquina seria algo como: **0100 1111 1010**. Essa linguagem é também classificada como uma linguagem de primeira geração.

```
00110001 00000000 00000000
00110001 00000001 00000001
00110011 00000001 00000010
01010001 00001011 00000010
00100010 00000010 00001000
01000011 00000001 00000000
01000001 00000001 00000001
00010000 00000010 00000000
01100010 00000000 00000000
```

### 2.1.2 LINGUAGEM SIMBÓLICA (ASSEMBLY):

É a linguagem de nível imediatamente acima da linguagem de máquina. Ela possui a mesma estrutura e conjunto de instruções que a linguagem de máquina, porém permite que o programador utilize nomes (chamados mnemônicos) e símbolos em lugar de números. A linguagem simbólica é também única para cada tipo de UCP (Unit Central Processor), de forma que um programa escrito em linguagem simbólica para uma UCP poderá não ser executado em outra UCP de uma família diferente. Nos primórdios da programação todos os programas eram escritos nessa linguagem.

Hoje a linguagem simbólica, é utilizada quando a velocidade de execução ou o tamanho do programa executável gerado são essenciais. A conversão da linguagem simbólica para a linguagem de máquina se chama montagem, e é feito por um programa chamado montador (ou assembler). Uma típica instrução em linguagem simbólica seria: ADD A,B. Essa linguagem é também classificada como linguagem de segunda geração, e, assim como a linguagem de máquina, é considerada uma linguagem de baixo nível.

C014 24 FA	BCC	INCH	RECIEVE NOT READY
C016 B6 80 05	LDA A	ACIA+1	GET CHAR
C019 84 7F	AND A	#\$7F	MASK PARITY
C01B 7E C0 79	JMP	OUTCH	ECHO & RTS
*****			
* FUNCTION: INHEX - INPUT HEX DIGIT			
* INPUT: none			
* OUTPUT: Digit in acc A			
* CALLS: INCH			
* DESTROYS: acc A			
* Returns to monitor if not HEX input			
C01E 8D F0	INHEX	BSR	INCH
C020 81 30		CMP A	#'0
C022 2B 11		BMI	HEXERR
C024 81 39		CMP A	#'9
C026 2F 0A		BLE	HEXRTS
C028 81 41		CMP A	#'A
C02A 2B 09		BMI	HEXERR
			NOT HEX

Figura 1: Linguagem Assembly



### 2.1.3 LINGUAGEM DE ALTO NÍVEL:

São as linguagens de programação que possuem uma estrutura e palavras-chave que são mais próximas da linguagem humana. Tornando os programas mais fáceis de serem lidos e escritos. Esta é a sua principal vantagem sobre as linguagens de nível mais baixo. Os programas escritos nessas linguagens são convertidos para a linguagem de baixo nível através de um programa denominado compilador ou de um interpretador.

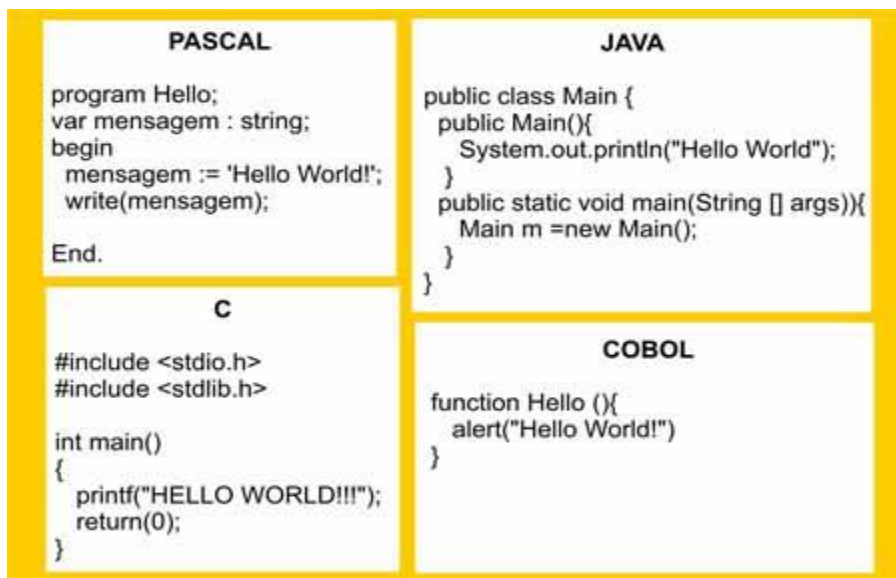


Figura 2: Exemplos de Hello World

Para relaxar ...

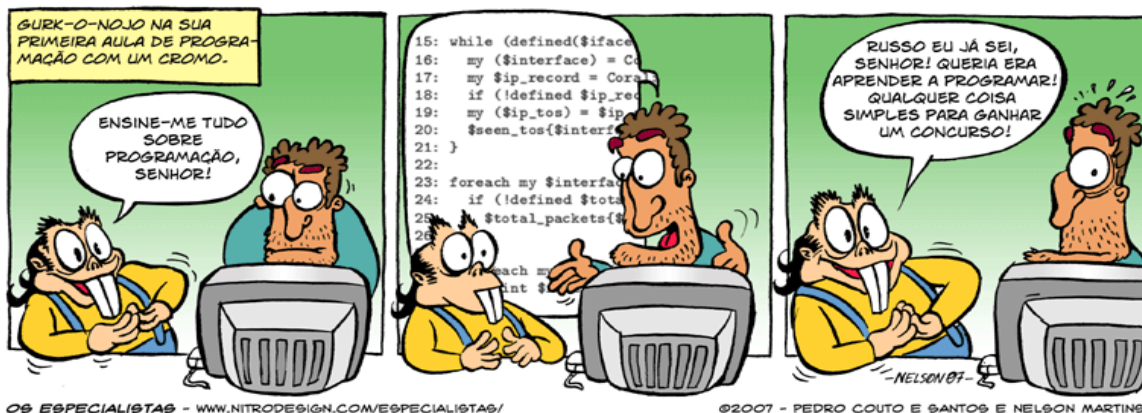


Figura 3: Para relaxar

## 2.2 MÉTODOS DE TRADUÇÃO: COMPILADORES x INTERPRETADORES

Quando estamos codificando um sistema, nem sempre nos preocupamos com o trabalho feito por trás daquela programação. No entanto, saber qual método de tradução pode ajudar bastante na hora de distribuir um projeto, especialmente quando se trata de portabilidade ou desempenho.

É nessas horas que um conhecimento prévio sobre qual técnica de implementação pode ser usada pra linguagem escolhida pode ajudar. Linguagens compiladas e interpretadas têm muito por trás do que apenas converte o código para linguagem de máquina, mas qual utilizar na minha aplicação?

### 2.2.1 QUAL DELES EU DEVO USAR?

Pode parecer descuidada ou sem preocupação com as peculiaridades específicas de um sistema, mas a resposta básica é assustadoramente simples e completa: tanto faz. Quê?

Isso mesmo, tanto faz. Na verdade, qualquer linguagem pode ser compilada ou interpretada, pois esses conceitos não são próprios de uma linguagem em si, ou seja, não é um atributo ou característica intrinsecamente ligada a ela.

Ambos são técnicas de implementação para executar (ou, melhor ainda, traduzir) o seu código fonte e podem ser aplicados para qualquer linguagem, desde que se tenha o tradutor necessário para tal.

Uma atitude comum a alguns programadores que reforça essa ideia é a de construir seu código em um editor de texto comum, tipo o bloco de notas. Depois de construído, o arquivo era usado em um programa para executado (quem nunca montou um HTML dessa forma?).

