UNIVERSIDADE FEDERAL DO PARA

CENTRO TECNOLOGICO

CURSO DE ENGENHARIA DA COMPUTACAO

"PROGRAMACAO ORIENTADA A OBJETOS: UMA ANÁLISE COM JAVA."

Joarley Wanzeler de Moraes.

Manoel Onório Netto.

"A programacao orientada a objetos basea-se em coisas simples que o ser humano aprende desde a infancia como objetos e atributos, classes e membros, todo e parte do todo."

Belém, 19 de junho de 2007.

Indice.

1- RESUMO;

O presente trabalho tem com um dos objetivos fazer uma análise sobre um dos mais novos e difundidos paradigmas de programação de computadores: **a Programação Orientada a Objetos(POO).** Paralelamente à apresentação dos conceitos básicos em POO, vamos mostrar algumas aplicações dos mesmos na linguagem **JAVA** de programação(que é uma linguagem de programação orientada a objetos). É importante dizer que uso de exemplos da sintaxe java não é uma tentativa ensinar plenamente tal sintaxe, pois isso foge do foco deste trabalho. Os exemplos são apenas ilustrativos. Por isso não seram detalhados.

Vamos enfatizar também os principais benefícios do uso da POO na produção de software, mostrando que tais benefícios foram conseguidos quando superou-se antigos problemas das tecnicas tradicionais de programação.

Em seguida, trataremos especificamente de Java, mostrando aspectos peculiares os quais a tornam uma das mais poderosas linguagem de programação da atualidade. Alem disso mostraremos o que pode ser feito com java e algumas de suas aplicaçoes interessantes.

2- PROGRAMAÇÃO ORIENTADA A OBETOS;
2.1- HISTÓRICO DA PROGRAMACÃO ORIENTADA A OBJETOS;

Por muitos anos as técnicas de programação mantiveram-se inalteradas independentes da linguagem utilizada baseando-se na estrutura, sempre consistente na visão tradicional de computadores.
Nos últimos trinta anos surgiu uma abordagem diferente de programação, onde os programas são montados com peças denominadas objetos, permitindo ao programador dividir o programa em componentes, incrementando o desenvolvimento em eficiência e rápida manutenção.
A ***POO*** (**Programação Orientada a Objetos)** ou ***OOP*** **(*Oriented Object Programing*)** não concentra-se em uma linguagem de programação, mas na maneira de programar.
A Programação Orientada a Objetos surgiu em 1967 quando dois cientistas dinamarqueses criaram a Linguagem ***Simula*** (*Simulation Language*) que compreendia conceitos de ***classe*** *e* ***herança.***

A primeira linguagem a suportar os métodos de programação orientado a objeto foi a ***Smalltalk (***parcialmente baseada *em Simula)* criada na década de 1970 no Centro de Pesquisas da Xerox em Palo Alto. *Smalltalk* viria a ser uma das mais populares linguagens de programacao até o aparecimento de C++.

A **linguagem de Programação Orientada a Objetos (LPOO)** levou algum tempo para decolar e somente na metade dos anos 1980 é que começaram a surgir recursos orientados a objetos permitindo uma abordagem poderosa e prática para o desenvolvimento de software.

Entre 1983 e 1985, Bjarne Stroustrup criou a linguagem **C++,** modelada na linguagem C. Originalmente chameda de "C com classes", essa linguagem obteve a padronizacao ANSI (o que significa que a linguagem é a mesma em todas as plataformas) em 1989 e veio a se tornar uma das principais linguagens de programacao de todos os tempos, continuando ate hoje a ser amplamente utilizada.

Em 1991, James Gosling, Patrock Naughton e Mike Sheridan, da **Sun Microsystems,** criaram a linguagem **JAVA**. Por volta de 1995, com a popularizacão da internet, o foco da linguagem Java mudou e passou a ser comercializada pela Sun como ambiente e linguagem de programacao multiplataforma.

No ano de 2000, Microsoft criou a linguagem de programacao **C#** que só está disponivel na plataforma Windows. Estas ultimas três linguagens de programacao (C++, Java e C#) são, hoje em dia, as mais difundidas em termos de programacao orientada a objetos.

Vale ressaltar que a POO não foi simplesmente "criada", mas se desenvolveu apartir de boas idéias e praticas comuns dos programadores.

Especificamente neste trabalho teremos como base para POO a linguagem **JAVA** de programação que será tratada com maiores detalhes mais a frente.

2.2- MOTIVAÇÃO DE POO;

Há várias décadas os programadores vem lidando com a enorme tarefa de automatizar processos. No comeco os programas continham uma longa série de instrucoes que serviam para executar uma tarefa especifica. Quando a febre da automacão se espalhou, as empresas comecaram a exigir que os programas de computadores gerenciassem e executassem diversas tarefas. Com isso, os programas comecaram a crescer em tamanho e complexidade, tornando-se caros e, as vezes, inadministraveis. Nesse contexto, a chamada **programacão procedural (**na qual liguagens como C, Pascal, Fortran, etc estao baseadas) surgiu para salvar os programadores.

