

The slide features a white background with abstract green geometric shapes. On the left, a solid green triangle points downwards. On the right, a complex arrangement of overlapping, semi-transparent green triangles and polygons creates a dynamic, layered effect. The main title is centered in a large, green, sans-serif font.

Arquiteturas, MVC e Componentes

Prof. Nelson Bellincanta Filho

Projeto de arquitetura

- ▶ O projeto de arquitetura visa compreender como um sistema de software deve ser organizado e projetar a estrutura geral desse sistema;
- ▶ No modelo do processo de desenvolvimento de software, o projeto de arquitetura é o primeiro estágio no processo de projeto de software;
- ▶ É o elo crítico entre o projeto e a engenharia de requisitos, pois identifica os principais componentes estruturais de um sistema e os relacionamentos entre eles.

Projeto de arquitetura

- ▶ A arquitetura é baseada em modelo de alto nível para permitir um maior entendimento e uma análise mais fácil do software que se pretende desenvolver;
- ▶ O resultado do processo de projeto de arquitetura é um modelo de arquitetura que descreve como o sistema está organizado em um conjunto de componentes de comunicação.

Projeto de arquitetura

- ▶ Segundo Sommerville (2018) é possível projetar as arquiteturas de software em dois níveis de abstração:
 1. A arquitetura em pequena escala está preocupada com a arquitetura de programas individuais;
 2. A arquitetura em grande escala preocupa-se com a arquitetura de sistemas corporativos complexos que incluem outros sistemas, programas e componentes de programas.

Padrões de arquitetura

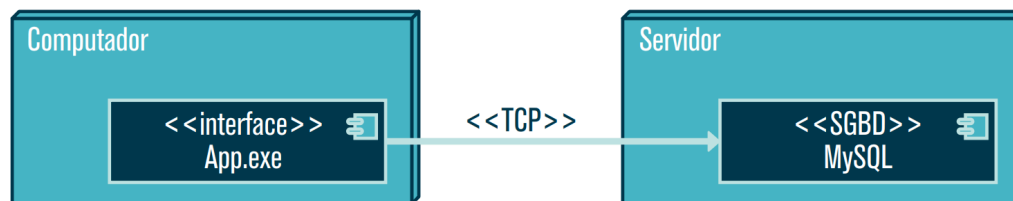
- ▶ Arquitetura monolítica:
 - ▶ Utilizada em projetos entre os anos de 1960-1980, voltados para ambiente Mainframe, onde havia o predomínio do uso da linguagem Cobol (COMmon Business Oriented Language);
 - ▶ Esses softwares têm como característica o processamento centralizado, com diversos terminais acessando a mesma Unidade Central de Processamento.
- ▶ Arquitetura multi-camadas:



Padrões de arquitetura

► Arquitetura multi-camadas:

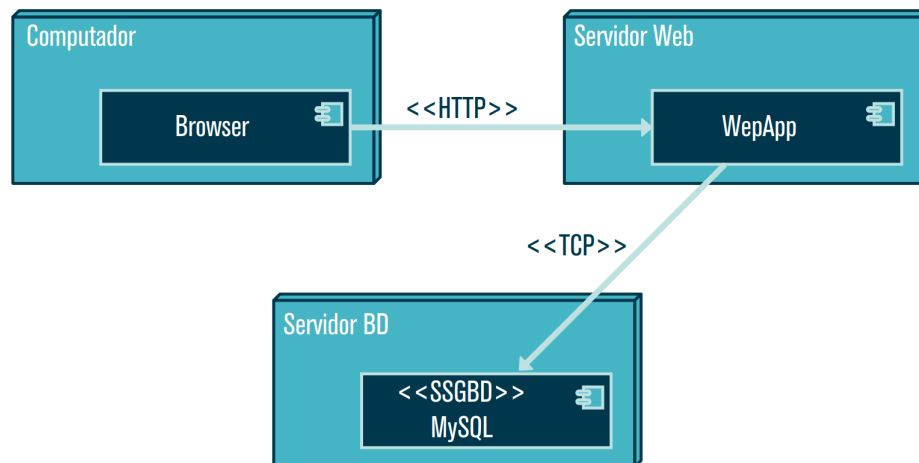
- Consiste numa evolução do modelo cliente servidor introduzindo uma ou mais camadas intermediárias de software;
- Cada camada fica responsável por parte da implementação do software, como se fossem elementos independentes, que se comunicam mutuamente afim de manter a harmonia no processamento dos dados.



Representação de um sistema 2 camadas.