### 2.2.2 COMPILADORES x INTERPRETADORES

Basicamente, um compilador é um programa (ou um conjunto deles) como qualquer outro, porém seu objetivo principal é o de traduzir todas as suas linhas de código para outra linguagem – normalmente, uma de alto nível para outra de baixo nível (Assembly ou linguagem de máquina). Delphi, Rust, C++ e Swift figuram na lista de compiladas.

Um interpretador também é um programa, mas, ao contrário do compilador, ele não converte o código todo para linguagem de máquina de uma vez. Ele executa diretamente cada instrução, passo a passo. MATLAB, Lisp, Perl e PHP são apontadas como interpretadas.

Em suma, a grande diferença está na forma de execução. Enquanto um compilador analisa todo o código a fim de traduzi-lo de uma vez (muitas vezes, o resultado é um arquivo executável ou uma biblioteca), o interpretador faz esse trabalho de conversão aos poucos, sempre que uma declaração ou função é executada, por exemplo.

Embora isso dê a sugestão de que essa interpretação leve muito tempo para ficar pronta, o compilador também caminha bastante para ser convertido. Análise léxica e semântica, pré-processamento e análise e otimização de código são apenas alguns dos passos cumpridos até a geração do produto final. Porém, uma vez compilado, ele não precisará fazer mais essa tradução – a menos que o código precise ser alterado.

### 2.2.3 VANTAGENS DE DESVANTAGENS

Uma das grandes vantagens dos compiladores é sua velocidade de execução, muito em função do que já falamos sobre traduzir todo o código de uma vez. Não precisar fazer a conversão toda vez que o sistema é executado dá uma eficiência muito maior do que um interpretador.

Uma compilação costuma dar resultados mais confiáveis graças às suas diversas etapas de validação e otimização. Uma checagem de tipos estáticos, por exemplo, é comum em compiladores, e identifica diversos erros de programação antes do executável ser gerado. Por sua vez, enquanto uma linguagem compilada precisa fazer essa tradução para cada plataforma destinada (como versões específicas do Adobe Photoshop para Windows, Linux e Mac), a interpretação, por poder rodar em tempo de execução, é independente. Não importa se é Linux ou Mac; basta ter o Python instalado na máquina que ela vai rodar seu código em Python.

Além disso, a capacidade de execução em runtime permite utilizar reflexão (ou seja, examinar e modificar sua própria estrutura em tempo de execução) e tipagem dinâmica (capacidade de escolher dinamicamente o tipo de uma variável, não exigindo uma declaração), uma característica básica do PHP, por exemplo.

Verificar e modificar o código de uma linguagem interpretada também é mais fácil, já que basta abrir o arquivo e ver o que tem escrito. Para fazer o mesmo com uma biblioteca compilada, é preciso utilizar um descompilador.

Isso, no entanto, também pode ser visto como uma desvantagem, pois qualquer pessoa com um mínimo de conhecimento pode ver a implementação de um JavaScript embutido numa página web ou até mesmo realizar uma injeção de código. Por isso mesmo, a segurança é sempre uma preocupação em momentos assim.

### 2.2.4 SCRIPTS

Essa preocupação com a segurança e com a flexibilidade das aplicações, entre outros motivos, foram responsáveis a dar vida para aqueles pequenos trechos de código (ou scripts) baseados em Java, transformando-os em uma linguagem propriamente dita (linguagem de script) como o JavaScript.

Por outro lado, sua principal característica ainda é a interpretação do código pelo lado do cliente, através do navegador (sim, uma das funções básicas de um browser é interpretação), sem que precise ser executado no servidor.

Mesmo assim, isso não impede que um script seja compilado. Na verdade, os games fazem uso disso com certa frequência, seja para adicionar novas funcionalidades ou corrigir problemas com a aplicação de atualizações.

Lua também é uma linguagem de script extremamente poderosa, bem famosa pela construção de macros e add-ons compilados por jogadores de World of Warcraft. Porém, como ela gera bytecodes como resultado, além de ter tipagem dinâmica e gerenciamento automático de memória, é mais comum sua associação com interpretadores.

### 2.2.5 COMPILADORES + INTERPRETADORES

Por terem características diferentes (e, de certa forma, complementares), algumas linguagens fazem uso dos dois conceitos, sendo o Java o maior exemplo disso.

Implementações do Java costumam compilar o código a fim de gerar um bytecode, ao invés de instruções em linguagem de máquina. A partir dessa conversão, o bytecode pode ser interpretado por uma JVM (Java Virtual Machine), que pode ser instalada em qualquer máquina. Assim, o Java combina a confiabilidade e a otimização da compilação com a flexibilidade da interpretação.

Uma evolução desse modelo está no conceito de compilação just-in-time (JIT). O JIT nada mais é que uma compilação feita em tempo de execução, ao invés de antes de rodar a aplicação (ahead-of-time). Ela pode ser feita por arquivo, função ou até fragmentos de código, traduzindo dinamicamente essas partes e executando diretamente na memória.

Essa técnica ainda apresenta algumas desvantagens como o atraso na inicialização ("startup delay"), pois ainda é necessário carregar os primeiros blocos do código para serem compilados. Assim, quanto mais o JIT for otimizado, melhor o código gerado, mas também esse atraso fica maior.

### 3 LINGUAGEM JAVA (HISTÓRIA)

As aulas de Técnicas de Programação serão baseadas na linguagem Java, mas antes de colocarmos as "mãos na massa", uma breve introdução à mesma.

A história começa em 1991, quando um grupo de empregados da Sun Microsystems iniciaram um projeto para pequenos dispositivos eletrônicos de consumo, tais como o PDA (Personal Digital Assistant), o projeto recebeu o nome de Projeto Green, e James Gosling assumiu sua coordenação.

A ideia era possibilitar a criação de programas portáteis que pudessem ser executados em diversos dispositivos. Mais a equipe teria que desenvolver programas específicos para cada tipo de dispositivo, daí surgiu a ideia de desenvolver um sistema operacional que permitiria a utilização de seus programas pelos mais diversos tipos de equipamento. A nova linguagem foi batizada de Oak (carvalho), uma referência ao carvalho que James Gosling visualizava a partir de seu escritório. O sistema operacional que foi desenvolvido, foi chamado de GreenOS, e junto com ele foi construída uma interface gráfica padronizada.

Após ter um sistema operacional e uma interface gráfica, a equipe desenvolveu um avançado PDA chamado de Star7, a Sun Microsystems participou de uma competição pública para o desenvolvimento de uma tecnologia para TV a Cabo interativa, onde seria aplicado o Star7, no entanto ela perdeu essa competição, mesmo sendo um produto de alta qualidade o mercado ainda não estava preparado para o Star7. Perto de cortar o financiamento do projeto, a Sun decidiu abandonar a ênfase nos dispositivos eletrônicos e se voltar para a internet que já começava a crescer.

O nome da linguagem desenvolvida pelo projeto Green foi mudada de Oak para Java, que foi uma homenagem à uma ilha da Indonésia de onde os Norte-Americanos importavam o café que era consumido pela equipe de James Gosling. Até 1994, não havia uma aplicação definida para o Java. Foi quando Jonathan Payne e Patrick Naughton criaram um novo navegador para Web que podia executar programas escritos em Java (applets), batizado de Web Runner. E em 1996, em uma iniciativa inédita, a Sun Microsystems resolveu disponibilizar gratuitamente um kit de desenvolvimento de software para a comunidade, que ficou conhecido como Java Developer's Kit (JDK). Desde então a aceitação da tecnologia Java cresceu rapidamente entre empresas e desenvolvedores. A Sun Microsystems lançou o JDK 1.1 com melhorias significativas para o desenvolvimento de aplicações gráficas e distribuídas. Depois disso, a empresa continuou lançando novas versões gratuitas com novas melhorias e recursos.

