

Desenvolvimento de Software

Cássio Seffrin

Desenvolvimento de Software

- Metodologia Procedural (PE)
- Metodologia Orientada o Objetos (POO)
- Objeto (estrutura de dados que representam estado / comportamento)
- Fortran IBM (1954)
- Basic 65 Microsoft
- Simula 67 POO
- Pascal foi criada em 1970 pelo suíço Niklaus Wirth
- Smalltalk 72-80 POO
- C(73), C++ (80), C#, Objective C/Swift
- Python 1991 POO
- Java James Gosling (1995): primeiro lançamento publico: 23/05/1995
- PHP Rasmus Lerdorf (1994)
- Ruby (1995)
- Lua (nmap Scripting Engine) - The Lua language is designed, implemented, and maintained at PUC-Rio in Brazil since 1993 (Waldemar, Roberto, Luiz)

Desenvolvimento de Software

O imperativo é um dos modos verbais, juntamente com o modo indicativo e o modo subjuntivo. No modo imperativo a pessoa falante leva o seu interlocutor a realizar uma ação, expressando o que quer que ele faça.

Assim, a ação transmitida por um verbo no imperativo é um pedido, convite, exortação, ordem, comando, conselho ou súplica.

O imperativo se divide em imperativo afirmativo e imperativo negativo, sendo conjugados de forma diferente. Em ambos não existe flexão na 1.^a pessoa do singular (eu).

Exemplos de uso do modo imperativo

Pare com essa brincadeira.

Jogue o lixo fora, por favor.

Resolva esse problema rápido.

Vai fazer o dever de casa.

Sai da frente!

Para!

Desenvolvimento de Software

“Na Ciência da Computação, programação imperativa é um paradigma de programação que descreve a computação como ações, enunciados ou comandos que mudam o estado (variáveis) de um programa. Muito parecido com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma sequência de comandos para o computador executar. O nome do paradigma, Imperativo, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo... Este paradigma de programação se destaca pela simplicidade, uma vez que todo ser humano, ao se programar, o faz imperativamente, baseado na ideia de ações e estados, quase como um programa de computador.”

Fonte wiki 2020

Desenvolvimento de Software

Exemplos de linguagens de programação que baseiam-se no modelo imperativo:

Assembler

Basic

C/C++/C#

Cobol

Fortran

Java

Javascript

Lua

Pascal

PHP

Python

Ruby

TypeScript

Desenvolvimento de Software

Exemplos de linguagens de programação que baseiam-se no modelo funcional:

Javascript
TypeScript
LISP
Clojure
Python
Erlang
Haskell

Linguagens funcionais puras - esses tipos de linguagens funcionais suportam apenas os paradigmas funcionais. Por exemplo - Haskell.

Linguagens Funcionais Impuras - Esses tipos de linguagens funcionais oferecem suporte aos paradigmas funcionais e à programação de estilo imperativo. Exemplo - LISP.

Desenvolvimento de Software

Exemplos de linguagens de programação que baseiam-se no modelo POO:

Simula

Smalltalk

C++, C#, Objective C

Swift

TypeScript

Java

Dart

Python

Clojure

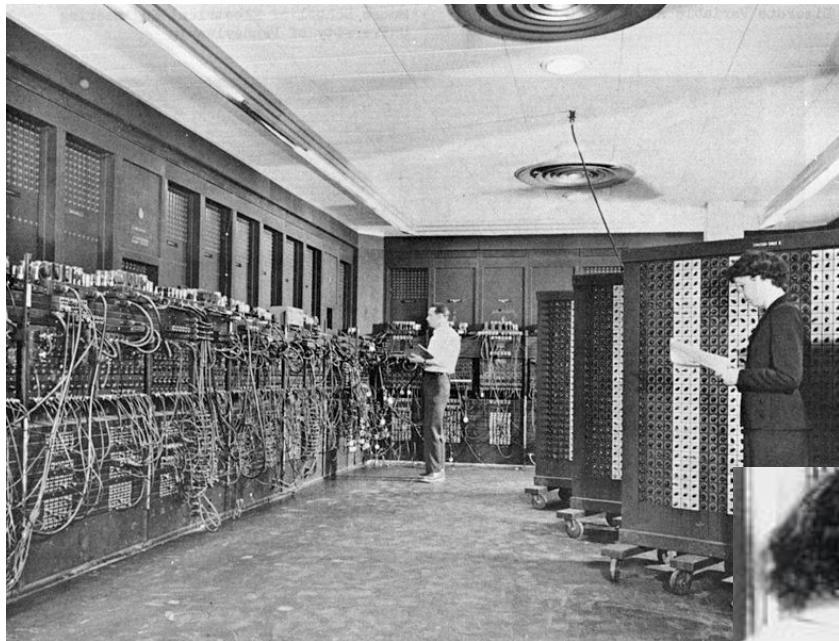
Desenvolvimento de Software

JavaScript é uma linguagem multi-paradigma, permite misturar e combinar paradigmas orientados a objetos, procedurais e funcionais. Recentemente, tem havido uma tendência crescente para a programação funcional. Em estruturas como Angular e React, você realmente obterá um aumento de desempenho usando estruturas de dados imutáveis. A imutabilidade é um princípio básico da programação funcional.

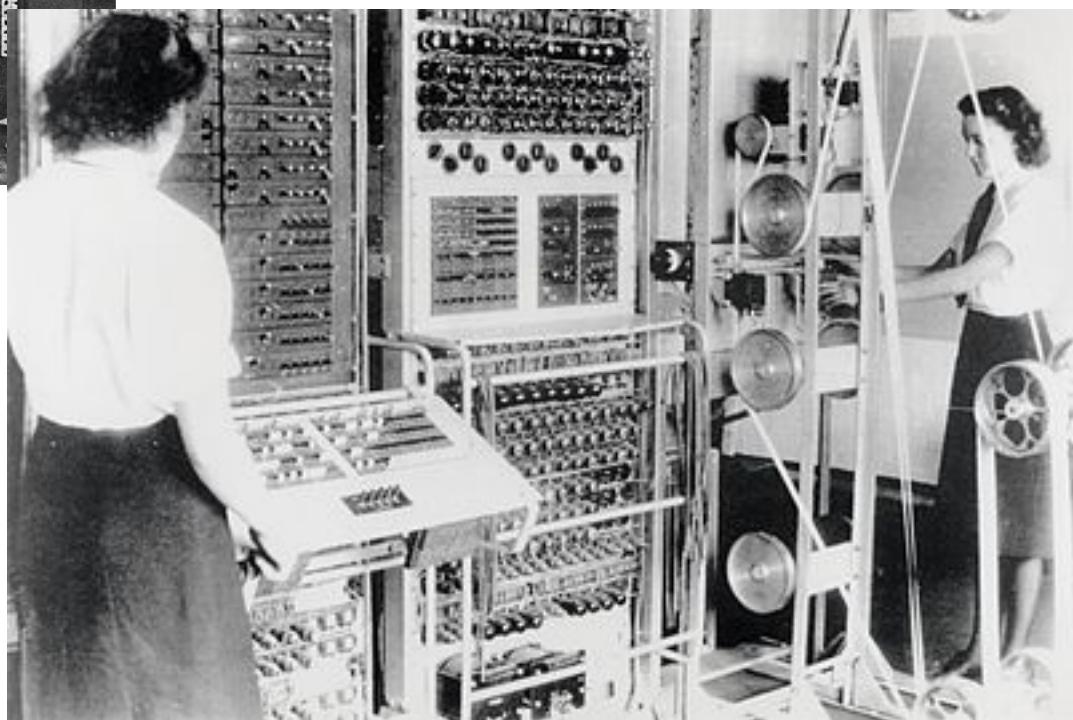
Uma função pura é uma função que possui as seguintes propriedades: Seu valor de retorno é o mesmo para os mesmos argumentos (sem variação com variáveis estáticas locais, variáveis não locais, argumentos de referência mutáveis ou fluxos de entrada de dispositivos de E / S)