Padrões de arquitetura

- ▶ Arquitetura multi-camadas:
 - ▶ Em um modelo de 3 camadas, há uma maior granularidade na distribuição das responsabilidades.



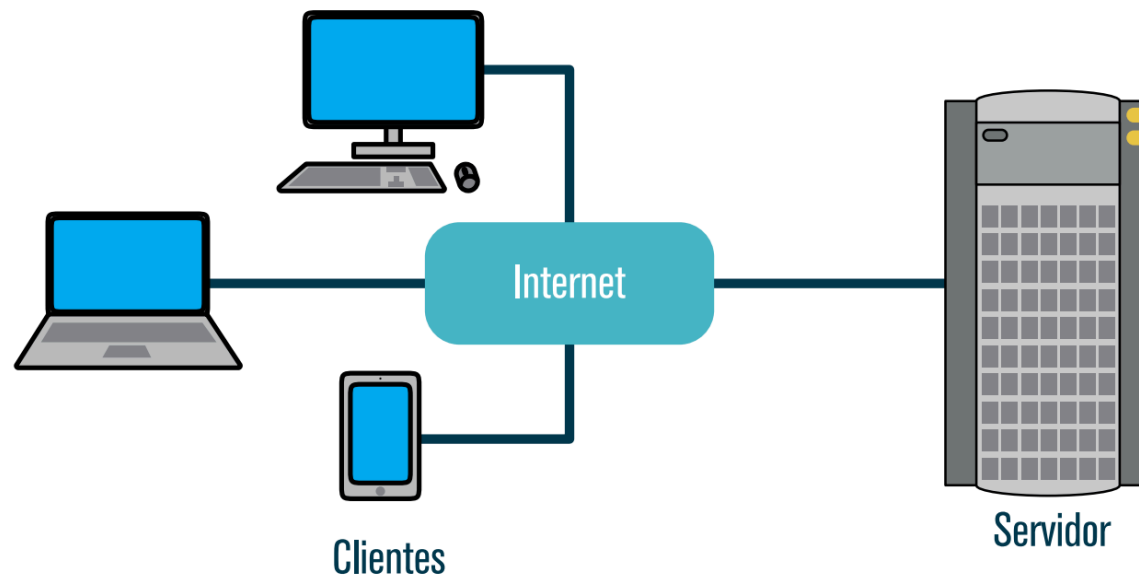
Representação de um sistema 3 camadas.

Padrões de arquitetura

► Arquitetura cliente-servidor:

- São aplicações desenvolvidas com dois macro módulos: o cliente e o servidor;
- Esses módulos podem interagir entre si através de protocolos de rede conhecidos, ou protocolos privados, presentes em redes locais;
- No módulo cliente, é oferecida uma interface para o usuário interagir, seja através da entrada de dados ou da leitura das informações processadas.
- No módulo servidor, encontra-se a aplicação implementada, contendo as regras de negócio das organizações.

Padrões de arquitetura



Representação da arquitetura Cliente Servidor.