Em abril de 2009, a Oracle ofereceu US\$ 7,4 bilhões pela aquisição da Sun Microsystems e a proposta foi aceita. Essa aquisição deu à Oracle a propriedade de vários produtos, incluindo o Java e o sistema operacional Solaris. Em comunicado, a Oracle afirmou

que o Java foi o software mais importante adquirido ao longo de sua história. Muitas especulações foram feitas a cerca do futuro do Java depois de passar a ser propriedade da Oracle. Mais com certeza essa aquisição contribuiu muito para que o Java tivesse um salto qualitativo.

Sua principal característica é ser multiplataforma, o que quer dizer que ela é uma aplicação que pode ser executada em qualquer plataforma que seja suportada por esta linguagem, sempre de acordo com as limitações impostas pelas plataformas que forem utilizadas.

Mesmo hoje pertencendo a Oracle, ela não perdeu a sua marca que é ser um software de código aberto e que é mantida pela comunidade de usuários e empresas Java além da própria Oracle.

### 3.1 INSTALAÇÃO DO JAVA (JDK) E NETBEANS (IDE PARA DESENVOLVIMENTO)

A primeira tarefa é fazer a instalação do mesmo, primeiramente precisamos instalar o JDK, mas o que significa JDK?

O JDK, abreviação para Java Development Kit, é um conjunto de utilitários cuja a finalidade é a permissão para criação de jogos e programas para a plataforma Java. Este pacote é disponibilizado pela Oracle, e nele vem todo o ambiente necessário para a criação e execução dos aplicativos Java.

Como bem sabemos, a linguagem de programação java foi desenvolvida pela Sun Microsystems por volta dos anos 90. Hoje, a linguagem de programação orientada a objetos mais utilizada nos mercados, pertence a Oracle.

A linguagem Java, difere mais uma vez em relação as linguagens convencionais com relação a compilação dos programas. Nas linguagens convencionais, os programas são compilados para código nativo dependendo da arquitetura e também do sistema operacional que serão utilizados, já com a linguagem Java, os programas são compilados para um bytecode, que posteriormente será executado por uma máquina virtual. Devido a isso, diversos programas como jogos online, páginas web, chats, dentre outros, são feitos utilizando a linguagem Java.

O Java JDK é composto pelo compilador e pelas bibliotecas (API's) necessárias para criação de programas em Java e ferramentas úteis para o desenvolvimento e para testes dos programas escritos por esta linguagem de programação. Além disso, uma máquina virtual Java é adicionada ao sistema operacional, no caso de ainda não ter uma instalada no computador. O JDK dispõe de um arquivo executável que faz todo o trabalho de instalação e configuração do ambiente, o que facilita ainda mais a execução de qualquer aplicação Java e criação de novos programas de forma mais simplificada e sem muitos esforços.

Após essa breve explicação com relação ao JDK, façamos o download do pacote pelo endereço <http://www.oracle.com/technetwork/pt/java/javase/downloads/index.html>. Ao acessar o site, veremos 2 opções de download. Na primeira opção, baixamos apenas o pacote JDK (hoje, na versão 8) e fazemos a instalação de acordo com as instruções dele. E na segunda opção, ainda temos a vantagem de instalar o netbeans, que é a melhor IDE para desenvolvimento java, além disso, ele já faz as devidas configurações para um melhor funcionamento do conjunto IDE-JDK.



Figura 4: Download do JDK



Após o download do pacote (dependendo da sua opção), é necessário aceitar o acordo de licença e ser um usuário cadastrado no site para então poder fazer o download do arquivo. Após o cadastro e devidas permissões, vocês escolham a plataforma que será utilizada e façam o download do arquivo.

### 3.1.1 INSTALAÇÃO CONJUNTA COM O NETBEANS

Interessante é realizar a instalação da JDK juntamente com a IDE netbeans para que desta forma vocês possam ver a facilidade que a ferramenta nos oferece. O netbeans que estaremos baixando aqui está na sua versão mais recente que é a versão 8.2. Utilizaremos também o sistema operacional Windows 7 32 bits para a realização da nossa instalação. Após a finalização do download e dos arquivos terem sido baixados, agora iremos executar o arquivo de instalação. Mão a obra!

Ao executar o arquivo de instalação que baixamos, é iniciada uma configuração realizada pelo netbeans que tem por finalidade primária verificar com relação aos possíveis itens que já possam estar instalados na nossa máquina e já não serem apresentados para instalação. No nosso caso, não temos nada ainda, então a nossa instalação será completa.

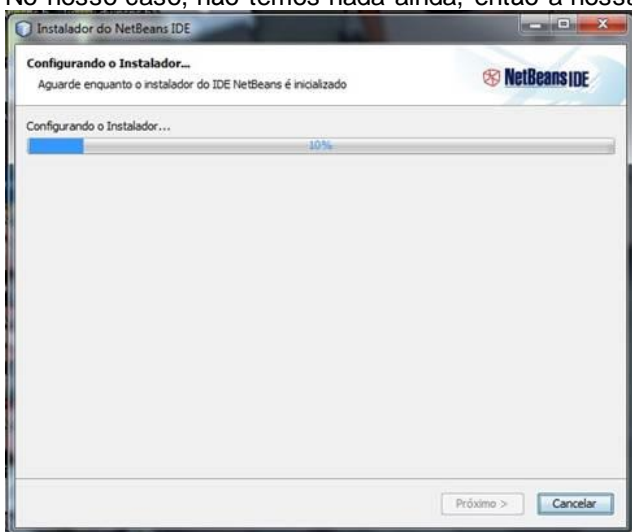


Figura 5: Configuração do instalador do netbeans.

A próxima tela apresentada informa com relação aos itens que serão instalados por padrão pelo netbeans, caso não queiramos ou não tenhamos a necessidade de instalar todas as opções disponibilizadas, clicamos em “Personalizar” e daí então selecionamos os itens que nos interessam. Como este é um exemplo de instalação onde queremos todos os itens, permaneceremos com as opções padrões passadas.

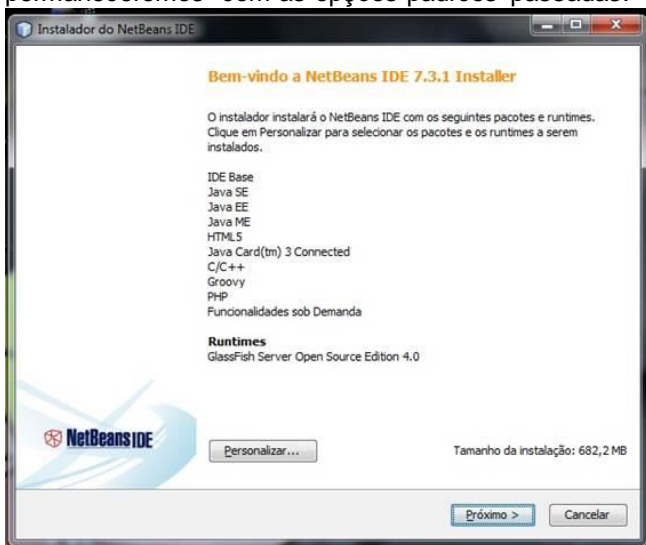


Figura 6: Pacotes que serão instalados por padrão.

Após clicar em “Próximo”, serão apresentados os termos de acordo de uso do netbeans e mais outro termo questionando se queremos instalá-lo já com o [JUnit](#) (Java unit - que é utilizado para realização de testes unitários). Aceitaremos então os dois termos e seguimos adiante com nossas configurações.