ImmutableJS

Immer



ENIAC 1945 - 1947 - US

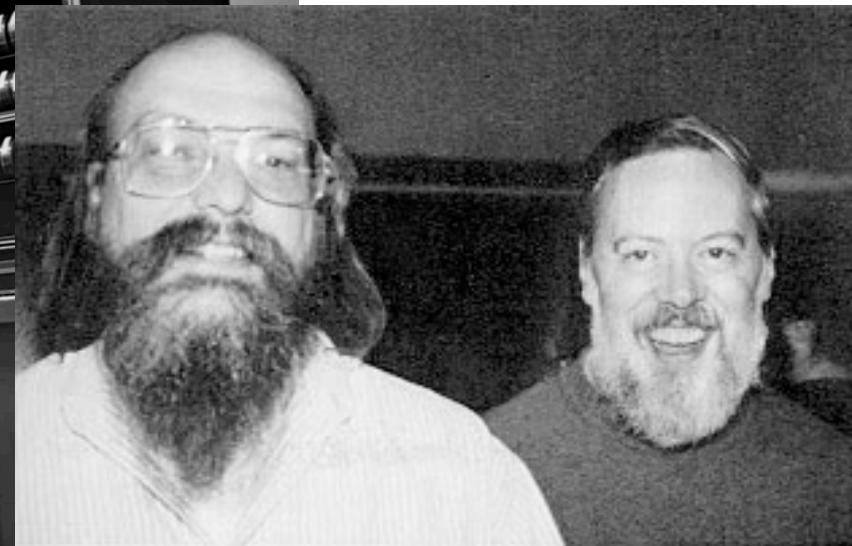


Computador Colossus sendo operado UK
Mark I- Dezembro de 1943 e
Mark II – 1 de Junho de 1944

Simula 67, cuja primeira versão foi apresentada em 1966 foi a 1ª linguagem orientada a objetos e introduziu os conceitos de classes e herança.

Exemplo de Classe em Simula:

```
Glyph Class Line (elements);
    Ref (Glyph) Array elements;
Begin
    Procedure print;
    Begin
        Integer i;
        For i:= 1 Step 1 Until UpperBound (elements, 1) Do
            elements (i).print;
        OutImage;
    End;
End;
```



Ken Thompson e Dennis Ritchie (C 1972 - Unix 1983)

AT&T Bell Labs entre 1969 e 1973



Richard Matthew Stallman: FSF e GNU

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

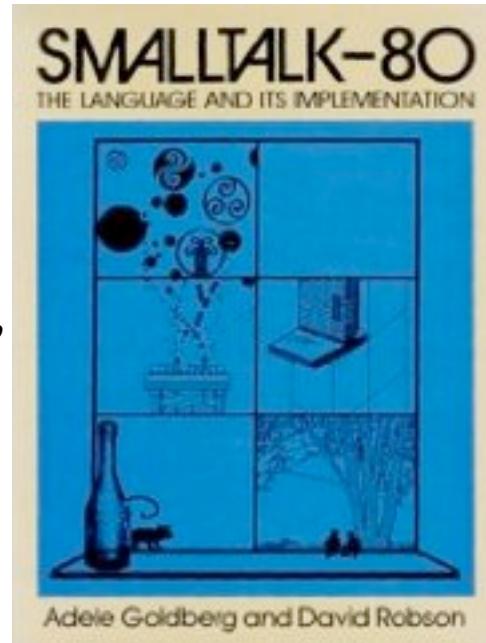
I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

— Linus Torvalds

Smalltalk-80, ou simplesmente Smalltalk, é uma linguagem de programação orientada a objeto fracamente tipada.



Em *Smalltalk* tudo é objeto: os números, as classes, os métodos, blocos de código, etc. Não há tipos primitivos, ao contrário de outras linguagens orientadas a objeto; strings, números e caracteres são implementados como classes em *Smalltalk*, por isso esta linguagem é considerada puramente orientada a objetos. Tecnicamente, todo elemento de *Smalltalk* é um objeto de primeira ordem.



James Gosling

Linguagem Java

- Oak (árvore de carvalho)
- Java
- POO
- Compilador Bytecode (javac Pessoa.java -> Pessoa.class)
- Tipagem forte
- Classes
- Atributos
- Construtores
- Métodos
- Modificadores de acesso (private, protected (subclasse), public)
- Outros Modificadores (final/static/abstract)
- Objeto

Linguagem Java

História e retrospectiva

O primeiro lançamento público do Java foi em 23 de maio de 1995, como um alfa disponível apenas no sistema operacional Solaris da Sun Microsystem. Desde aquele primeiro lançamento, Java se tornou um dos ambientes de programação mais amplamente implantados do mundo. Ela ganhou força e participação no mercado em essencialmente todos os ambientes de programação, especialmente software corporativo, desenvolvimento da Web do lado do servidor e programação de telefones móveis.

Java é uma linguagem de colarinho azul. Não é material de tese de doutorado, mas uma linguagem para um trabalho.

“James Gosling”

Foi uma longa jornada com muitas surpresas ao longo do caminho. Essa jornada começou nos primeiros dias com James Gosling e sua equipe, que tinham certeza de que os desenvolvedores queriam recursos de produtividade avançados (como orientação a objetos e coleta de lixo), mas sentiram que muitos engenheiros estavam assustados com a complexidade das linguagens que os implementaram .

Linguagem Java

O design de alto nível de Java

Em termos de design de linguagem, Java sempre foi baseado em uma série de decisões de design deliberadas e opinativas com objetivos específicos em mente. Os objetivos primários iniciais da plataforma Java podem ser resumidos como:

Para fornecer um contêiner para execução simples de código de aplicativo orientado a objeto

Para remover a contabilidade tediosa das mãos dos desenvolvedores e tornar a plataforma responsável pela contabilidade da memória

Para remover vulnerabilidades de segurança da plataforma C / C ++ sempre que possível

Para permitir a execução multiplataforma

Notavelmente, esses objetivos foram perseguidos mesmo às custas do controle do desenvolvedor de baixo nível e do custo de desempenho nos primeiros anos.