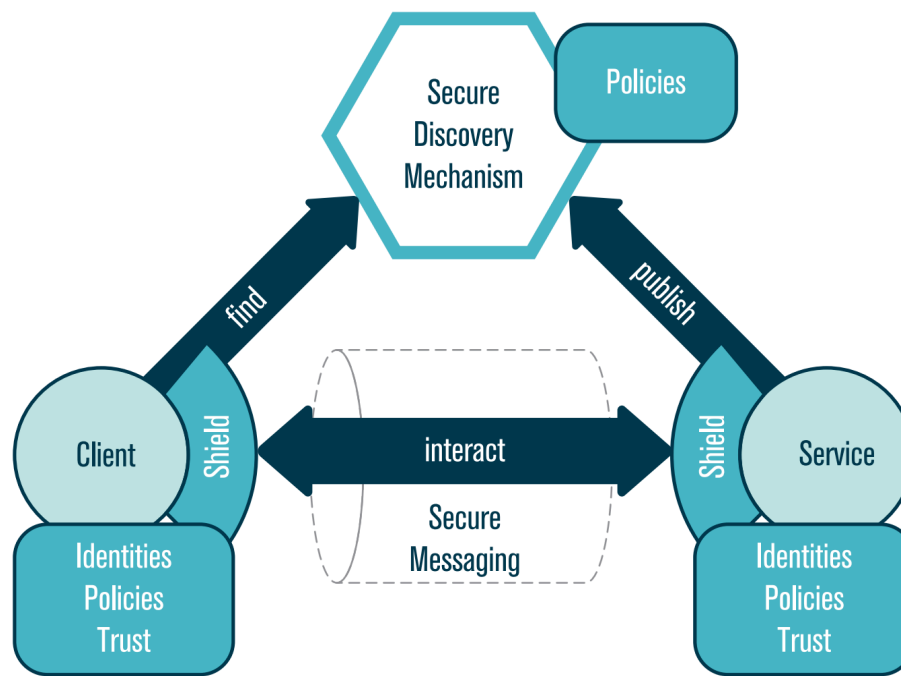
Padrões de arquitetura

- ▶ SOA - Service-Oriented Architecture ou Arquitetura Orientada a Serviço:
 - ▶ É um conceito de arquitetura, que tem como objetivo apresentar as funcionalidades implementadas pelas aplicações corporativas na forma de serviços;
 - ▶ Esses serviços são conectados a um componente conhecido como ESB (Enterprise Service Bus - barramento de serviços), que disponibiliza interfaces acessíveis através de webservices;
 - ▶ A arquitetura SOA é baseada nos princípios da computação distribuída e utiliza o paradigma Request/Reply (Requisição/Resposta), para estabelecer a comunicação entre os sistemas clientes e os sistemas que implementam os serviços.

Padrões de arquitetura

- ▶ SOA - Service-Oriented Architecture ou Arquitetura Orientada a Serviço:
 - ▶ A comunicação entre Cliente e Serviço ocorre através uma padronização de troca de informações;
 - ▶ Tecnologias baseadas em XML como SOAP (Simple Object Access Protocol) e WSDL (Web Services Description Language), desde o início dos anos 2000, são consideradas padrão de interoperabilidade entre os sistemas;
 - ▶ Outros tipos de padrões, como REST(Representational State Transfer), baseado em JSON (Java Script Object Notation).

Padrões de arquitetura



Representação da arquitetura SOA.

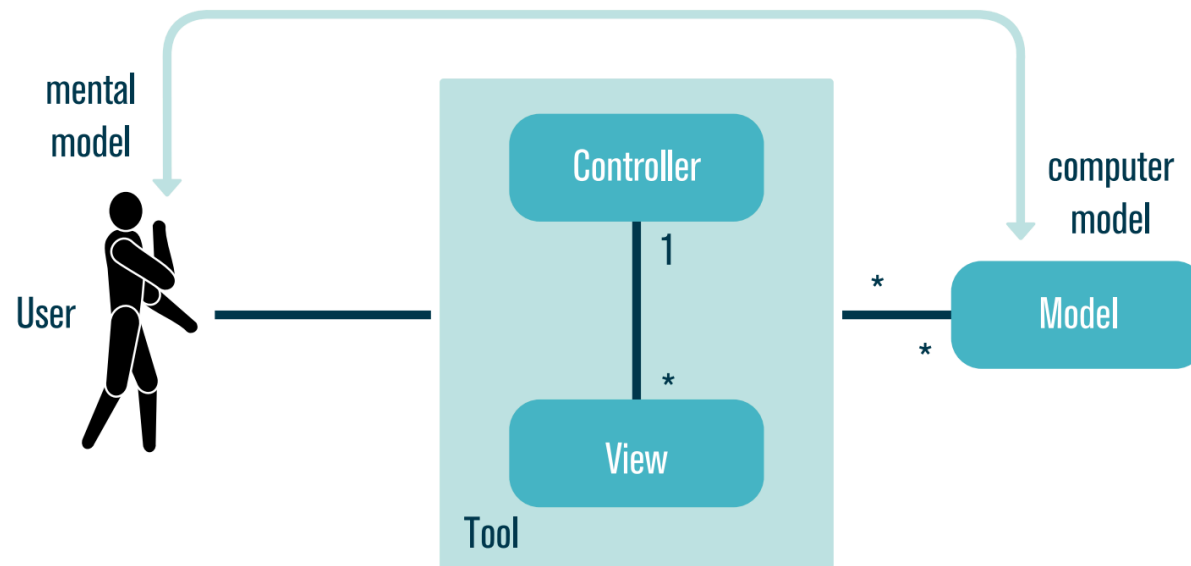
Modelo MVC



Modelo MVC

- ▶ Criada pelo professor Trygve Reenskaug, da Universidade de Oslo, em 1978, no famoso Centro de Pesquisas de Palo Alto da Xerox (PARC);
- ▶ A abordagem MVC (Modelo/Visão/Controlador) define a lógica da implementação em camadas;
- ▶ Composta por três tipos de objetos: Modelo, Visão e Controlador.

Modelo MVC



Representação da arquitetura MVC (REENSKAUG, 1979).