Como o próprio nome sugere, a programacao procedural esta baseada no conceito de ***procedimentos*** **(**tabem conhecidos como *funcões, método ou rotinas*), isto é, um agrupamento lógico de instrucões que destina-se a realizar uma tarefa bem definida. A programacão procedural simplificou a tarefa de criacão de programas, permitindo que os programadores modificassem e/ou atualizassem rapidamente centenas de linhas de instrucões, examinando algumas linhas de um procedimento.

Com esse novo paradigma os programadores comecaram a poupar muito tempo. Os novos programas acabavam utilizando partes de outros programas por meio da utilizacao de funcoes já criadas (os programadores chamam isso de *código reutilizável*). Isso permitia um código mais compreensivel e de mais fácil manutencão.

De fato, a programacao procedural havia revolucionado a forma de como os programas eram escritos, mas ainda restava um problema a ser resolvido: o mundo real, que é aquilo que um programa de computador tenta imitar, não é organizado em procedimentos, mas sim em objetos!

2.3- CONCEITOS BÁSICOS EM POO;

2.3.1- PENSANDO EM OBJETOS;

De fato, a POO mudou a forma de se conceber os programas de computadores. Entretanto, embora POO seja um paradigma inovador, seus fundamentos estão embasados em conceitos que fazem parte do cotidiano de todo ser humano.

O conceito fundamental de objeto , por exemplo, nasce da visão que ser humano tem do mundo que o cerca. Todos nós, seja qual for o ponto de vista, vemos o mundo como uma coleção de objetos. Um objeto pode ser qualquer "coisa" que existe(não necessariamente palpável) : uma casa, uma pessoa, um computador, um carro, etc.(a figura1 mostra alguns exemplos de objetos do mundo real) E se formos mais além um podemos dizer que alguns objetos sao formados de outros objetos: os automoveis, por exemplo, são concebidos como um agrupamentos de objetos, como roda, volante, portas, som, etc

Num sistema orientado a objetos podem existir um número indefinidos de objetos que interagem entre si para a execução de funções propostas para o sistema.

***Figura 1-Alguns exemplos de objetos do mundo real**

2.3.2- ATRIBUTOS E COMPORTAMENTOS;

No mundo da programação, os objetos são descritos com base em dois grupos de características: *os atributos e os comportamentos. (**Em particular em java os comportamentos são chamados de métodos)**.* Um atributo é um aspecto do objeto e um comportamento, por sua vez, é uma ação que o objeto pode realizar. Para entendermos melhor, vamos analisar o objeto que, para nós, talvez seja o mais familiar: nós mesmos! Uma pessoa é um objeto. Uma pessoa, por exemplo, tem nome, altura, peso, idade, etc..(esses são alguns de seus muitos atributos). Além disso, uma pessoa pode andar, falar, correr, sentar, enfim pode ter inúmeros comportamentos.

Em programacão os atributos representam o estado interno de um objeto que podem ser modificados ao longo da execução de um programa. É imporatante notar ainda que o atributo de um objeto pode ser outro objeto (um carro, por exemplo, possui um motor como um de seus atributos).

Nos programas os comportamentos sao conhecidos como métodos ou funcões. Quando um objeto executa um serviço, dizemos que ele apresenta um determinado comportamento.

2.3.3- MENSAGENS;

Os objetos comunicam-se entre si atraves de mensagens que emitem e recebem. Quando uma mensagem é enviada devem ser especificados:

*um receptor;

*um requisição de serviços;

*argumentos ou parametros (se necessário);

Caso o serviço solicitado possa ser realizado, o receptor aceita e responde à mensagem ativando o método correspondente. Esse método executa alguma operação sobre seus dados (atributos). Quando um objeto é criado, o acesso a suas características é feito através de mensagens. Para cada mensagem recebida pelo objeto existe um método associado para respondê-la ***(em java as mensagens são representadas atraves das chamadas de métodos).*** Os argumentos ou parametros são informações adicionais necessárias a execução de

um métodos.

```
instance.andar()  //Exemplo de chamada de metodo em java
```

Imaginemos o exemplos do objeto pessoa dirigindo um objeto carro. Entre os muitos comportamentos possíveis de serem realizados pelo carro está a capacidade de frear. Para realizar tal tarefa, entretanto, é necessário que o objeto pessoa envie uma mensagem ao carro (no caso pressionado o pedal de freio). Nesse caso a pessoa ainda envia uma informação adicional (argumento) ao carro: a intensidade com que ela pressiona o pedal informa a intensidade da freada.

2.3.4- CLASSES;

Já falamos muito sobre os objetos. Agora vamos falar um pouco das entidades que dão origem aos objetos: as classes. Vamos pensar nas classes como se fossem moldes para a criaçãa de objetos. Pensem nas classes como se fossem uma fôrma de bolo por exemplo. Com ela podemos porduzir diversos bolos distintos mudando somente o sabor da massa. De fato podemos ter bolos de chocolate, morango, balmilha, etc, mas se forem feitos com uma mesma fôrma, de certo terão a mesma forma.

Em POO uma classe é um conjunto de intruções do programa que informa ao computador os atributos e comportamentos de todos os objetos que forem criados com base nessa classe .

```
//Exemplo de definição de classe em java
public class Mamiferos{
        String nome;
public void aleitar(){};
}
```

Objetos costumam ser chamardos de instância de uma determinada classe, o que significa dizer que ele possui os comportamentos e características definidos pela sua classe. Este processo é chamado de instanciação.