Linguagem Java

Ao eliminar quase completamente o custo de portabilidade, o Java permitiu que os desenvolvedores se concentrassem na solução de problemas de negócios.

“Georges Saab”

A meta de portabilidade foi entusiasticamente chamada de “Escreva uma vez, execute em qualquer lugar” (WORA). Ele representa a ideia de que os arquivos de classe Java podem ser movidos de uma plataforma de execução para outra e executados sem alterações. Ele depende da existência e disponibilidade de uma Java Virtual Machine (JVM) para a plataforma host. Portar Java para uma nova plataforma torna-se, portanto, uma questão de implementar uma JVM que será executada na nova plataforma de acordo com a especificação da máquina virtual (geralmente chamada de VMspec).

Uma breve história de Java

O mundo em que Java chegou era muito diferente daquele em que vivemos hoje. A Microsoft estava em contagem regressiva para o lançamento do Windows 95 em agosto (que seria lançado sem um navegador da Web). O Netscape ainda não tinha se tornado público (IPO) na bolsa NASDAQ, embora seu navegador estivesse ganhando popularidade cada vez mais. A Internet, como um todo, ainda não havia entrado verdadeiramente na consciência do público.

Linguagem Java

Como um prenúncio do que estava por vir, o lançamento inicial da linguagem Java da Sun foi acompanhado pelo HotJava, um navegador destinado a competir com o então dominante navegador Mosaic, e substituí-lo por meio da introdução de aplicativos mais ricos e dinâmicos usando a tecnologia de mini aplicativo Java. Apesar disso, poucos poderiam ter previsto o impacto que a versão aparentemente modesta do Java teria.

O rápido crescimento do interesse público na Internet, alimentado pelo IPO da Netscape e outros eventos de mercado, produziu condições que deram início a uma primeira onda de entusiasmo (e mais do que um pouco de exagero) para Java. Isso acabaria por levar a algumas consequências surpreendentes, entre as quais a renomeação de uma linguagem de script não relacionada para “Javascript”, a fim de lucrar com o perfil público do ecossistema Java.

Desde aqueles primeiros dias inebriantes, Java teve uma filosofia de design bastante conservadora e uma taxa lenta de mudança muitas vezes ridicularizada. Esses atributos, no entanto, têm um outro lado esquecido - uma tentativa consciente de proteger o investimento das empresas que adotaram a tecnologia Java.

*Após a sessão USENIX na qual James Gosling falou pela primeira vez publicamente sobre Java, as pessoas já estavam dançando sobre o túmulo de Java.
“Mike Loukides”*

Linguagem Java

Não apenas isso, mas a visão de Gosling foi mais do que justificada - as decisões de design dos primeiros dias de Java agora são consideradas incontroversas. Isso fornece um exemplo claro da velha máxima do software de que "as linguagens de programação mais odiadas são inevitavelmente as mais utilizadas". No caso do Java, isso se estende ao plágio das idéias e princípios de design do Java.

Por exemplo, muito poucos desenvolvedores de aplicativos tentariam defender a opinião de que a memória deve ser gerenciada manualmente atualmente. Mesmo as linguagens de programação de sistemas modernos, como Go e Rust, consideram um dado adquirido que o tempo de execução deve gerenciar a memória em nome do programador.

A linguagem Java passou por uma revisão gradual compatível com versões anteriores, mas nenhuma reescrita completa. Isso significa que algumas das escolhas de design originais do Java, feitas por conveniência devido às restrições e convenções da tecnologia do final dos anos 90, ainda estão restringindo a plataforma hoje.

Os primeiros anos do século 21 testemunharam o surgimento do Enterprise Java, fortemente impulsionado pela Sun (e mais tarde pela Oracle) como a forma futura de desenvolver aplicativos. Versões iniciais da plataforma (conhecidas originalmente como J2EE, e mais

Linguagem Java

Eventualmente, estruturas mais leves (por exemplo, Spring) surgiram. À medida que esses desafiadores adicionavam recursos, eles inevitavelmente aumentavam em tamanho e complexidade. Ao mesmo tempo, os padrões Java EE progrediram, eliminando camadas desnecessárias de configuração e começaram a se concentrar nas necessidades básicas dos desenvolvedores dentro da empresa.

Hoje, o espaço empresarial permanece vibrante, com competição saudável entre diferentes estruturas e abordagens para o desenvolvimento em grande escala. Nos últimos 10 anos, apenas o framework .NET da Microsoft ofereceu alguma competição séria ao Java para o desenvolvimento empresarial.

Enquanto isso, a plataforma Java principal (a “Standard Edition” ou Java SE) não estava parada no início dos anos 2000. Java 5, lançado em 2004, foi um marco significativo e introduziu mudanças importantes na linguagem central. Isso inclui tipos genéricos, tipos enumerados, anotações e autoboxing.

A biblioteca padrão e as principais interfaces de programação de aplicativos (APIs) também foram substancialmente atualizadas - especialmente nas áreas de programação simultânea e gerenciamento remoto (tanto para aplicativos quanto para a própria JVM).

Essas mudanças foram julgadas como uma grande mudança na evolução do Java, e nenhum lançamento até o Java 8 teria o mesmo impacto no mundo Java. A Sun também finalmente abandonou o esquema de nomenclatura “Java 1.X” e começou a usar o número principal, de forma que este lançamento foi o Java 5.

As ondas de mudança, desde mudanças de linguagem em Java 5, a atualizações técnicas de baixo nível, como compilação sob demanda ou Just-In-Time (JIT) (Java 1.3), até estruturas de procedimentos e padronização, como a Comunidade Java O processo, ou especificação da linguagem Java, levou o Java adiante como uma linguagem, plataforma e ecossistema.

Linguagem Java

História do Java de código aberto

Isso pode ser visto através da evolução da atitude da Sun (e posteriormente da Oracle) em relação à comunidade e ao código aberto. Apesar de declarar grande apoio ao código aberto, a Sun procedeu com cautela no que diz respeito à propriedade intelectual do Java.

O Java Community Process (JCP) foi estabelecido pela Sun em 1998 como uma forma de trazer outras empresas e partes interessadas para o processo de desenvolvimento de padrões Java. A Sun queria trazer rivais em potencial "para dentro da tenda", mas não queria perder o controle do Java no processo. O resultado foi um órgão da indústria que representou um compromisso entre empresas concorrentes que ainda tinham uma causa comum.

Respondendo à pressão do mercado e da comunidade em geral, Jonathon Schwartz (então CEO da Sun) anunciou o código-fonte aberto do Java ao vivo no JavaOne 2006. Diz a lenda que esse anúncio foi feito sem o conhecimento total de sua equipe de gerenciamento. Isso levou à criação do projeto OpenJDK (Open Java Development Kit) em 2007, que ainda hoje é responsável pelo desenvolvimento da implementação de referência da plataforma Java.