Modelo MVC - Visão (View)

- ▶ Constituída por artefatos de software, que proporcionam aos usuários a visualização dos dados e a apresentação dos elementos de interação que serão imputados por esses usuários para processamento.
- ▶ Exemplos de camadas de apresentação:
 - ▶ Sistema de menus baseados em texto;
 - ▶ Página escrita em HTML ou XHTML com JavaScript apresentada em
 - ▶ Navegador de Internet;
 - ▶ Interface gráfica construída em algum ambiente de programação (PyGTK, Swing do Java, Delphi etc.).

Modelo MVC - Controlador (Controller)

- ▶ Composta de classes que implementam as regras do negócio no qual o sistema está para ser implantado.
- ▶ Nessa camada, são efetuados os processamentos com base nos dados armazenados ou nos dados de entrada, provenientes da camada View.
- ▶ As validações de dados podem ser efetuadas, antes do seu processamento principal.
- ▶ Como exemplo de implementações, as classes baseadas no padrão DAO (Data Access Model), do catálogo Core J2EE, que contêm as operações de manipulação do SGBD, conhecidas como CRUD.

Modelo MVC - Modelo (Model)

- ▶ Contém classes que se comunicam com outros sistemas para realizar tarefas ou adquirir informações;
- ▶ Esta camada é implementada utilizando a tecnologia de banco de dados, em que um SGBD executa em um ou mais nós de processamento de alto desempenho;
- ▶ Como exemplo de implementações temos as Stored Procedures presentes nos SGBDs (MySQL, Oracle SQL Server etc).

Modelo MVC - Vantagens

- ▶ Permite acoplamento baixo entre as camadas, maior grau de manutenção e reutilização de objetos;
- ▶ Facilidade em novas implementações (ex.: incluir uma camada de apresentação para utilização de ambiente web);
- ▶ Mais adaptáveis a uma quantidade maior de usuários. Pode-se dividir a carga de processamento do sistema entre as camadas de lógica da aplicação e acesso;
- ▶ Quando bem projetada, oferece uma arquitetura independente de SGBD, facilitando assim uma eventual migração de base de dados.

Modelo MVC - Desvantagens

- ▶ Diminuir potencialmente o desempenho: a cada camada, as representações dos objetos sofrem modificações, e essas modificações levam tempo para serem realizadas;
- ▶ Dificuldade de implementação, na definição de quais classes irão compor cada uma das camadas, e como comunicá-las entre si.

Componentes



Componentes em Java

"Componentes são unidades de software autocontidas e reusáveis que podem ser compostas visualmente em componentes compostos, applets, aplicações e servlets, usando ferramentas visuais de construção de aplicações." (Sun,2006)



JavaBeans

- ▶ Características:
 - ▶ Construtor sem argumentos;
 - ▶ Propriedades;
 - ▶ Introspecção;
 - ▶ Customização;
 - ▶ Persistência;
 - ▶ Eventos.

JavaBeans

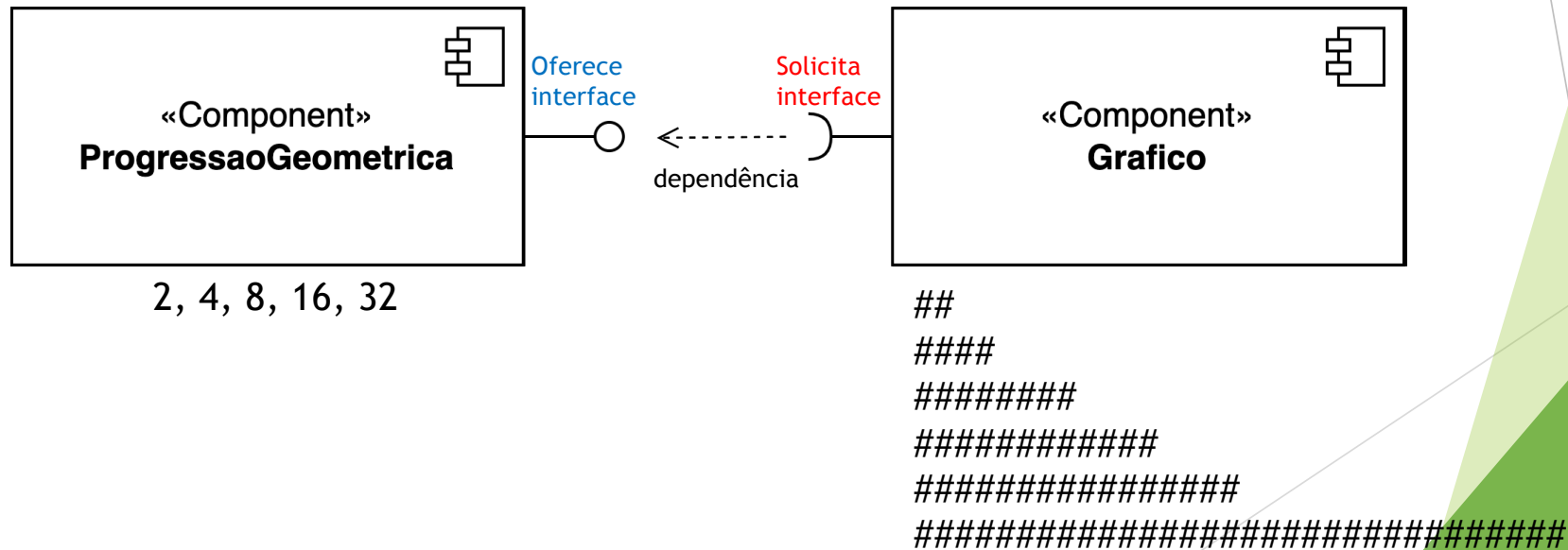
- ▶ Componentes são associados a classes
 - ▶ São instanciados como objetos.
- ▶ Propriedades externamente observáveis
 - ▶ Customiza a instância do componente.