Desta forma, pode-se dizer que todo objeto pertence a uma classe. É conveniente, portanto, colocar o nome da classe no plural, já que uma classe define os atributos e métodos de todas as intâncias(objetos) da mesma.

É importante ressaltar que os objetos de uma determinada classe não são iguais, mas similares. A similaridade reside no fato de possuirem as mesmas propriedades.

A figura 2 mostra um diagrama de classes em que são ressaltados, entre outras coisas, os atributos e métodos que estao associados as classes mamíferos, cão, gato, etc

2.3.5- ABSTRAÇÃO E HIERARQUIZAÇÃO;

Abstracão é a capacidade de se olhar apenas uma parte de um todo, ou seja, retirar da realidade apenas as entidades e fenômenos considerados essenciais. Através do mecanismo de abstracão, o ser humano pode entender formas complexas, ao dividi-las em partes e analisar cada parte separadamente. Nesse

contexto a abstração refere-se à capacidade de modelar o mundo real. Podemos ainda considerá-la como um mecanismo pelo qual restringimos o nosso universo de análise e as variáveis e constantes que compõem esse universo, desprezando os dados que não nos interessa na análise

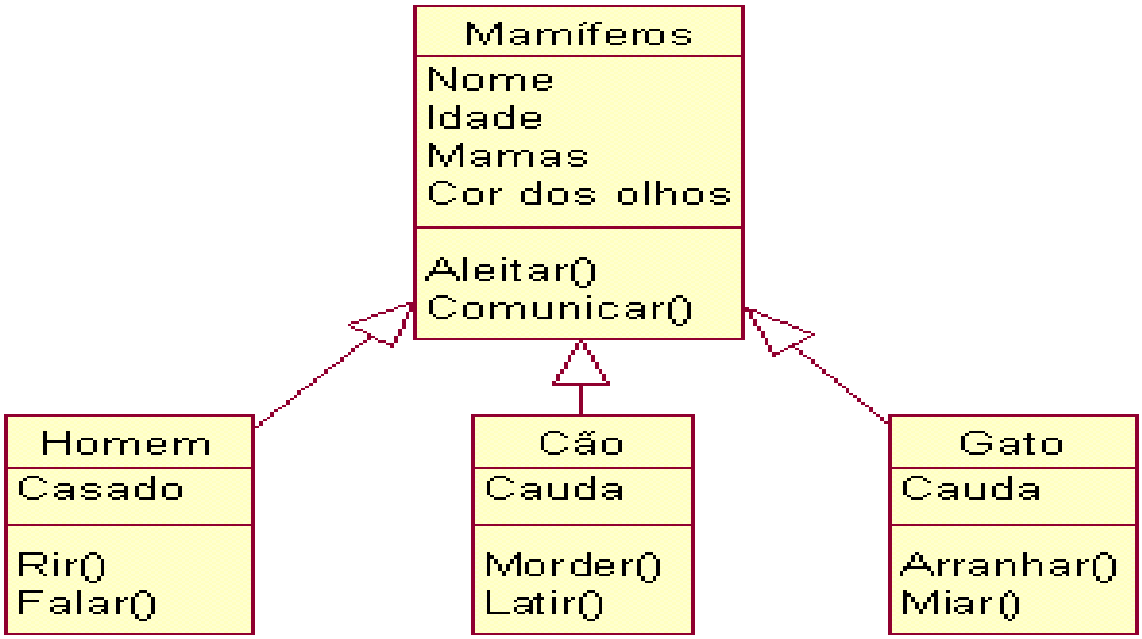

Figura 2- Diagrama de classe;

Podemos demostrar o uso de abstração facilmente, quando fechamos os olhos e pensamos em uma mesa; esta mesa imaginária provavelmente não vai ser igual à uma outra imaginada por outras pessoas, mas o que importa é que todos as pessoas que imaginaram uma mesa, colocaram nessa as informações que para elas são necessárias para a sua função (de ser uma mesa!). Não importa se a mesa é de três pés ou quatro, ou se o tampão é de vidro, madeira ou mármore; o que importa é que a imagem que idealizamos em nossa cabeça é de uma mesa e tenha as informações necessárias para cumprir sua função
O conceito de abstrção é muito usado em POO quando um objeto do mundo real, a qual será imitado no programa, possuir muitos atributos e compotamentos. Nesse caso, deve abstrair ao máximo, de modo a se implementar somente dados e métodos necessários ao objetivo do projeto do programa, descartando aspectos triviais.
A hierarquização por sua vez está relacionada a bastração no sentido de que após dividirmos em várias partes um problema complexo é comum ordená-las de alguma forma, isto é hirearquizá-las.
Hierarquizar um todo não significa dar prioridade a determinadas partes. A escolha de qual subsistema é fundamental para o sistema depende exclusivamente do observador.
É importante frisar que o conceito de abstração e hierarquização (e alguns

outros) já existiam desde à época da programacão estruturada (procedural) e foi incorporada pela POO. Aproveitando o ensejo, vale ressaltar que muitos conceitos foram herdados da programação estruturada por serem considerados bosa praticas de programacão. E a POO acrescentou a esses novos principios como objetos, classes, herança, encapsulamento, etc.