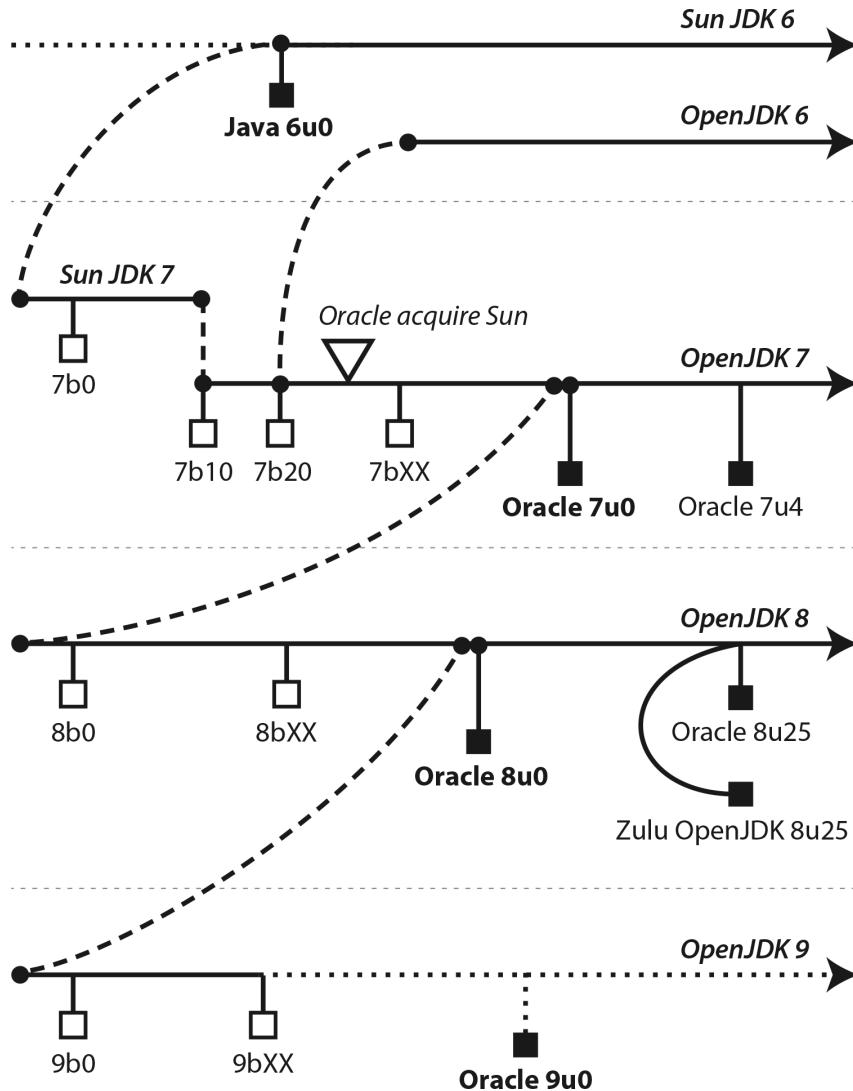
Linguagem Java

A Sun agora tinha um processo de padrões um tanto aberto e uma implementação de referência de código aberto para Java. No entanto, o caminho para um Java aberto não foi completamente tranquilo. O trem de lançamento do Java 6 já estava em andamento e parecia muito difícil tentar mover a cadeia de ferramentas da Sun e a prática de desenvolvimento para um processo aberto. Em vez disso, uma gota de código do JDK 7 proprietário da Sun em desenvolvimento foi obtida, depurada e lançada como a semente para o OpenJDK 6.

O lançamento foi ainda mais complicado pela decisão da Sun de isentar certos componentes do lançamento de código aberto, citando problemas na obtenção de acordo dos detentores dos direitos autorais.

Devido a essa abordagem, o OpenJDK 6 nunca teve um lançamento que correspondesse precisamente a um lançamento do Sun JDK. No entanto, o trem de lançamento do JDK 7 corrigiu esse processo, movendo-se rapidamente para um processo aberto e fazendo com que todos os commits normais fossem feitos diretamente nos repositórios abertos a partir de então. Este processo ainda é seguido hoje.

Linguagem Java



Linguagem Java

Para alguns participantes da comunidade Java aberta, a Sun não foi longe o suficiente no código-fonte aberto da plataforma. Por exemplo, o Kit de compatibilidade de teste (TCK) não foi aberto e continuou sendo um software proprietário. Isso representou um obstáculo para implementações de Java que não sejam da Sun, porque uma implementação deve passar 100% do TCK para ser certificada como uma implementação compatível de Java.

A Apache Foundation iniciou uma implementação de código aberto em sala limpa de Java em maio de 2005 e, no outono de 2006, tornou-se um projeto Apache de nível superior chamado Apache Harmony. Quando o projeto estava em andamento, o Apache abordou a Sun para um TCK. Embora não tenha dito abertamente "Não", Sun hesitou em aceitar o pedido. Isso era particularmente verdadeiro no que diz respeito às restrições de "Campo de Uso" que a Sun havia adicionado à licença Java SE TCK, que impedia o Java SE de ser executado legalmente em um telefone móvel.

Finalmente, em 10 de abril de 2007, a Apache Foundation enviou uma carta aberta à Sun exigindo uma licença de código aberto para o kit de teste que seria compatível com a licença Apache, em vez da GNU Public License (GPL) que o OpenJDK usa. A Sun não produziu uma licença adequada e a disputa continuou.

O Harmony seria usado como base para a estrutura de programação de aplicativos Android do Google para dispositivos móveis. Em 2015, ainda é a biblioteca padrão principal para desenvolvedores Android.

Linguagem Java

Após o lançamento do Java 6 em dezembro de 2006, a Sun embarcou em um plano ambicioso para o Java 7, que deveria crescer com o passar do tempo, de forma que no final de 2009 incluiu uma proposta para alguns dos mais procurados novos recursos.

Estes incluíam:

Expressões lambda e, com elas, um estilo de programação um pouco mais funcional

Um sistema completamente novo de modularização e empacotamento de classes Java para substituir o antigo formato Java Archive (JAR) que era baseado em arquivos ZIP

Um novo coletor de lixo chamado Garbage First (G1)

Suporte aprimorado para linguagens dinâmicas não Java em execução na JVM

Muitos na comunidade começaram a questionar quando, se é que chegaria, a versão 7 realmente chegaria. No entanto, eles logo descobririam que a Sun (e o Java com ele) potencialmente tinham problemas muito maiores do que apenas uma data de lançamento interminável.

Linguagem Java

The Age of Oracle

Em 2009, ficou claro que a receita e os lucros da Sun estavam com sérios problemas, e o Conselho da Sun começou a buscar compradores para o negócio. Após o fracasso das negociações com a IBM (e supostamente com a HP), a Oracle emergiu como um comprador e fez uma oferta substancial na primavera de 2009. Após obter a aprovação dos governos dos EUA e da UE, a aquisição da Sun foi concluída em 27 de janeiro de 2010.