Plotando um gráfico de
progressão geométrica



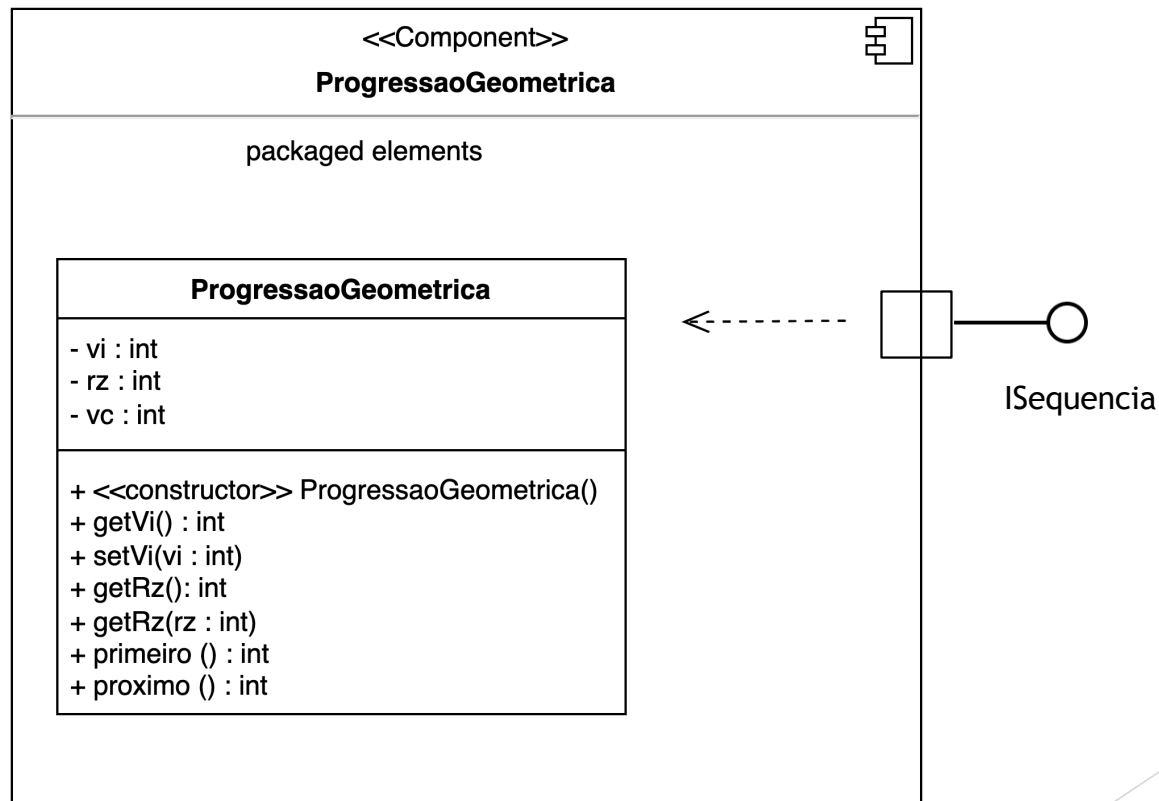
Plotando um gráfico de progressão geométrica