2.3.6- ENCAPSULAMENTO;

Encapsulamento é um mecanismo que utilizamos para esconder detalhes de estruturas complexas, que de outra forma iriam interferir em nossa anáilise. Na prática o objetivo é proteger os dados, isto é restringir o acesso aos atributos por determinados métodos de classe. Isso garante que nenhum dado será alterado por engano ou de forma descontrolada.

Encapsulamento garante que a classe seja uma caixa preta para o utilizador. Ele não sabe o que há dentro do objecto, sabe apenas para que ele serve e quais os métodos disponíveis para a manipulação deste.

Vejamos um exemplo: ao dividirmos o corpo humano em cabeça, tronco e membros, estamos escondendo o fato de que a cabeça por sua vez ser composta de vários elementos. Estes elementos estão encapsulados no objeto cabeça.

Em java o encapsulameto é implementado usando-se as palavras-chave public, private, e protected.

obs: *Abstração, hierarquização e encapsulamento* estão relacionados. Como vimos, *abstração* é a capacidade de dividirmos estruturas em partes, para analisá-las separadamente. Conforme vamos decompondo sucessivamente cada uma das partes de um todo, geramos uma *hierarquia*, que se estende até o componente mais específico. Desta forma quando agregamos componentes, formando um novo todo, estamos *encapsulando* estes componentes, que não mais aparecerão.

3.6- Herança;

Através do mecanismo de herança a classe mais especializada herda os métodos e atributos da classe mais geral. Por exemplo imaginemos a classe Pessoa. Dentre outros atributos, ela possui nome e idade. Agora imginemos uma outra classe denominada Funcionários. Digamos que os atributos necessários para implementar essa classe seja nome, idade, rg, cpf e funcao na empresa. Nesse caso, poderíamos utilizar o mecanismo de heranca, fazendo com que a classe funcionários herde os atributos e métodos da classe pessoa. Nesse exemplo, nao seria preciso mais definir os atributos nome e idade na classe funcinários, pois já existiam na classe pessoa.

De um modo geral, a classe que herdou uma outra classe é chamada de ***subclasse ou classe derivada*** e a que serviu de herança é denominada de ***superclasse ou classe pai.*** Em nosso exemplo, pessoa é superclasse e funcionários é a classe derivada.

obs: Em JAVA herda-se uma classe colocando-se a palavra-chave extends.

A figura 2, mostra tambem um típico exemplo de herança.

2.4- POR QUE USAR POO?

A POO não é difícil, mas é uma forma especial de pensar, as vezes até subjetiva de quem a programa, de forma que a maneira de fazer as coisas possa ser diferente de acordo com o programador. Embora possamos fazer um programa de formas distintas, nem todas elas são corretas! O difícil, portanto, não é programar orientado a objetos e sim programar bem! Programar é importante pois assim podemos aproveitar as vantagens da POO.
A orientação a objetos foi o maior avanço na área de software destes últimos anos. Ela nos permite criar sistemas melhores e, além disso, de maneira mais fácil. As técnicas estruturadas desde o final da década de 70 (quando foram lançadas) sempre obtiveram grande aceitação. Contudo, a medida que foram sendo utilizadas, a decomposição funcional mostrou-se inadequada em situações de sistemas complexos e principalmente para profissionais iniciantes.
A Orientação a Objetos traz vários benefícios no desenvolvimento e manutenção de software. Para melhor compreenção dessas vantagens vamos dividi-las em dois grupos. No primeiro, que chamaremos de "Vantagens Diretas" colocamos aquelas que representam consequências diretas da adoção da Orientação a Objetos e , no segundo grupo, as "Vantagens Reais", estarão aquelas que são de fato o que procuramos com essa tecnologia.
***Vantagens Diretas** :
1 - Maior facilidade para reutilização de código e por consequência do projeto;
2 - Possibilidade do desenvolvedor trabalhar em um nível mais elevado de abstração;
3 - Utilização de um único padrão conceitual durante todo o processo de criação de software;

***Vantagens Reais**
1 - ciclo de vida mais longo para os sistemas;
2 - desenvolvimento acelerado de sistemas;
3 - possibilidade de se construir sistema muito mais complexos, pela incorporação de funções prontas ;
4 - menor custo para desenvolvimento e manutenção de sistemas;

***A Reutilização**

Os objetos que compoem um sitema pederão ser aproveitados em outros projetos de software sem qualquer alteração. Caso haja necessidade um objeto, por exemplo, poderia ser criado herdando caracterísiticaticas de outros objetos já existentes.

***Manutenibilidade**

Como o objeto encapsula todos os mecanismos de seu funcionamento, caso haja necessidade de alterá-lo, os

sistemas que dependem deste objecto não necessitarão de qualquer alteração. Isto é possível pois como o funcionamento do objecto foi escondido do sistema, este não sofrerá nenhum impacto com alterações internas ao objecto.