Na próxima tela, são apresentadas as informações referentes ao caminho que será utilizado para a instalação do netbeans (deixaremos esta informação da forma que nos foi passada) e também é apresentado o caminho onde se encontrará a nossa jdk.

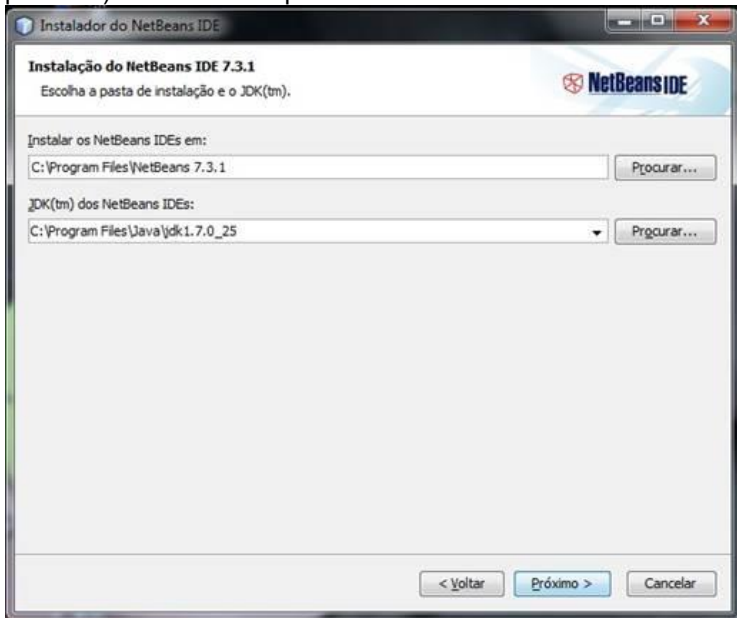


Figura 7: Localização do jdk e do netbeans.

Ao clicarmos em “próximo” neste passo, será apresentada uma nova tela onde por fim, clicaremos em “instalar” e então teremos dado início a instalação da nossa IDE com a nossa JDK atualizada e devidamente configurada para nosso uso. Assim que a instalação for concluída, é só começarmos a descobrir o mundo que o netbeans nos oferece com todas as nossas necessidades básicas pré-configuradas.

#### 4 APLICAÇÃO DE TÉCNICAS DE PROGRAMAÇÃO (OPERADORES, ESTRUTURA DE DECISÃO)

Nesse primeiro momento iremos criar um programa que receberá o nome de um determinado aluno, juntamente com 4 notas, o nosso objetivo é calcular uma média aritmética simples e verificar se o mesmo foi aprovado ou reprovado se a mesma for superior ou igual a 6.

Para isso abriremos a IDE Netbeans, e iremos em Arquivo->Novo Projeto, conforme a seguir:

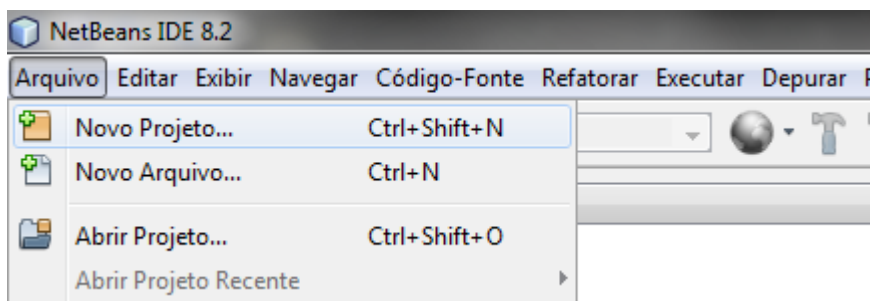


Figura 8: Iniciando um novo projeto.

Em seguida definimos que iremos trabalhar com uma aplicação Java, conforme a seguir:

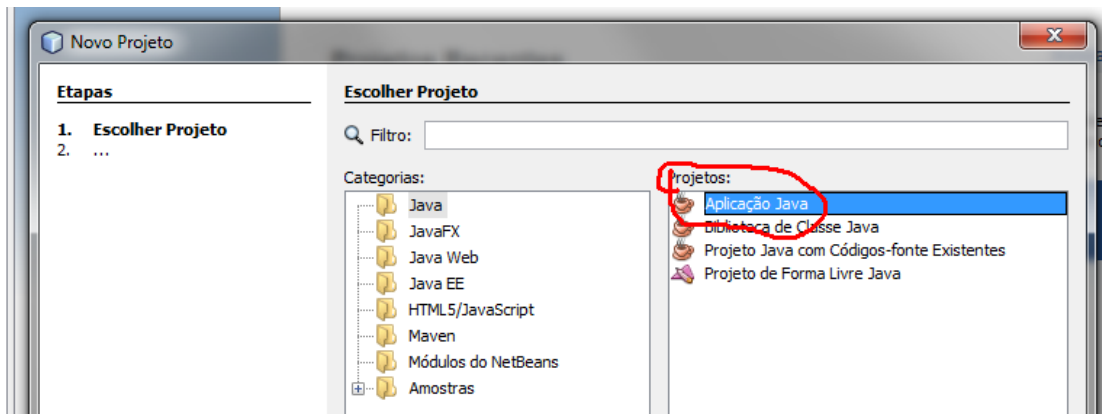


Figura 9: Iniciando um novo projeto.

Em seguida definimos o nome do Projeto, que será AulaTP, conforme a seguir:

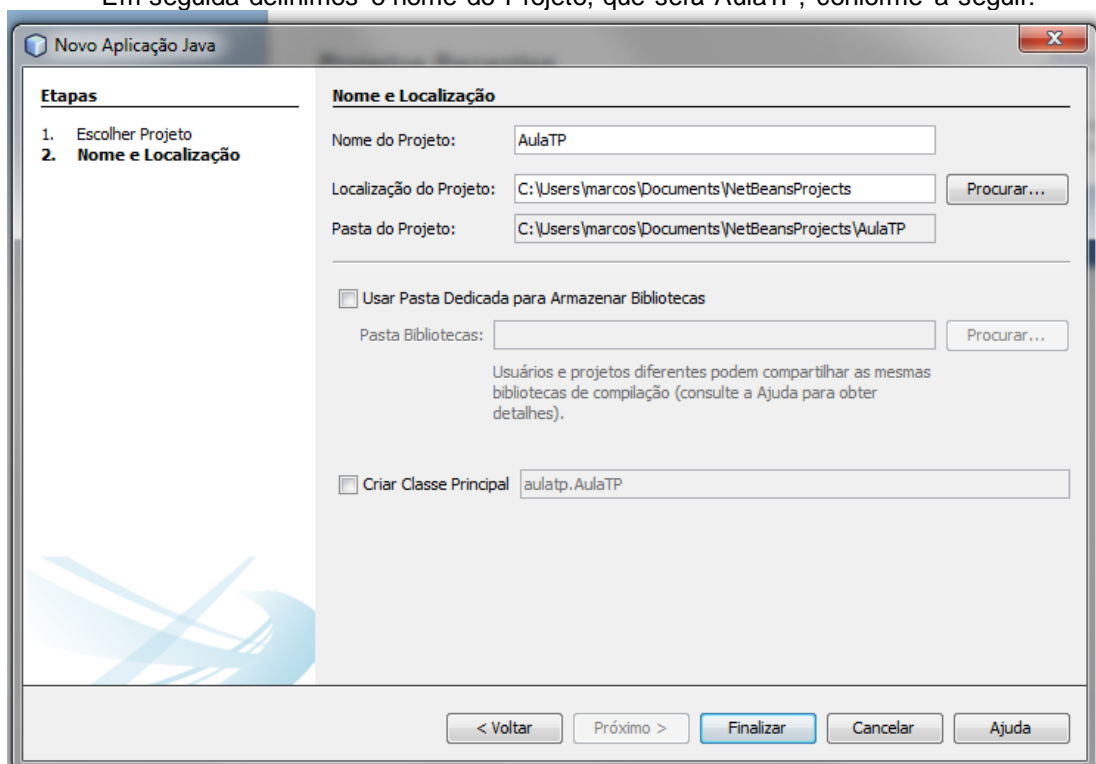


Figura 10: Definindo o nome do projeto.