Os primeiros meses após a aquisição contiveram alguns contratemplos e saídas de alto nível da Oracle (incluindo James Gosling, o inventor original do Java). Fora do espaço Java, a Oracle se saiu mal e foi seriamente criticada por lidar com algumas das tecnologias de código aberto que adquiriu da Sun.

No espaço Java, o problema estava para chegar no final de 2010, quando as negociações entre a Oracle e o Google sobre o licenciamento da tecnologia Java usada no Android fracassaram, e a Oracle processou o Google por violação de direitos autorais e patentes.

O ecossistema receberia outro choque, quando o Apache retirou oficialmente sua filiação ao JCP depois que a Oracle se recusou a conceder ao Apache uma licença TCK para Harmony em termos aceitáveis, apesar de ter apoiado o Apache contra a Sun na disputa original. O projeto Harmony lutou até novembro de 2011 antes de finalmente escolher se separar, com o Android sendo o único verdadeiro herdeiro da tecnologia.

Linguagem Java

O Apache ainda não retornou ao JCP, mas muitos dos projetos do Apache são implementados em Java ou outras linguagens JVM. A ação da Oracle / Google estava para durar até maio de 2012, quando o juiz decidiu decisivamente a favor do Google (embora alguns aspectos menores do caso continuem nos tribunais de apelação).

Com o fim efetivo do processo Oracle / Google e a incerteza inicial em relação ao desvanecimento da administração da Oracle em Java, a plataforma Java parece ter entrado em um período de relativa calma.

No lado da engenharia, isso foi visto em vários lugares - mais claramente no fato de que o plano monolítico para Java 7 foi cortado em pedaços mais gerenciáveis. Primeiro, alguns recursos relativamente simples e trabalho de engenharia interna foram enviados como Java 7 em julho de 2011. Em seguida, as expressões lambda e o suporte de programação funcional seguiram como parte do Java 8 em março de 2014. Finalmente, o suporte de modularidade atrasado deve chegar como parte do Java 9, que deve ser lançado em setembro de 2016.

Tipos Primitivos

		Valores possíveis				
Tipos	Primitivo	Menor	Maior	Valor Padrão	Tamanho	Exemplo
Inteiro	byte	-128	127	0	8 bits	byte ex1 = (byte)1;
	short	-32768	32767	0	16 bits	short ex2 = (short)1;
	int	-2.147.483.648	2.147.483.647	0	32 bits	int ex3 = 1;
	long	-9.223.372.036.854.770.000	9.223.372.036.854.770.000	0	64 bits	long ex4 = 1l;
Ponto Flutuante	float	-1,4024E-37	3.40282347E + 38	0	32 bits	float ex5 = 5.50f;
	double	-4,94E-307	1.79769313486231570E + 308	0	64 bits	double ex6 = 10.20d; ou double ex6 = 10.20;
Caractere	char	0	65535	\0	16 bits	char ex7 = 194; ou char ex8 = 'a';
Booleano	boolean	false	true	false	1 bit	boolean ex9 = true;

Linguagem Java

Exemplo de um método em java

```
class Matematica {  
    int multiplicar(int a, int b){  
        return a*b;  
    }  
}  
  
class Principal{  
    public static void main(String a[]){  
        Matematica mat = new Matematica();  
        System.out.println("resultado de 2*4: "+  
mat.multiplicar(2,4) );  
        System.out.println("raiz de 9: "+mat.raiz(9) );  
    }  
}
```

Linguagem Java

recursividade

- Recursividade é quando uma função/método chama ele mesmo.

```
public class FatorialRecursivo {

    public void imprimirNumero(int fatorial, long numero) {
        System.out.printf("O fatorial de %d é: %s",
fatorial,  numero);
    }

    public long fatorial(long numero) {
        if (numero <= 1) {
            return 1;
        }
        return numero * fatorial(numero-1);
    }
}
```

Linguagem Java

Elementos da operação de divisão:

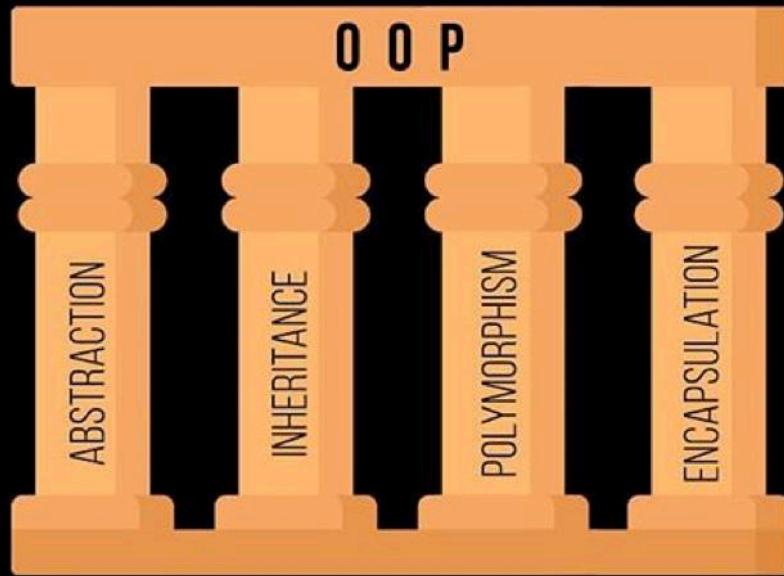
dividendo

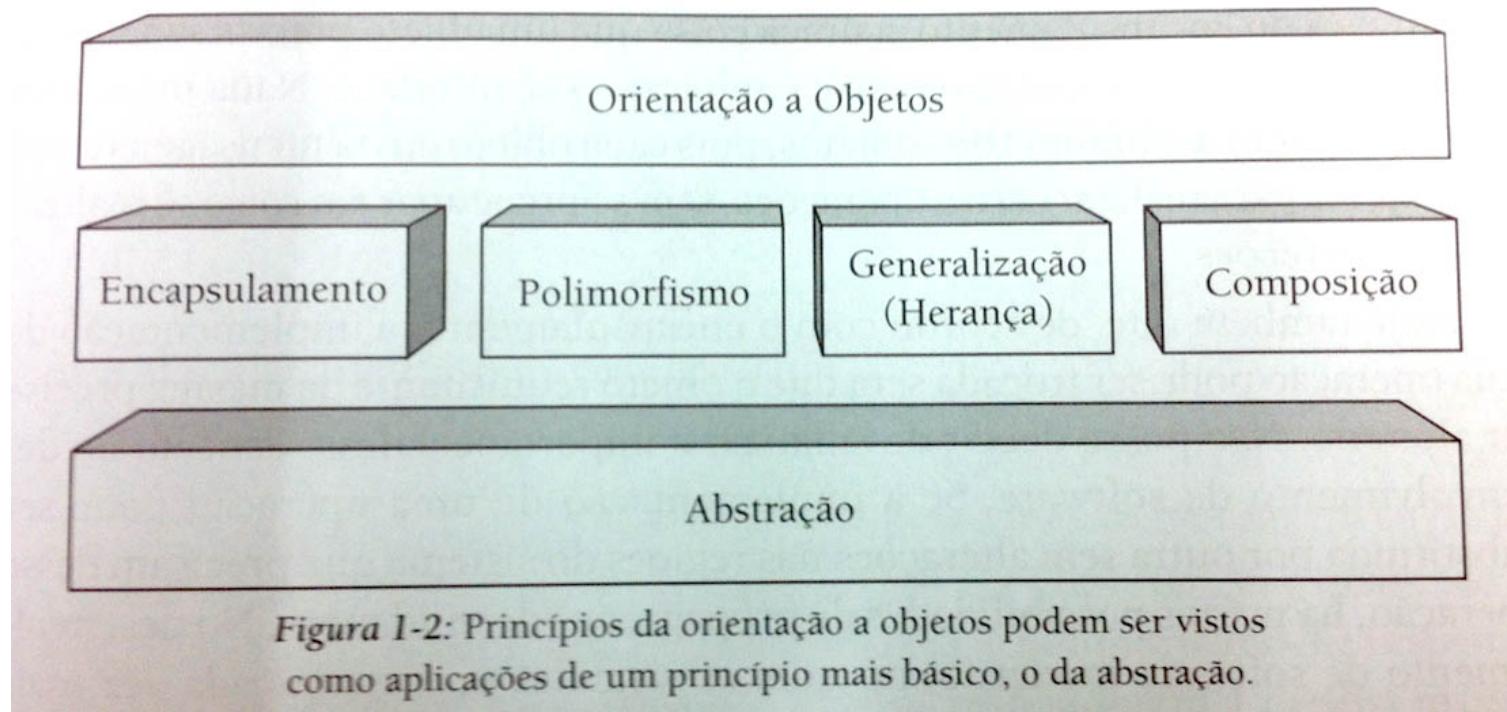


$$\begin{array}{r} 17 \\ \underline{-16} \\ 1 \end{array}$$

17 2 → divisor
-16 8 → quociente
 1 → resto

PILLARS OF OBJECT-ORIENTED PROGRAMMING

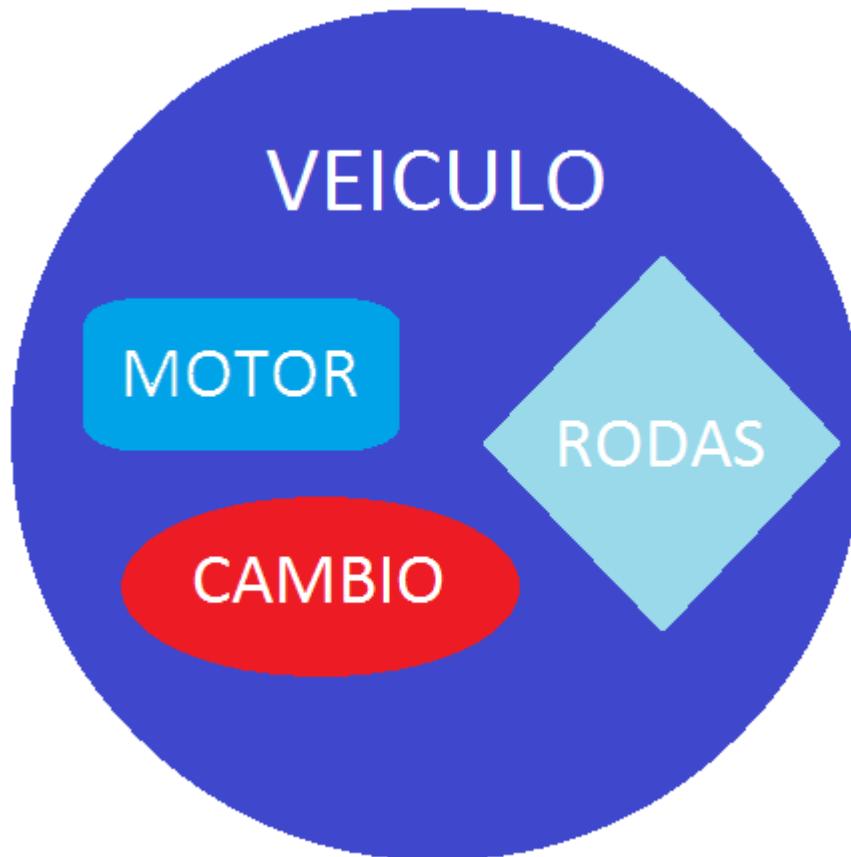




Fonte: Lorenzo Ridolfi, Sérgio Colcher (2007)

Linguagem Java

Exemplo de Composição



Linguagem Java

Exemplo de uma interface

```
interface BicicletaInterface {  
    void mudarMarcha(int newValue);  
    void aumentarVelocidade(int increment);  
    void freiar();  
}  
  
class Bicicleta implements BicicletaInterface {  
    /* comentário Java:  
       implementar os 3 métodos da interface  
       BicicletaInterface  
    */  
}
```

Linguagem Java

Exemplo de uma herança

```
abstract class Pessoa {  
    private String nome;  
    private Integer idade;  
    private String sexo;  
    private Date dataNascimento;  
        public Pessoa(){}
    /* .... Getters e setters .... */  
}
```

```
class Funcionario extends Pessoa{  
    private int salario ;  
    public Funcionario(){}
    /* .... Getters e setters .... */
```