Componente Progressão Geométrica - Black Box



Componente Progressão Geométrica - White Box



Classe que implementa o
componente



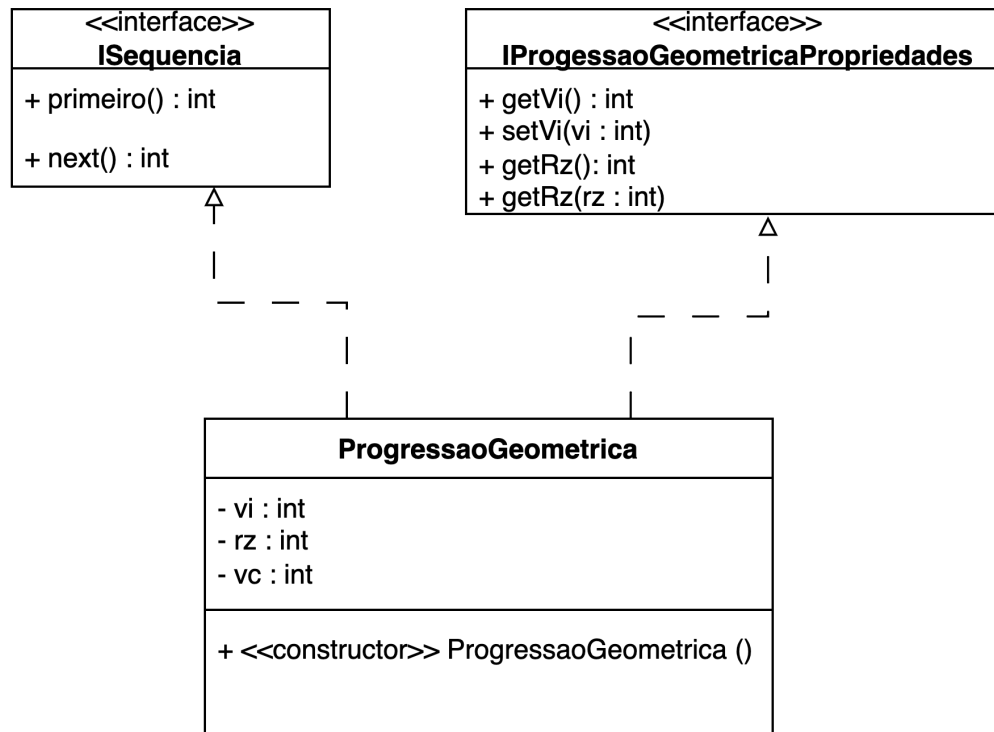
Classe que implementa o componente

ProgressaoGeometrica
- vi : int - rz : int - vc : int
+ <<constructor>> ProgressaoGeometrica() + getVi() : int + setVi(vi : int) + getRz(): int + getRz(rz : int) + primeiro () : int + proximo () : int