***Productividade**

A partir do momento que temos a disposição uma coleção de classes devidamente testadas e com um funcionamento a prova de erros, para criar novos sistemas basta usar estes objetos, sem nenhuma necessidade de refazer código. Este aspecto traduz-se em maior qualidade no software desenvolvido, pois
os objectos já estão devidamente testados e seguros.
Isto gera, sem sombra de dúvida, maior rapidez e, consequentemente, produtividade no desenvolvimento de sistemas OO.
Com a metodologia da POO, passa a haver uma similaridade entre as linguagens do analista e do utilizador, já que os objetos dizem respeito a dados do mundo real, que fazem parte do dia a dia do usuário.
Os softwares desenvolvidos através da POO são mais rápidos de serem desenvolvidos, possuem maior qualidade, e são maisc barato.
Vejamos um exemplo: Construção de aparelhos electrônicos na industria.
Para construir um rádio ou um microcomputador montamos o projeto do aparelho (sistema) baseado em componentes electrônicos (objetos) disponíveis no mercado, que possuem características (atributos) e comportamentos (métodos) bem definidos pelos fabricantes de componentes. Basta ler a documentação do componente para montarmos o aparelho electrônico.
Caso um determinado componente seja alterado internamente pelo fabricante no sentido de melhorá-lo ou corrigi-lo de algum eventual problema, basta substitui-lo sem a necessidade de alterar todo o aparelho electrônico, desde que suas características e comportamentos não mudem.

3-JAVA: UMA PODEROSA LINGUAGEM DE PROGRAMAÇÃO;
3.1-HISTÓRICO DE JAVA;

Em 1991, na Sun Microsytems foi iniciado o Green Project, o berço do Java uma linguagem de programação orientada a objetos. Os mentores do projeto eram Patrick Naughton, Mike Sheridan, e James Gosling. O objetivo do projeto não era a criação de uma nova linguagem de programação, mas antecipar e planejar a "próxima onda" do mundo digital. Eles acreditavam que em algum tempo haveria uma convergência dos computadores com os equipamentos e eletrodomésticos comumente usados pelas pessoas no seu dia-a-dia.

Para provar a viabilidade desta idéia, 13 pessoas trabalharam arduamente durante 18 meses. No verão de 1992 eles emergiram de um escritório de Sand Hill Road no Menlo Park com uma demonstração funcional da idéia inicial. O protótipo se chamava *7 (lê-se "StarSeven"), um controle remoto com uma interface grafica *touchscreen*. Para o *7 foi criado um mascote, hoje amplamente conhecido no mundo Java, o **Duke(figura3)**. O trabalho do Duke no *7 era ser um guia virtual ajudando e ensinando o usuário a utilizar o equipamento. O *7 tinha a habilidade de controlar diversos dispositivos e aplicações. James Gosling especificou uma nova linguagem de programação para o *7. Gosling decidiu batizá-la de "Oak", que quer dizer *carvalho*, uma árvore que ele podia observar quando olhava pela sua janela.

O próximo passo era encontrar um mercado para o *7. A equipe achava que uma boa idéia seria controlar televisões e vídeo por demanda com o equipamento. Eles construíram um *demo* chamado MovieWood, mas infelizmente era muito cedo para que o vídeo por demanda bem como as empresas de TV a cabo pudessem viabilizar o negócio. A idéia que o *7 tentava vender, hoje já é realidade em programas interativos e também na televisão digital. Permitir ao telespectador

***Figura3- Duke**

interagir com a emissora e com a programação em uma grande rede cabos, era algo muito visionário e estava muito longe do que as empresas de TV a cabo tinham capacidade de entender e comprar. A idéia certa, na época errada.

A sorte é que o estouro da Internet aconteceu, e rapidamente uma grande rede interativa estava se estabelecendo. Era este tipo de rede interativa que a equipe do *7 estava tentando vender para as empresas de TV a cabo. E, da noite para o dia, não era mais necessário construir a infra-estrutura para a rede, em um golpe de sorte, ela simplesmente estava lá. Gosling foi incumbido de adaptar o Oak para a Internet e em janeiro 1995 foi lançada uma nova versão do Oak que foi rebatizada para **JAVA**. A tecnologia Java tinha sido projetada para se mover por meio das redes de dispositivos heterogêneos, redes como a Internet. Agora aplicações poderiam ser executadas dentro dos *browsers* nos Applets Java e tudo seria disponibilizado pela Internet instantaneamente. Foi o estático html dos *browsers* que promoveu a rápida disseminação da dinâmica tecnologia Java. A velocidade dos acontecimentos seguintes foi assustadora, o número de usuários cresceu rapidamente, grandes fornecedores de tecnologia, como a IBM anunciaram suporte para a tecnologia Java.

Desde seu lançamento, em maio de 1995, a plataforma Java foi adotada mais rapidamente do que qualquer outra linguagem de programação na história da computação. Em 2003 Java atingiu a marca de 4 milhões de desenvolvedores em todo mundo. Java continuou crescendo e hoje é uma referência no mercado de desenvolvimento de *software*. Java tornou-se popular pelo seu uso na Internet e hoje possui seu ambiente de execução presente em web browsers mainframes, SOs, celulares, palmtops e cartões inteligentes, entre outros.

3.2- O QUE É JAVA?

Java é uma tecnologia. É um mundo tão grande que ninguém se arrisca a dizer: Eu sei Java!. Basicamente constitui-se de uma linguagem de programação e um programa para execução chamado de **máquina virtual** ou **virtual machine** (discutida mais adiante). Quando programa-se em Java usa-se a linguagem de programação Java e um ambiente de desenvolvimento Java para gerar um software que será executado em um ambiente de distribuição Java. Tudo isso é a tecnologia Java.