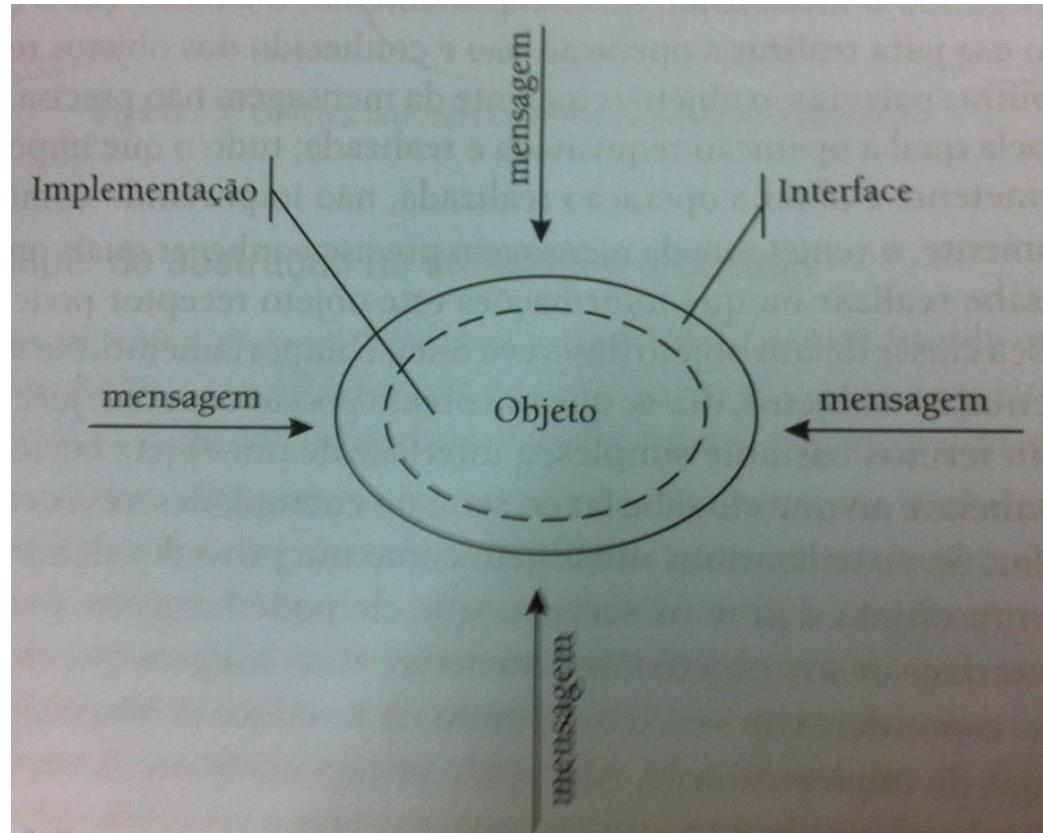
```
public static void main (String a[]){  
    Funcionario f = new Funcionario();  
    f.setNome("Cassio");  
    f.setSalario(2000);  
    System.out.println("Nome: "+ f.getNome() + " Salario: "  
+f.getSalario());  
}
```

Linguagem Java

Classes Abstratas

- Servem como modelo para uma classe concreta
- Não podem ser instanciadas diretamente
- Podem conter ou não métodos abstratos
- Pode implementar ou não um método

Modelagem de sistema de software



Fonte: Lorenzo Ridolfi, Sérgio Colcher (2007)

Encapsulamento

- Os objetos possuem comportamento (métodos)
- O encapsulamento é uma forma de restringir o acesso ao comportamento interno dos objetos.
- A relação entre objetos é feita através da troca de mensagens

Herança (Generalização)

- Na generalização, classes semelhantes são agrupadas em uma hierarquia.
- Exemplo: um funcionário estende de uma classe de pessoa.

Composição

- Objetos que compõe outros objetos
- Um livro é composto de paginas, capa, títulos, parágrafos.

Polimorfismo

- O polimorfismo indica a capacidade de abstrair varias implementações diferentes em uma única interface. (Lorenzo Ridolfi, Sérgio Colcher)
- O polimorfismo permite programar no geral ao invés de programar no específico (deitel & deitel).

Exemplos de polimorfismo.

Sobrescrita e Sobrecarga de Métodos;

Abstração

- Em java, a abstração é obtida por interfaces e classes abstratas. Podemos atingir 100% de abstração usando interfaces.
- O processo de abstração é usado para esconder certos detalhes e somente mostrar as características essenciais de um objeto, ou seja, apresentar uma visão externa de um objeto (interface)

Propostas genéricas não decidem casos concretos.
– Oliver Wendell Holmes

Um filósofo de estatura imponente não pensa em um vazio.
Mesmo suas ideias mais abstratas são, em alguma medida,
condicionadas pelo que é ou não conhecido na época em
que ele vive.
– Alfred North Whitehead

UML

- Grandy Booch, James Rumbaugh e Ivar Jacobson. Os 3 amigos.
- Em 1997 a UML foi aprovada como padrão pelo OMG (Object Management Group. Consórcio Internacional que define e ratifica padrões na área de orientação a objetos.

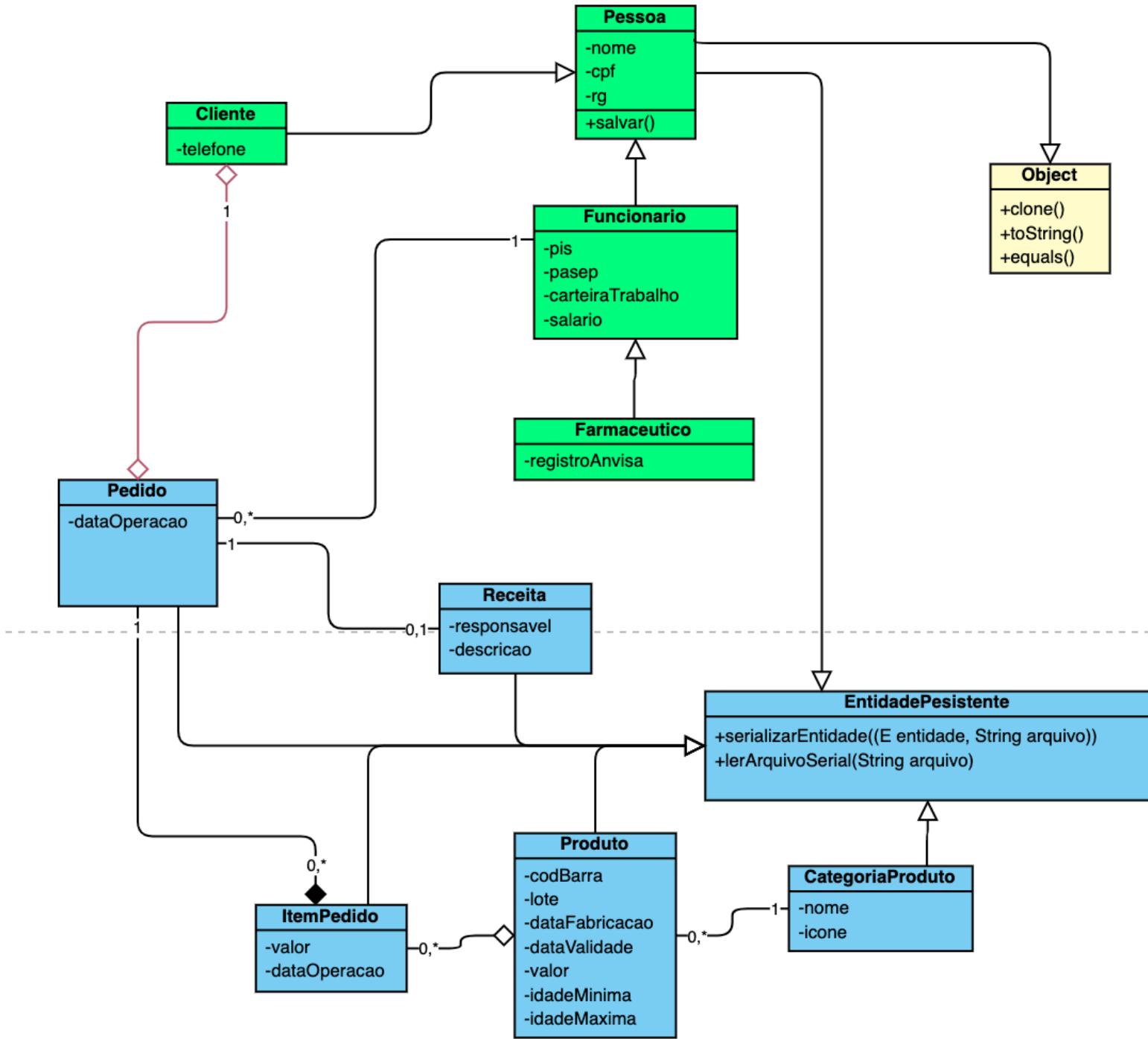
Modelagem de sistema de software

- Complexidade cresce a medida que o software aumenta
- Casa para o cachorro
- Casa para família
- Edifício
- Software
- Características tratadas com diversos modelos, analogamente ao um avião que possui um diagrama elétrico, outro para aerodinâmica.
- Gerenciamento da complexidade
- Comunicação entre as pessoas envolvidas
- Redução dos custos no desenvolvimento
- Previsão do comportamento futuro do sistema