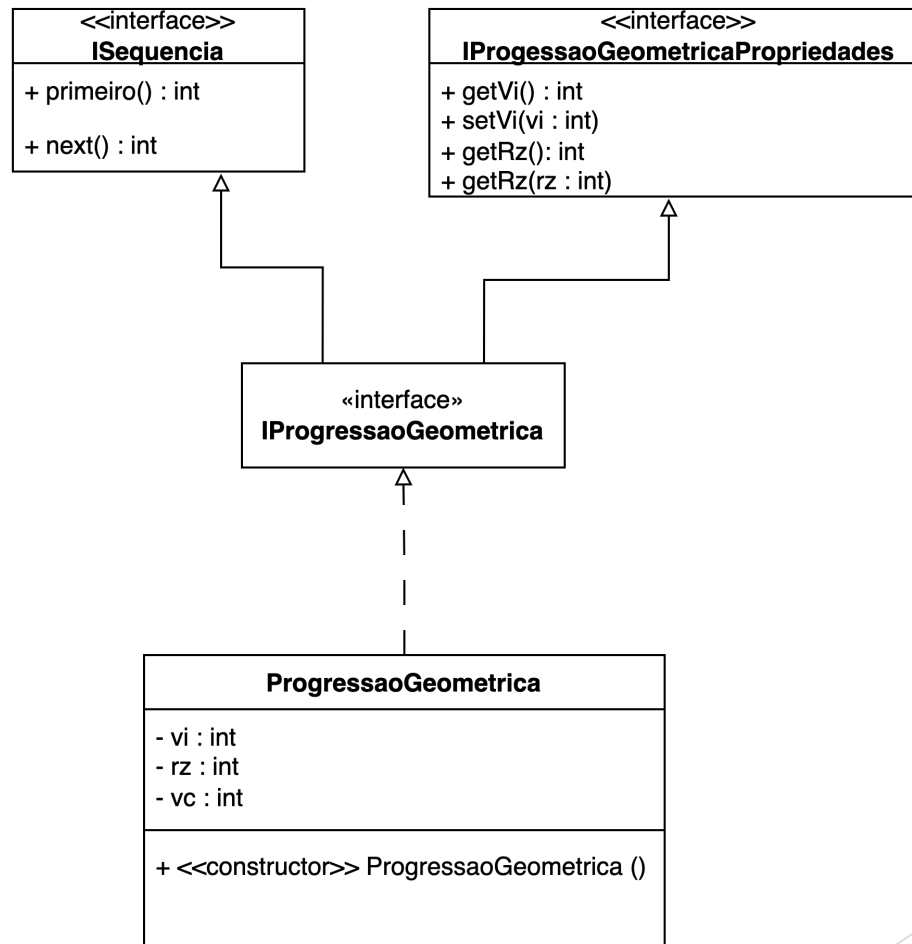
Acesso via interface



Acesso via interface



Acesso via interface








Montando um componente em
um pacote



Montando um componente em um pacote

package A18Componentes.graficoPG.sequencia

- ▼  A18Componentes.graficoPG.sequencia
 - >  IPgressaoGeometrica.java
 - >  IPgressaoGeometricaPropriedades.java
 - >  ISequencia.java
 - >  ProgressaoGeometrica.java

Construtor sem argumentos

- ▶ Permite a criação de componente;
- ▶ Construtor com ação padrão.

```
IProgressaoGeometrica pg = new ProgressaoGeometrica();
```



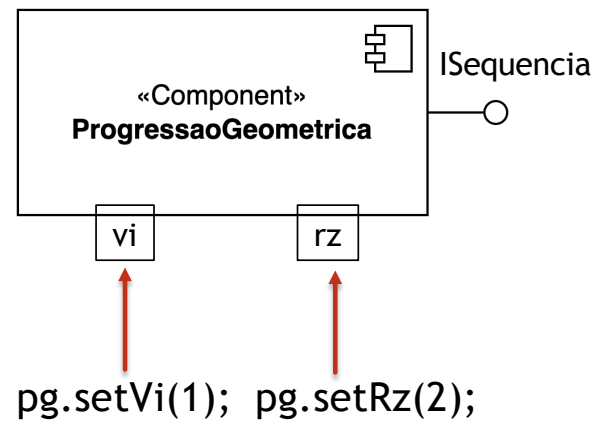
Propriedades



Propriedades em Java

- ▶ Não criar atributos públicos.
- ▶ Gravação:
 - ▶ "get" leitura;
 - ▶ "set" modificação.
- ▶ Somente leitura
 - ▶ Não tem método "set"