JAVA foi desemvolvida pela empresa Sun Microsystems é uma linguagem parecida com a C++(mas não herdou as complexidades de C++.) e constituída de tal forma a **possibilitar que seus programas, depois de compilados , possam rodar em qualquer plataforma (talvez essa seja uma das maiores vantagens de java).**

Java já nasceu na rede, e foi feita para a rede, enquanto outras linguagens são limitadas pela arquitetura da maquina aonde roda, sem falar na segurança, Java roda praticamente em qualquer lugar e fala a linguagem da Internet.

3.3- POSIÇÃO DA LINGUAGEM JAVA;

A linguagem JAVA está em todo lugar nas revistas e suplementos de informática

dos jornais, nas notícias quentes sobre tecnologia na internet, nos sites e ultimamente até nos microondas, geladeiras e eletrodomésticos em geral, graças a tecnologia JAVA.
E todo mundo aposta alto na idéia. Praticamente todas as empresas de porte do mundo da informática tem produtos voltados para a linguagem. Só para citar alguns: Oracle, Sun Microsystems, IBM, Nokia Corporation, Hewlett-Packard, Borland Software Corporation, Apache Software Foundation, SAP AG, SavaJe Tenologies, Apple Computer Inc, Cisco Systems, Fujitsu Limited, Macromedia Inc, Rational Software, Unisys, America Online, Hitachi LTDA, Mitsubishi Eletric Corp, NEC, Sony Internacional, Sharp, Ericsson Inc, Matsushita Eletric, Motorola, Samsung Eletronics, Siemens AG, Symbian, Novell, PalmSource Inc, BEA Systems, Object People e muitas outras.

3.4- O FUNCIONAMENTO DAS APLICAÇOES JAVA;

Java tem a fantástica característica de multiplataforma devido a peculiaridades no funcionamento de suas aplicações: programas Java não são traduzidos para a linguagem de máquina como outras linguagens estaticamente compiladas e sim para uma representação intermediária, chamada de ***bytecodes*.**
Os bytescodes são interpretados pela máquina virtual Java (JVM - *Java Virtual Machine*). Muitas pessoas acreditam que por causa desse processo, o código *interpretado* Java tem baixo desempenho. Durante muito tempo esta foi uma afirmação verdadeira. Porém novos avanços tem tornado o compilador dinâmico (a JVM), em muitos casos, mais eficiente que o compilador estático.(ver figura4)

Java hoje já possui um desempenho próximo do C++. Isto é possível graças a

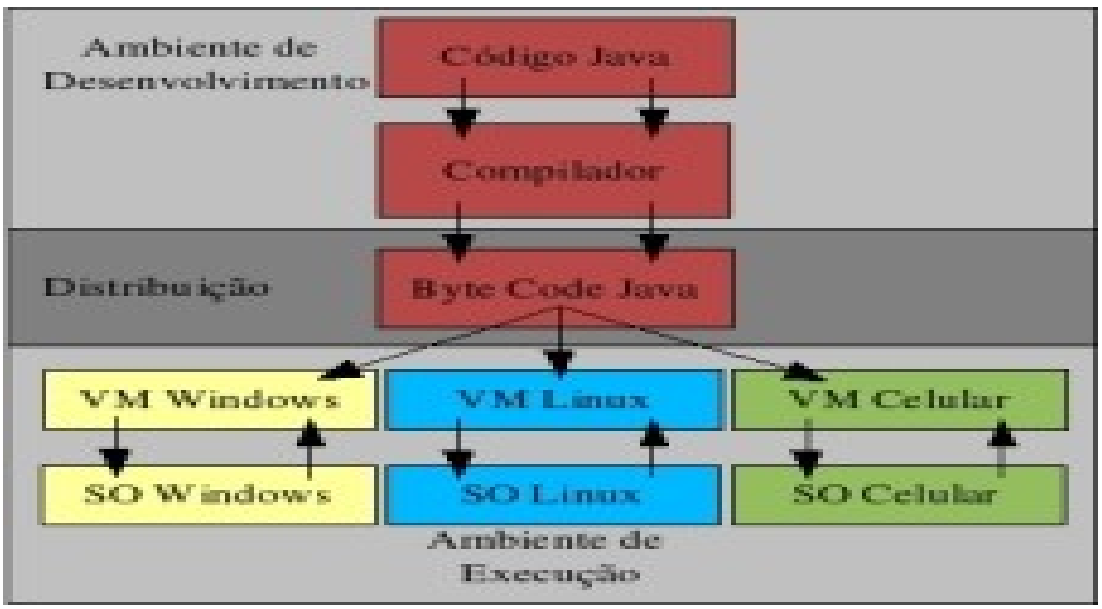

Figura4-Funcionamento das aplicações java;

otimizações como a compilação especulativa, que aproveita o tempo ocioso do processador para pré-compilar *bytecode* para código nativo. Outros mecanismos

ainda mais elaborados como o HotSpot da Sun, que guarda informações disponíveis somente em tempo de execução (ex.: número de usuários, processamento usado, memória disponível), para otimizar o funcionamento da JVM, possibilitando que a JVM vá "aprendendo" e melhorando seu desempenho. Isto é uma realidade tão presente que hoje é fácil encontrar programas corporativos e de missão crítica usando tecnologia Java. No Brasil, por exemplo, a maioria dos Bancos utiliza a tecnologia Java para construir seus *home banks*, que são acessados por milhares de usuários diariamente. Grandes sítios como o *eBay* utilizam Java para garantir alto desempenho. E a cada ano Java tem se tornado mais rápido, na medida que se evolui o compilador dinâmico.