Clicando em finalizar, o projeto será aberto na IDE Netbeans, conforme abaixo:

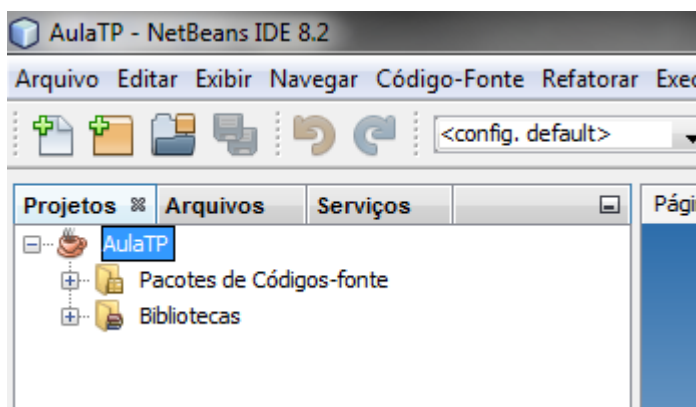


Figura 11: Projeto criado e pronto para início

Agora vamos iniciar o nosso projeto, clicando com o botão direito no projeto, iremos em Novo->Form JFrame

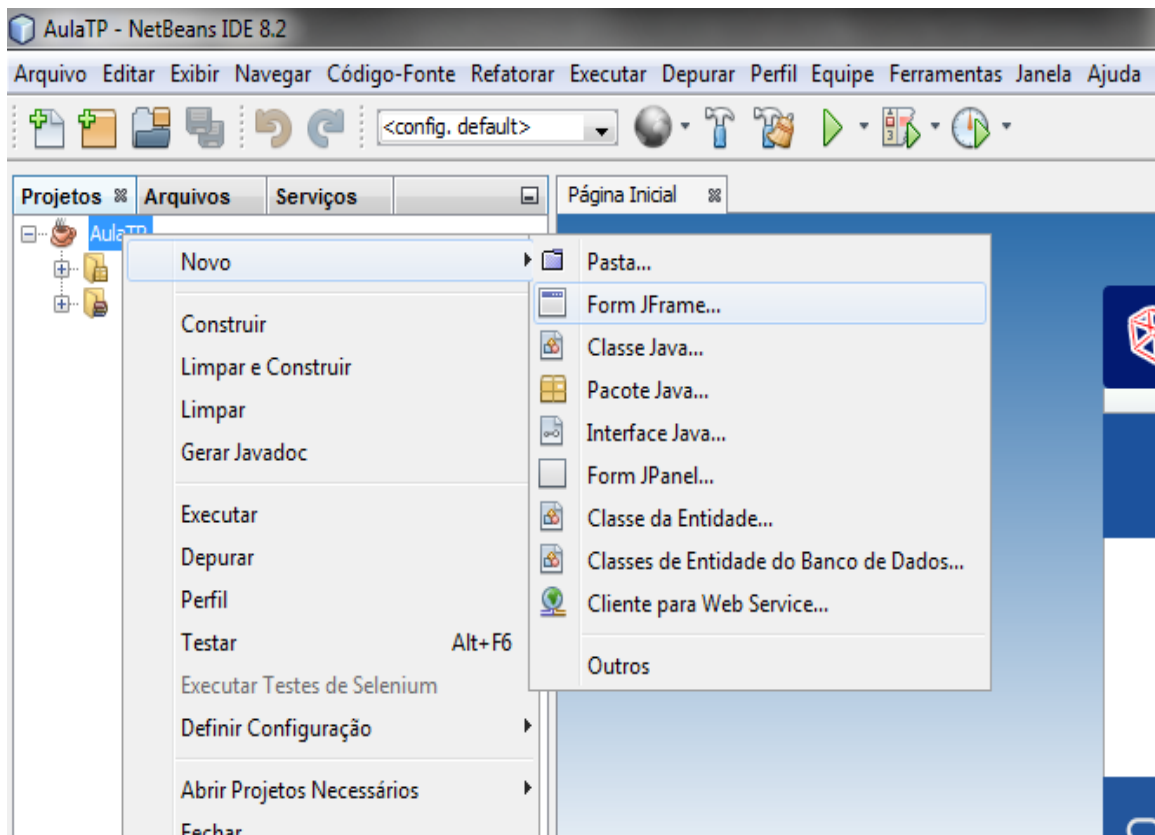


Figura 12: Criando um Formulário

Definimos o nome da classe, ou por enquanto, nome do Formulário, MediaAluno e clicamos em finalizar, conforme abaixo:

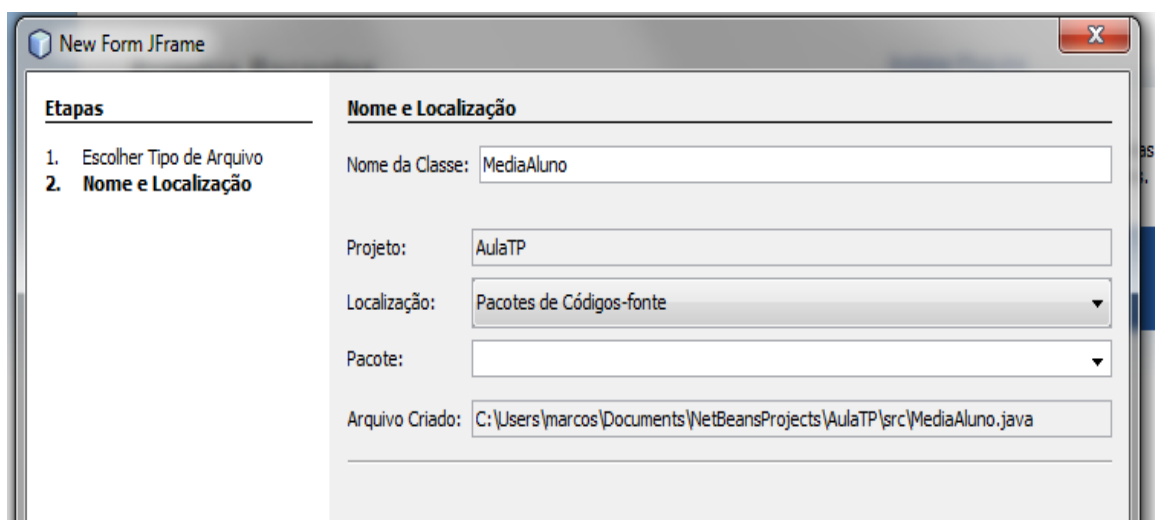


Figura 13: Nomeando a classe



Após isso, o Netbeans apresenta o projeto da seguinte forma:

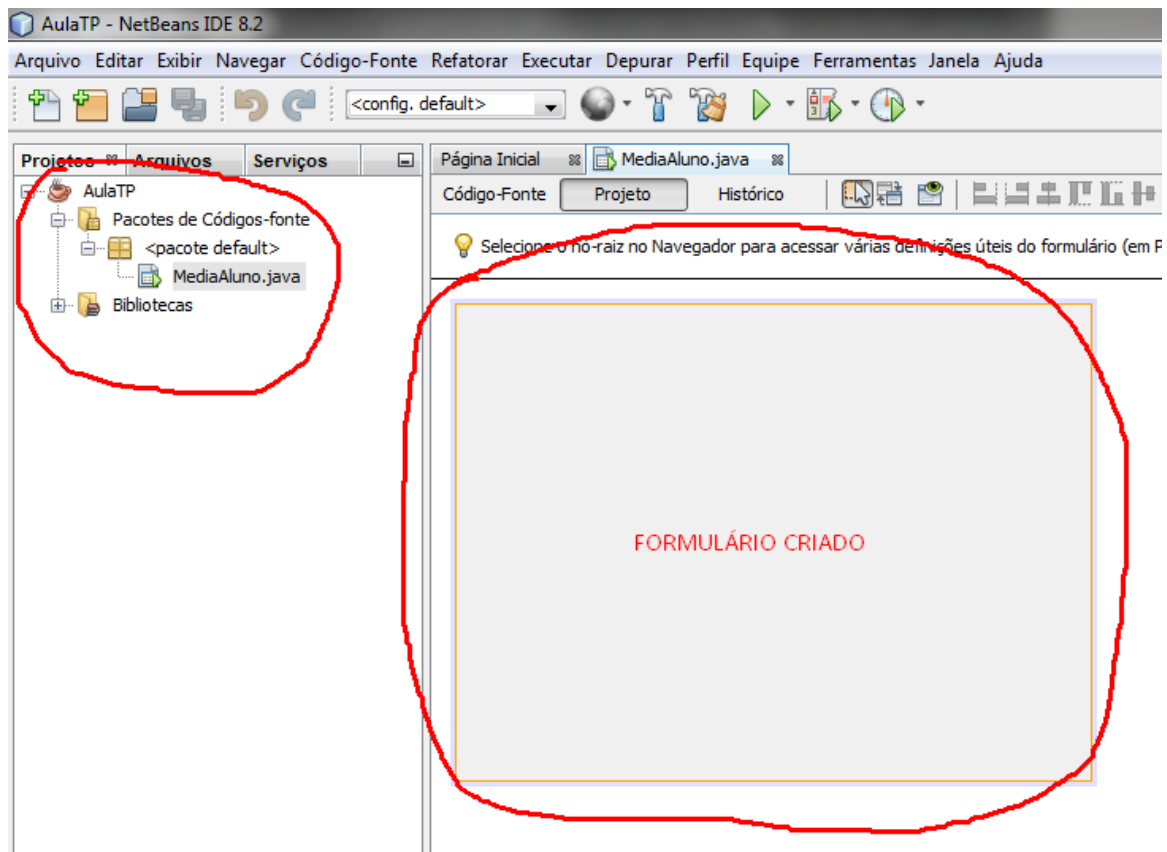


Figura 14: Projeto criado juntamente com Formulário.

A ideia inicial é trabalharmos conceitos e técnicas de programação como:

- Atribuição dos dados informados pelo usuário a variáveis;
- Cálculos;
- Concatenação de Strings e valores numéricos;
- Exibição de mensagem ao usuário final;

Para exemplificar os conceitos, iremos trabalhar com uma validação e cálculo de média de um determinado aluno, a ideia da tela é ficar assim:

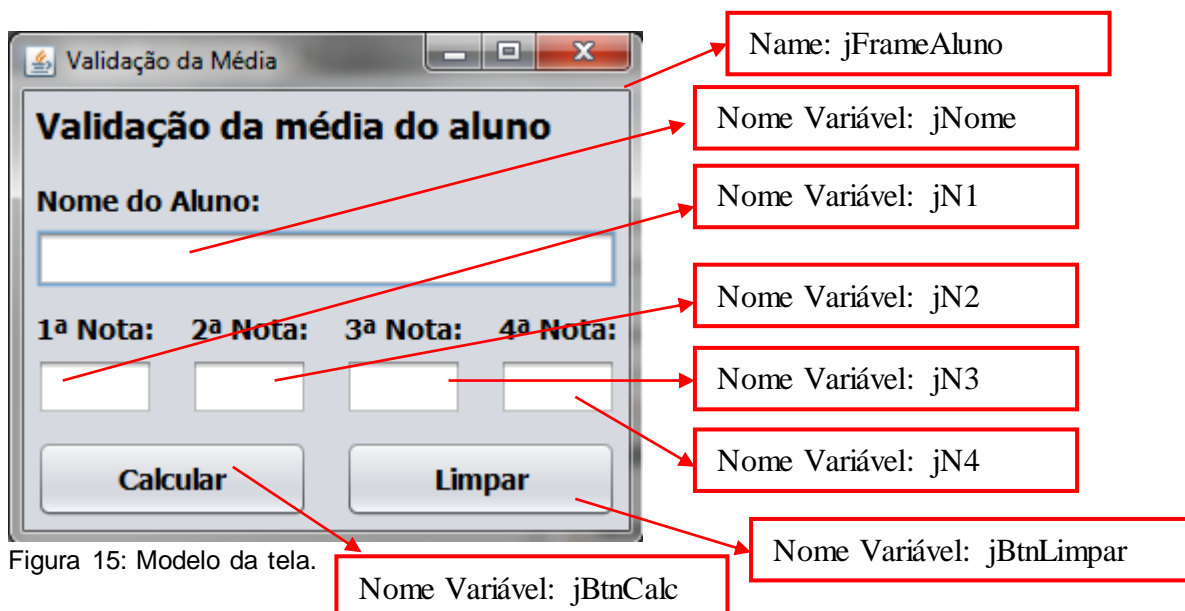


Figura 15: Modelo da tela.

Agora temos o código fonte do nosso programa.

```

1 //Importando a biblioteca de Caixa de Mensagens
2 import javax.swing.JOptionPane;
3
4 public class MediaAluno extends javax.swing.JFrame {
5
6     /** Creates new form MediaAluno ...3 linhas */
9     public MediaAluno() {...3 linhas }
12
13     /** This method is called from within the constructor to initialize the form ...5 linhas */
18     @SuppressWarnings("unchecked")
19     Generated Code
149
150     private void jBtnCalcActionPerformed(java.awt.event.ActionEvent evt) {
151         // Declarando as variáveis
152         String vNome, vMensa;
153         double vN1, vN2, vN3, vN4, vMedia;
154
155         //Atribuindo os valores dos objetos as variáveis
156         vNome = jNome.getText();
157         vN1 = Double.parseDouble(jN1.getText());
158         vN2 = Double.parseDouble(jN2.getText());
159         vN3 = Double.parseDouble(jN3.getText());
160         vN4 = Double.parseDouble(jN4.getText());
161
162         //Fazendo o cálculo da média
163         vMedia = (vN1 + vN2 + vN3 + vN4) / 4;
164         vMensa = "A Média para o Aluno " + vNome + " é " + vMedia;
165
166         //Informar ao usuário a média calculada
167         JOptionPane.showMessageDialog(null, vMensa, "Informações", JOptionPane.INFORMATION_MESSAGE);
168
169     }
170
171     private void jBtnLimparActionPerformed(java.awt.event.ActionEvent evt) {
172         //Limpando o conteúdo dos jTexts utilizados
173         jN1.setText("");
174         jN2.setText("");
175         jN3.setText("");
176         jN4.setText("");
177         jNome.setText("");
178
179         //Colocando o Foco do cursor em Nome
180         jNome.requestFocus();
181
182     }

```

Conforme vemos, temos três trechos de código para analisarmos, seguem:

### 1 – Importação da biblioteca JOptionPane:

JOptionPane é uma classe que possibilita a criação de uma caixa de dialogo padrão que ou solicita um valor para o usuário ou retorna uma informação. Abaixo encontra-se alguns métodos e parâmetros mais utilizados quando se opta pelo JOptionPane.