Modelagem de sistema de software

O paradigma da orientação a objetos (uma forma de abordar um problema)

1. Qualquer coisa é um objeto
2. Objetos realizam tarefas por meio da requisição de serviços a outros objetos.
3. Cada objeto pertence a uma determinada classe. Uma classe agrupa objetos similares.
4. A classe é um repositório para comportamento associados ao objeto.
5. Classes são organizadas em hierarquias.



Generics

A partir do Java 5
O que é um tipo Genérico
Para que serve?
Exemplo de Array[] simples

Generics

```
Integer[] intArr = {1, 2, 3, 4, 5};  
Double[] douArr = {1.2, 2.1, 3.4, 4.5, 5.6};  
String[] strArr = {"Cassio", "Fred", "Juliana"};
```

Como imprimir sem Generics?
Como imprimir com Generics?

Generics

- Tipos genéricos em Arrays funcionam somente substituindo tipos por referencias como Integer, Float, String. Tipos primitivos (int, char, float) não funcionam
- Os nomes dos parâmetros de tipo por toda declaracao do método devem corresponder aqueles declarados na assinatura do método
- O tipo Object pode ser usado invés do conceito generics quando o retorno do método é void.

Generics

Convenção para determinar nomes para Genéricos

- E – Element (usado para todos elementos do Java Collections Framework, exemplo ArrayList, Set etc.)
- K – Key (usado na coleção Map)
- N – Number
- T – Type
- V – Value (usado na coleção Map)
- S,U,V etc. – T, U, V etc.. quando não sabemos o tipo
- U (undefined) = indefinido

Escrita de arquivos

```
File f = new File("/Users/cassioseffrin/teste.txt");
FileOutputStream fos = new FileOutputStream(f, true);
fos.write("nome; cpf; rg; endereco; ".getBytes());
fos.flush();
fos.close();
```

Leitura de arquivos

```
File f = new File("/Users/cassioseffrin/teste.txt");
FileInputStream fis = new FileInputStream(f);
Scanner scanner = new Scanner(fis);
while (scanner.hasNextLine()) {
    System.out.println(scanner.nextLine());
}
scanner.close();
```

Escrita Objetos

```
FileOutputStream fos = new FileOutputStream(f);
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(colecaoPessoa);
```

Leitura Objetos

```
FileInputStream fis = new FileInputStream(f);
ObjectInputStream ois = new ObjectInputStream(fis);
Object col = ois.readObject();
Set<Pessoa> colecaoDeserializable = (Set<Pessoa>) col;

for (Pessoa p : colecaoDeserializable) {
    System.out.println(p);
}
```

GIT servidor/cliente

```
servidor git
```

```
#mkdir repo
```

```
#cd repo
```

```
#git init --bare
```

```
cliente com projeto existente
```

```
entrar no diretório
```

```
git init
```

```
git remote add origin /Users/cassioseffrin/Desktop/repo2
```

```
git add .
```

```
git commit -m"primeiro commit"
```

```
git push origin master
```

```
cliente2 com git clone em outro host
```

```
#mkdir cliente1
```

```
#cd cliente1
```

```
#git clone cassio@192.168.0.102:/Users/cassioseffrin/Desktop/repo
```

```
#echo "teste" > teste.txt
```

```
#git add teste.txt
```

```
#git commit -m"primeiro commit"
```

```
#git push origin master
```

GIT servidor/cliente

Passo 1: dentro do diretório workspace fazer o clone

Passo 2: git clone <https://github.com/cassioseffrin/DevSoftware2021.git>

Pull Request (PR) no GitHub

Clicar o botão Fork no canto superior direito. Isso cria uma nova cópia do meu repositório de demonstração em sua conta de usuário do GitHub com um URL como:

`https://github.com/<seunome>/farmaciaMvnFx`

A cópia inclui todo o código, branches e commits do repositório original.

Em seguida, clone o repo abrindo o terminal em seu computador e executando o comando:

`git clone https://github.com/<seunome>/farmaciaMvnFx`

Depois que o repo é clonado, você precisa fazer duas coisas:

Crie uma nova branch com o seguinte comando:

`git checkout -b novaBranch`

Adicionar a URL do repositório:

`git remote add upstream https://github.com/cassioseffrin/farmaciaMvnFx`

Nesse caso, "repo upstream" se refere ao repositório original a partir do qual você criou seu fork.

Agora você pode fazer alterações no código. O código a seguir cria uma nova branch, faz uma alteração arbitrária e o envia para nova_branch:

Pull Request (PR) no GitHub

```
$ git checkout -b nova_branch
Switched to a new branch 'nova_branch'
$ echo "texto qualquer" > teste.txt
$ git status
On branch nova_branch
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    teste.txt
nothing added to commit but untracked files present (use "git add" to track)
$ git add teste.txt
$ git commit -m "commit na nova_branch"
[nova_branch (root-commit) 4265ec8] Adding a test file to new_branch
  1 file changed, 1 insertion(+)
   create mode 100644 teste.txt
$ git push -u origin nova_branch
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 918 bytes | 918.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
Remote: Create a pull request for 'nova_branch' on GitHub by visiting:
Remote: https://github.com/cassioseffrin/farmaciaMvnFx/pull/new/new_branch
Remote:
 * [new branch]      nova_branch -> nova_branch
```

Depois de fazer o push, voltar ao GitHub e verificar o botão verde Pull Request que irá aparecer.

Bibliografia

Lorenzo Ridolfi, Sérgio Colcher. Princípios de Análise e Projeto de Sistemas com UML,

FREEMAN & FREEMAN, Eric & E. Padrões de Projetos: Seu cérebro em padrões de projetos. Rio de Janeiro: ALTABOOKS, 2007.

METSKER, S. J. Padrões de projeto em Java. Porto Alegre: Bookman, 2004.

ANSELMO, F. Aplicando Lógica OO [Orientada a Objetos] em JAVA. 2. ed. Florianópolis: Visual Books Ltda., 2005. 178 p. ISBN 85-7502-162-1.