Essa implementação no entanto tem um desvantagem intrínseca. A pré-compilação exige tempo, o que faz com que programas Java demorem um tempo significamente maior para começarem a funcionar. Soma-se a isso o tempo de carregamento da máquina virtual. Isso não é um grande problema para programas que rodam em servidores e que deveriam ser inicializados apenas uma vez. No entanto isso pode ser bastante indesejável para computadores pessoais onde o usuário deseja que o programa rode logo depois de abri-lo..

Os ***bytecodes*** produzidos pelos compiladores Java podem ser usados num processo de engenharia reversa para a recuperação do programa-fonte original. Esta é uma característica que atinge em menor grau todas as linguagens compiladas. No entanto já existem hoje tecnologias que "embaralham" e até mesmo criptografam os *bytecodes* praticamente impedindo a engenharia reversa.

3.5- O QUE SAO JAVA-APLLETS?

Applets são pequenos programas escritos em Java, e chamados de dentro de uma pagina HTML. Essa linguagem de programação foi desenvolvida especialmente para programas a serem utilizados em redes.
Esses programas são complemente independentes de plataforma : um único programa Java roda em Windows, Mac, UNIX, OS/2, Amiga, etc.
Os applets são usados principalmente para pequenos efeitos coo animações e joguinhos, ou como interface com o usuário para programas no servidor.
Mas o Java também pode ser usado para escrever programas de verdade, no momento o Wordperfect está sendo escrito em Java.
Para criar Java applets é necessário Ter conhecimentos de programação Java, agora para inserir Java applets prontos é muito fácil.
Um aapplets será sempre derivada da classe applet fornecida pela linguagem Java.
Java traz liberdade de escolha para quem desenvolve e para quem se serve da Informática. Liberdade na escolha de plataformas operacionais potentes e variadas. Liberdade na escolha de ferramentas de desenvolvimento consoante a sua qualidade e adequação ao projecto.
Java traz maior qualidade aos projectos. Qualidade ligada ao valor intrínseco das suas tecnologias, à estabilidade dos sistemas em que se apoia, e à disponibilidade de técnicos com excelente formação.
Java traz economia. Os utilizadores podem optar por entre uma gama variada de

sistemas de acordo com as suas necessidades, e a menores custos. Quem desenvolve tem também custos menores, sendo menores os investimentos necessários para a produção de software de qualidade.

3.6- QUE PODE SER FEITO COM JAVA?

TUDO! Java é uma linguagem que não se prende a nenhuma arquitetura e a nenhuma empresa, é rápida e estável. Pode construir sistemas críticos, sistemas que precisam de velocidade e até sistemas que vão para fora do planeta, como a sonda Spirit enviada pela Nasa para Marte. Java tem um mar de projetos open source, que estão lá, esperando por usuários e desenvolvedores.

3.7-AS TRES GRANDES EDIÇÕES;

JAVA se divide em três grandes edições.

- **Java 2 Standard Edition** (**J2SE**): É a tecnologia Java para computadores pessoais, notebooks e arquiteturas com poder de processamento e memória consideráveis. Várias APIs acompanham esta versão e tantas outras podem ser baixadas opcionalmente no site da Sun. É com elas que a maioria das aplicações são construídas e executadas. O J2SE possui duas divisões:
  - Java Development Kit (JDK) ou Standard Development Kit (SDK): um conjunto para desenvolvimento em Java e deveria ser instalado apenas pelos desenvolvedores por possuir ferramentas para tal tarefa.
  - Java Runtime Edition JRE: uma versão mais leve da JDK pois é preparada para o ambiente de execução, ou seja, é esta versão que executará os sistemas construídos com a SDK.
- **Java 2 Mobile Edition (J2ME)**: É a tecnologia Java para dispositivos móveis com limitações de memória ou processamento. Possui APIs bem simples e leves para economizar espaço, memória e processamento. São utilizadas para sistemas em celulares, palm tops, pocket pcs, smartphones, javacards e demais dispositivos. O J2ME se divide em dois grupos de bibliotecas. É dividida em dois grupos:
  - Connected Limited Device Configuration (CLDC): Para celulares e smartphones, que são mais limitados
  - Connected Device Configuration (CDC): Para Palmtops e Pocket pcs e alguns dispositívos mais poderosos.
- **Java 2 Enterprise Edition (J2EE)**: É a tecnologia Java para aplicações corporativas que podem estar na internet ou não. Possui um grande número de APIs onde a segurança é a principal preocupação. É ideal para a construção de servidores de aplicação, integração de sistemas ou distribuição de serviços para terceiros.

3.8- APLICAÇÕES JAVA;

Você já deve ter usado Java antes e não sabe. Por exemplo, em uma fila de banco, onde você fica jogando em seu telefone celular enquanto aguarda a sua vez. Os aplicativos feitos em Java estão presentes em uma infinidade de dispositivos, desde relógios até mainframes. Tudo isso graças a Máquina Virtual Java, que passaremos a chamar simplesmente de JVM a partir deste ponto.