### Métodos

Método	Descrição
showConfirmDialog	Solicita uma confirmação como(YES, NO, CANCEL)
showInputDialog	Solicita algum valor
showMessageDialog	Informa ao usuário sobre algo
showOptionDialog	Unificação dos tres acima

## Parâmetros

Parâmetros	Descrição
parentComponent	Define a caixa de diálogo onde irá aparecer todo o conteúdo. Há duas maneiras de definir a caixa de diálogo a primeira você mesmo cria utilizando os conceitos da classe JFrame. A segunda, você define esse parametro como null e o java irá gerar uma caixa de diálogo padrão.
message	É a mensagem que o usuário deve ler. Esta mensagem pode ser uma simples String ou um conjunto de objetos.
messageType	Define o estilo da mensagem. Ele pode expor a caixa de dialogo de formas diferentes, dependendo deste valor, pode fornecer um ícone padrão. Exemplos: ERROR_MESSAGE INFORMATION_MESSAGE WARNING_MESSAGE QUESTION_MESSAGE PLAIN_MESSAGE
optionType	Define o conjunto de botões que irá aparecer na parte inferior da caixa de diálogo. Exemplos: DEFAULT_OPTION YES_NO_OPTION YES_NO_CANCEL_OPTION OK_CANCEL_OPTION

## Exemplos:

1. Mostra um diálogo de erro que exibe a mensagem “alerta”:

```

1
2 JOptionPane.showMessageDialog(null, “alerta”,
3     “alerta”, JOptionPane.ERROR_MESSAGE);
4
5

```



2. Mostra um painel de informação com as opções Sim/Não e exibe a mensagem: ‘Escolha um.’

```

1
2 JOptionPane.showConfirmDialog(null, “Escolha um:”,
3     “Escolha um”, JOptionPane.YES_NO_OPTION);
4
5

```



3. Mostrar uma janela de aviso com as opções OK, CANCELAR, o texto 'Aviso' no título e a mensagem 'Clique em OK para continuar':

```
1
2     Object[] options = { "OK", "CANCELAR" };
3     JOptionPane.showOptionDialog(null, "Clique OK para continuar", "Aviso",
4         JOptionPane.DEFAULT_OPTION, JOptionPane.WARNING_MESSAGE,
5         null, options, options[0]);
6
```



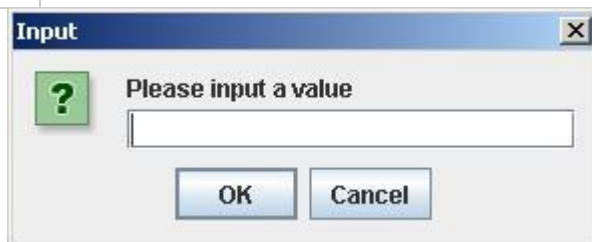
Note que se eu adicionar mais um item no vetor options, automaticamente adicionar mais um botão de opção com o nome que eu colocar.

```
1
2     Object[] options = { "OK", "CANCELAR", "VOLTAR" };
3     JOptionPane.showOptionDialog(null, "Clique OK para continuar", "Aviso",
4         JOptionPane.DEFAULT_OPTION, JOptionPane.WARNING_MESSAGE,
5         null, options, options[0]);
6
```



4. Mostra uma caixa de diálogo solicitando que o usuário digite uma string:

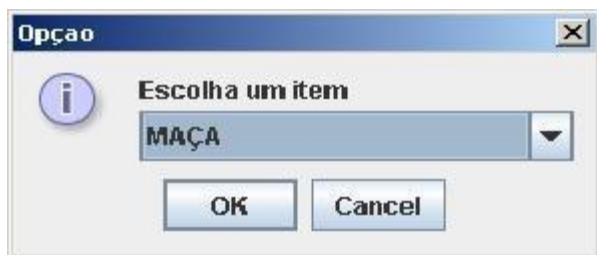
```
1
2     String inputValue = JOptionPane.showInputDialog("Please input a value");
3
4
```



5. Mostra uma caixa de diálogo solicitando que o usuário selecione uma item:

```
1
2     Object[] itens = { "MAÇA", "PERA", "BANANA" };
3     Object selectedValue = JOptionPane.showInputDialog(null,
4         "Escolha um item", "Opção",
5         JOptionPane.INFORMATION_MESSAGE, null,
6         itens, itens [0]); //
7
8
```





## 2 – Programação do botão de cálculo:

Temos as declarações de variáveis, atribuições das mesmas com os objetos criados no JFrame, e de acordo com o tipo de cada variável criada, utilizando o Parse, depois realizamos o cálculo da média utilizando essas variáveis atribuindo a uma média. Em seguida criamos uma variável de mensagem contendo uma frase concatenando o nome do aluno, o texto e a média calculada.

## 3 – Programação do botão limpar:

Fazemos a limpeza dos objetos utilizados e colocamos o cursor no jNome para que o usuário possa iniciar um novo processo.

Codifique esse projeto de acordo com a explicação passada.

## 4.1 MELHORANDO NOSSO PROJETO DE ACORDO COM TÉCNICAS DE PROGRAMAÇÃO E NOVAS EXIGÊNCIAS DO USUÁRIO

Sabemos que um projeto pode sofrer alguma alteração a pedido do usuário, ou até mesmo, nós, como desenvolvedores, podemos melhorar ou alterar esse programa de acordo com uma necessidade que julgemos necessária.

Nesse primeiro momento, vamos dizer que o usuário do nosso sistema, adorou a implementação, mas quer acrescentar o seguinte processo a mesma:

*“Gostaria que após o cálculo da média, o sistema me informasse se esse determinado aluno foi aprovado ou reprovado, teria possibilidade?”*

O que nos remete nesse momento, é entender o problema e enxergar o que esse usuário entende em “Aprovado” ou “Reprovado”, ou seja, ele precisa nos informar uma faixa de média para que possamos dizer ao nosso programa como será realizada essa verificação. Aí perguntamos:

*“Sim, existe a possibilidade, mas você precisa me informar, qual faixa de valores nessa média nosso sistema deverá considerar para isso.”*

O usuário entende a pergunta e diz:

*“Vamos considerar que os alunos que obtiverem uma média igual ou superior a 5, estarão aprovados, caso contrário estarão reprovados. Pode ser assim?”*


Nós temos a informação necessária para incrementarmos nosso projeto de acordo com as exigências de nosso usuário e dizemos:

*“Perfeito! Com isso agora, conseguimos efetuar a alteração que pediste! Assim que terminarmos, voltamos a conversar!”*

Para nós agora, refletir sobre nossa conversa com o usuário, e entendermos como alterar esse projeto com o menor impacto possível na estrutura já criada é fundamental, para investirmos um tempo mínimo na adequação do mesmo, por isso que sempre é importante investir um tempo maior pensando em que fazer, para no futuro não termos o retrabalho, pois isso ocorrendo, perderemos um grande tempo com essa nova adequação.

Para essa solução, vamos utilizar a estrutura de decisão if (Se), e de acordo com o cálculo da média diremos ao usuário a situação de aprovação ou reprovação.

Como foi visto na disciplina de Programação e Algoritmos, a estrutura de decisão SE em nosso caso do Java ... if, teremos uma condicional testada através de operadores relacionais e o resultado dessa relação será um valor booleano, no código a seguir veremos bem isso! Mas vale salientar que no Java, na segunda estrutura if (observação 3 e que está comentada) que utilizamos, ao invés do operador relacional == (igual) podemos utilizar a função equals() do Java, as duas fazem a mesma ação, ou seja, comparam valores, a função equals() é recomendada quando utilizamos comparação com objetos (Iremos ver isso durante nosso curso), mas com ==

também funciona, só estou fazendo um momento cultural, e vocês perceberão que aparecerá no número da linha uma advertência .