A JVM é, em poucas palavras, um mecanismo que permite executar código em Java em qualquer plataforma. Segundo a definição da Sun, a JVM pode ser entendida como "uma máquina imaginária implementada via software ou hardware que executa instruções vindas de bytecodes".

Para servir de exemplo, suponha que você desenvolveu um aplicativo para um telefone celular. Com poucas modificações, você poderá rodar esse mesmo aplicativo em um palmtop, como mostra a figura5.

Figura5- Aplicação java em palmtop e celular

Outro exemplo: um fabricante de geladeiras constatou que é mais confiável controlar a temperatura desse eletrodoméstico por software. Assim, seus engenheiros criaram um programa para esse fim e ele foi feito em Java. Para executá-lo, eles criaram uma JVM para essa moderna geladeira.

Imagine agora que um fabricante de aparelhos de som desenvolveu um software em Java para permitir que um de seus produtos fosse compatível com músicas no formato MP3. Meses depois, a empresa lançou um outro aparelho e aproveitou esse mesmo recurso nele. Tudo isso, graças a JVM desenvolvida para os aparelhos.

Os exemplos acima deixam claro que praticamente todo dispositivo pode rodar aplicações em Java. Basta que ele tenha uma JVM. A implementação de uma JVM pode ser feita em hardware, como em chips, ou em software, como a JVM existente para o Linux.

3.9- FINALIZANDO;

Devido as características vistas anteriormente (e outras que não foram citadas neste artigo), a linguagem Java tem sido cada vez mais utilizada. O fato de se tratar de uma linguagem multi-plataforma, permite o desenvolvido de aplicativos e soluções para os mais diversos fins. E um dos grandes responsáveis por isso são as JVMs. Até mesmo você pode desenvolver uma JVM para um dispositivo qualquer, desde que tenha conhecimentos sólidos para isso. Sim, pois construir uma JVM não é fácil. Ela envolve uma série de conceitos complexos, como instruções equivalentes a de processadores, controle de acesso à memória, registradores, etc. Isso pode até parecer estranho, mas é necessário entender que as JVMs atuam, de fato, como uma máquina.

Mesmo que você não seja um desenvolvedor em Java, é interessante ter uma JVM em seu computador. Certamente você usará algum programa que usa recursos da linguagem Java. Portanto, procure pela JVM desenvolvida para seu sistema operacional no site da Sun (http://java.sun.com) e bom proveito!

3.10-(texto complementar) Quebrada segurança de Máquina Virtual Java;

O estudante indiano Sudhakar Govindavajhala, atualmente estudando na Universidade Princeton (Estados Unidos), iniciou um verdadeiro alvoroço na comunidade mundial de informática, ao anunciar que consegue quebrar o código de uma máquina virtual Java rodando em qualquer processador, desde um PC comum, até um "smart card" (cartão inteligente), utilizando apenas uma lâmpada e um pequeno programa escrito em Java.

Após fazer uma palestra descompromissada em um simpósio sobre segurança no Institute of Electrical and Electronic Engineers (IEEE), demonstrando o funcionamento da sua teoria em máquinas virtuais Java e .Net, o estudante teve sua agenda lotada, concedendo palestras privativas para profissionais da NASA, Intel, fabricantes de "smart cards" e da gigante IBM. Para preocupação de todas essas empresas, o método do estudante funciona muito bem na maioria dos sistemas. Mas o pior é que nenhum sistema até agora testado mostrou-se imune ao ataque.

A descoberta da falha é, sem dúvida, a mais importante dos últimos tempos e deverá agitar a comunidade envolvida com segurança da informação e com os fabricantes de hardware pelos próximos anos. Não será mais possível fabricar memórias de computador como se fez até hoje, sob pena de perda de credibilidade junto aos clientes, uma vez que não é possível garantir-se a segurança de praticamente nenhum sistema de segurança baseado em software. O risco é maior para equipamentos portáteis

4-CONCLUSÃO

De fato, desde o surgimento dos primeiros computadores por volta da decada de 40, os profissionais da area vem tentando utilizar cada vez mais os computadores nas mais variadas tarefas.

Para tanto, como foi visto neste trabalho, tem-se desenvolvido as mais diversas tecnicas de programação e, consequentemente os mais diversos estilos de software. Entretanto, é difícil afirmar qual é, de fato, a melhor tecnica de programação. Há quem diga que isso varia de acordo com a aplicação que se dejeja contruir.

O presente trabalho incubiu-se, portanto, de mostrar um modo especifico de programar(POO) e mostrar suas varias vantagens. Alem disso, procuramos em nosso texto incentivar os leitores com aplicação das tecnicas de POO em um linguagem especifica, Java, que embora fabulosa tem suas deficiencias.(como a mostrada no texto complementar)

Portanto, o mundo aqui mostrado é pequeno. Mas ai está o universo a ser conquistado!

5-REFERÊNCIAS BIBLIOGRAFICAS

[1]-**JIM KEOGH e MARIO GRANNINI,** OOP desmistificado – Programação orientada a objetos, Altas books, 2005.

[2]- **HARVEY M. DEITEL e PAUL J. DEITEL**, Java: como programar, Bookman, 2002.

[3]- **http://www.mundojava.com.br**

[4]- **http://www.guj.com.br/posts**

[5]- http://www.portaljava.com