Abaixo temos o código alterado para essa nova implementação sugerida por nosso usuário.

```

152 private void jBtnCalcActionPerformed(java.awt.event.ActionEvent evt) {
153     // Declarando as variáveis
154     String vNome, vMensa, vStatus;
155     double vN1, vN2, vN3, vN4, vMedia;
156
157     //Atribuindo os valores dos objetos as variáveis
158     vNome = jNome.getText();
159     vN1 = Double.parseDouble(jN1.getText());
160     vN2 = Double.parseDouble(jN2.getText());
161     vN3 = Double.parseDouble(jN3.getText());
162     vN4 = Double.parseDouble(jN4.getText());
163
164     //Fazendo o cálculo da média
165     vMedia = (vN1 + vN2 + vN3 + vN4) / 4;
166
167     //Verificando a média do aluno
168     if (vMedia >= 5){
169         vStatus = "Aprovado";
170     }else{
171         vStatus = "Reprovado";
172     }
173
174     //De acordo com a comparação acima montamos a mensagem
175     vMensa = "A Média para o Aluno " + vNome + " é " + vMedia +
176             " portanto ele está " + vStatus + "!";
177
178     //Informar ao usuário a média calculada e mostrar a msg de acordo com o status, podemos usar
179     //a função equals() ao invés de ==, por se tratar de uma comparação com String.
180
181     //Forma operador ternário
182     JOptionPane.showMessageDialog(null, vMensa, (vStatus == "Aprovado")? " : ( Informações",
183             (vStatus == "Aprovado")? JOptionPane.INFORMATION_MESSAGE : JOptionPane.INFORMATION_MESSAGE);
184
185     /* Forma operador if/else
186     if(vStatus == "Aprovado"){
187         JOptionPane.showMessageDialog(null, vMensa, " : ( Informações", JOptionPane.INFORMATION_MESSAGE);
188     }else{
189         JOptionPane.showMessageDialog(null, vMensa, " : ( Informações", JOptionPane.ERROR_MESSAGE);
190     }*/
191 }

```

Conforme vemos, temos mais três trechos de código para analisarmos que acrescentamos algo para satisfazer a necessidade de nosso usuário, segue:

### 1 – Criação de mais uma variável ... vStatus:

Criamos essa variável para armazenar o status do alunos, se o mesmo será aprovado ou reprovado para uma posterior utilização.

### 2 – Condicional if para verificação da média:

De acordo com a parametrização de nosso usuário, montamos a condicional if para testar e atribuir o status de acordo com essa verificação, para isso utilizamos a condicional if, onde faremos abaixo uma explicação mais aprofundada sobre essa condicional, que podem ter algumas variações, como abaixo:

#### If/else

A estrutura condicional if/else permite ao programa avaliar uma expressão como sendo verdadeira ou falsa e, de acordo com o resultado dessa verificação, executar uma ou outra rotina.

Na linguagem Java o tipo resultante dessa expressão deve ser sempre um boolean, pois diferentemente das demais, o Java não converte null ou inteiros como 0 e 1 para os valores true ou false.

Sintaxe do if/else:

```
if (expressão booleana) {  
    // bloco de código 1  
} else {  
    // bloco de código 2  
}
```

As instruções presentes no bloco de código 1 serão executadas caso a expressão booleana seja verdadeira. Do contrário, serão executadas as instruções presentes no bloco de código 2.

O Java utiliza as chaves como delimitadores de bloco e elas têm a função de agrupar um conjunto de instruções. Apesar do uso desses delimitadores ser opcional caso haja apenas uma linha de código, ele é recomendado, pois facilita a leitura e manutenção do código, tornando-o mais legível.

**else if**

Complementar ao if/else temos o operador else if. Esse recurso possibilita adicionar uma nova condição à estrutura de decisão para atender a lógica sendo implementada.

Sintaxe do if/else com else if:

```
if (expressão booleana 1) {  
    // bloco de código 1  
} else if (expressão booleana 2) {  
    // bloco de código 2  
} else {  
    // bloco de código 3  
}
```

Dessa forma, se a expressão booleana 1 for verdadeira, o bloco de código 1 será executado. Caso seja falsa, o bloco de código 1 será ignorado e será testada a expressão booleana 2. Se ela for verdadeira, o bloco de código 2 será executado. Caso contrário, o programa vai ignorar esse bloco de código e executar o bloco 3, declarado dentro do else.

Podemos utilizar quantos else if forem necessários. Entretanto, o else deve ser adicionado apenas uma vez, como alternativa ao caso de todos os testes terem falhado.

**Operador ternário**

O operador ternário é um recurso para tomada de decisões com objetivo similar ao do if/else, mas que é codificado em apenas uma linha.

Sintaxe do operador ternário:

```
(expressão booleana) ? código 1 : código 2;
```

Ao avaliar a expressão booleana, caso ela seja verdadeira, o código 1, declarado após o ponto de interrogação (?) será executado; do contrário, o programa irá executar o código 2, declarado após os dois pontos (:).

**Exemplo prático**

Para fixar esse monte de coisa que expliquei, nada mais que um bom exemplo prático para fixarmos, suponha que você está desenvolvendo um software para controle de estoque que precisa informar como está a quantidade de itens de cada produto: se suficiente, para

quantidades superiores a 100; em alerta, para quantidades entre 100 e 50; e abaixo do ideal, para quantidades menores do que 50. Como programar esse código?

Exemplo de uso de if/else:

```
int estoque = //valor recuperado do sistema

if (estoque >= 100) {
    System.out.println("Produto com quantidade suficiente.");
} else if (estoque < 100 && estoque > 50) {
    System.out.println("Alerta: Avaliar possibilidade de novo pedido.");
} else {
    System.out.println("ATENÇÃO! Faça um novo pedido.");
}
```

Agora, considere que você precisa de uma estrutura de decisão mais simples, apenas para indicar se estamos na primeira ou na segunda quinzena de um mês.

Exemplo de uso do operador ternário:

```
int numeroDias = //valor entre 1 e 30
System.out.println((numeroDias < 15) ? "Primeira quinzena" : "Segunda quinzena");
```

### 3 – Condicional ternária ou if/else para qualificar a mensagem a ser mostrada ao usuário:

Ao final de nosso processo “printamos” através de uma mensagem ao usuário, o resultado da média calculada, mas também o status apurado no if/else anterior, para melhorarmos, usamos a condicional ternário dentro da função JOptionPane para estilizarmos essa mensagem de acordo com o status verificado. Fiz de duas formas, uma com operador ternário, onde somente em uma linha de código resolvo o problema ou, com if/else tradicional (que está comentado para não executar), testem as duas formas.

#### 4.1.1 EXERCÍCIO PRÁTICO PARA ENTREGA

Utilizando o projeto anterior como base, nosso usuário nos chama novamente perguntando se poderíamos fazer mais um ajuste no sistema. Como somos bons desenvolvedores, nos dispomos a trabalhar nesse novo desafio, agora o usuário pede que tenhamos a seguinte situação na média do aluno:

Caso esse aluno tenha uma média superior a 7, ele estará aprovado, mas caso a média fique abaixo de 5, estará reprovado, e caso fique entre 5 e 7, o mesmo estará de recuperação, coloquem essa situação no sistema.

Essa atividade poderá ser feita em duplas, ao final chamem os professores para conferência e lançamento da menção, o prazo para realização dessa atividade será a aula de hoje.