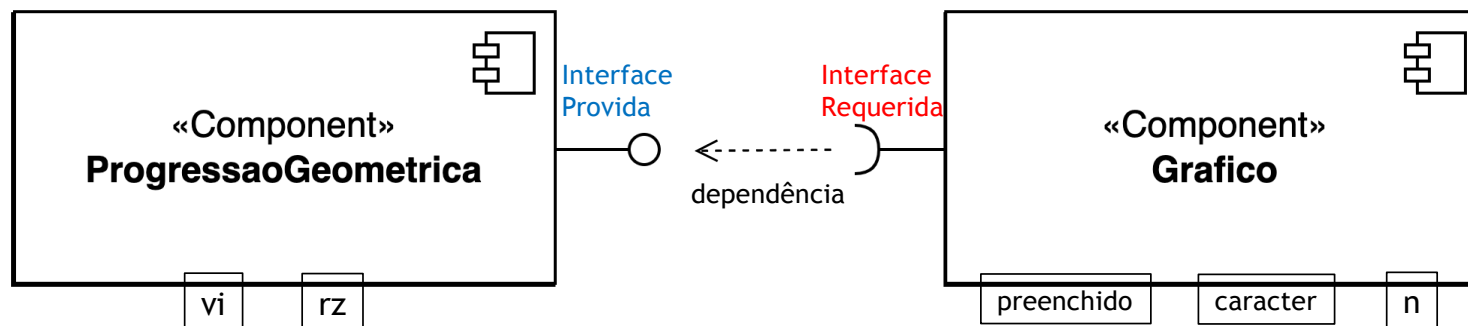
Propriedades em Java



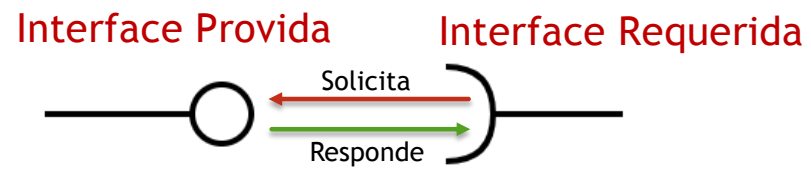
Interface Requerida



Interface Provida e Interface Requerida



Interface Provida e Interface Requerida

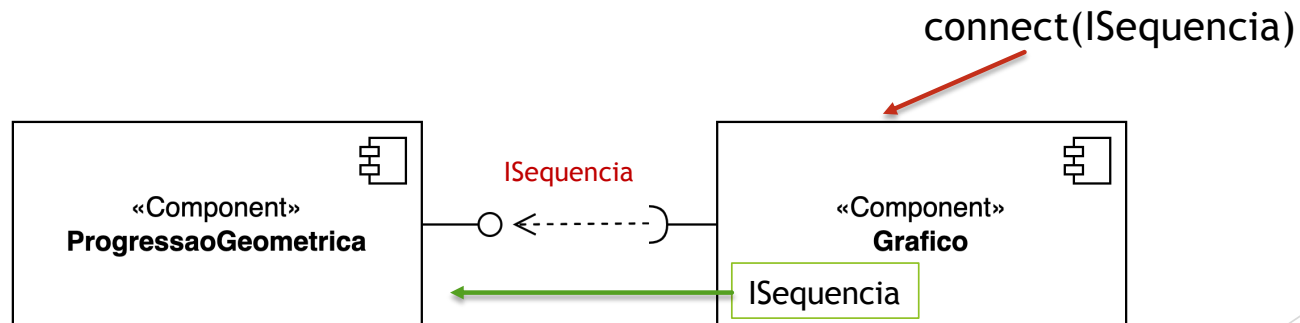


Pattern de Conexão

- ▶ Estabelecida por meio de uma operação **connect**;
- ▶ Componente com Interface Requerida guarda o endereço do componente com uma Interface Provida;
- ▶ O **connect** é uma preparação para uma ação futura.

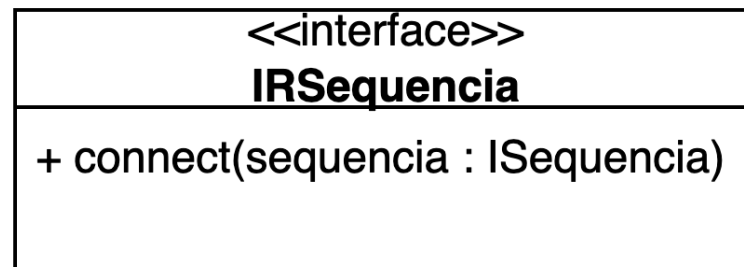
Operação connect

- ▶ Fornecido pelo componente com a Interface Requerida;
- ▶ Por meio do **connect** ele recebe e guarda o endereço da Interface Provida.

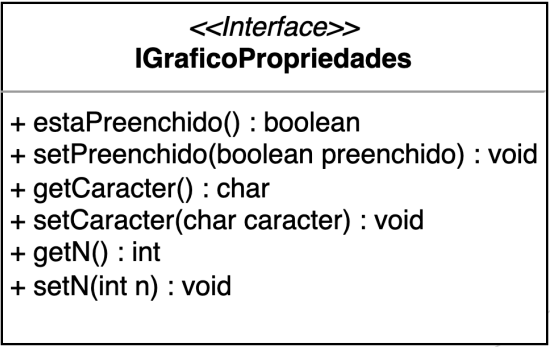
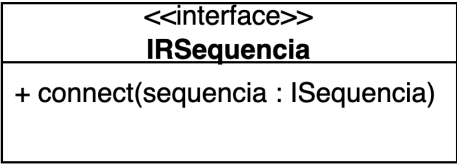
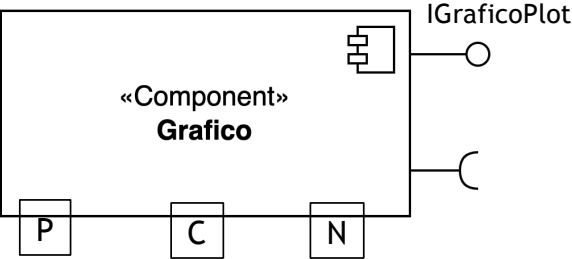


Realizando uma Interface Requerida

- ▶ Em Java não existe conceito de Interface Requerida;
- ▶ Assim, a Interface Requerida deve ser implementada como uma nova interface;
- ▶ Esta nova interface define apenas a operação **connect**;









Componente Grafico

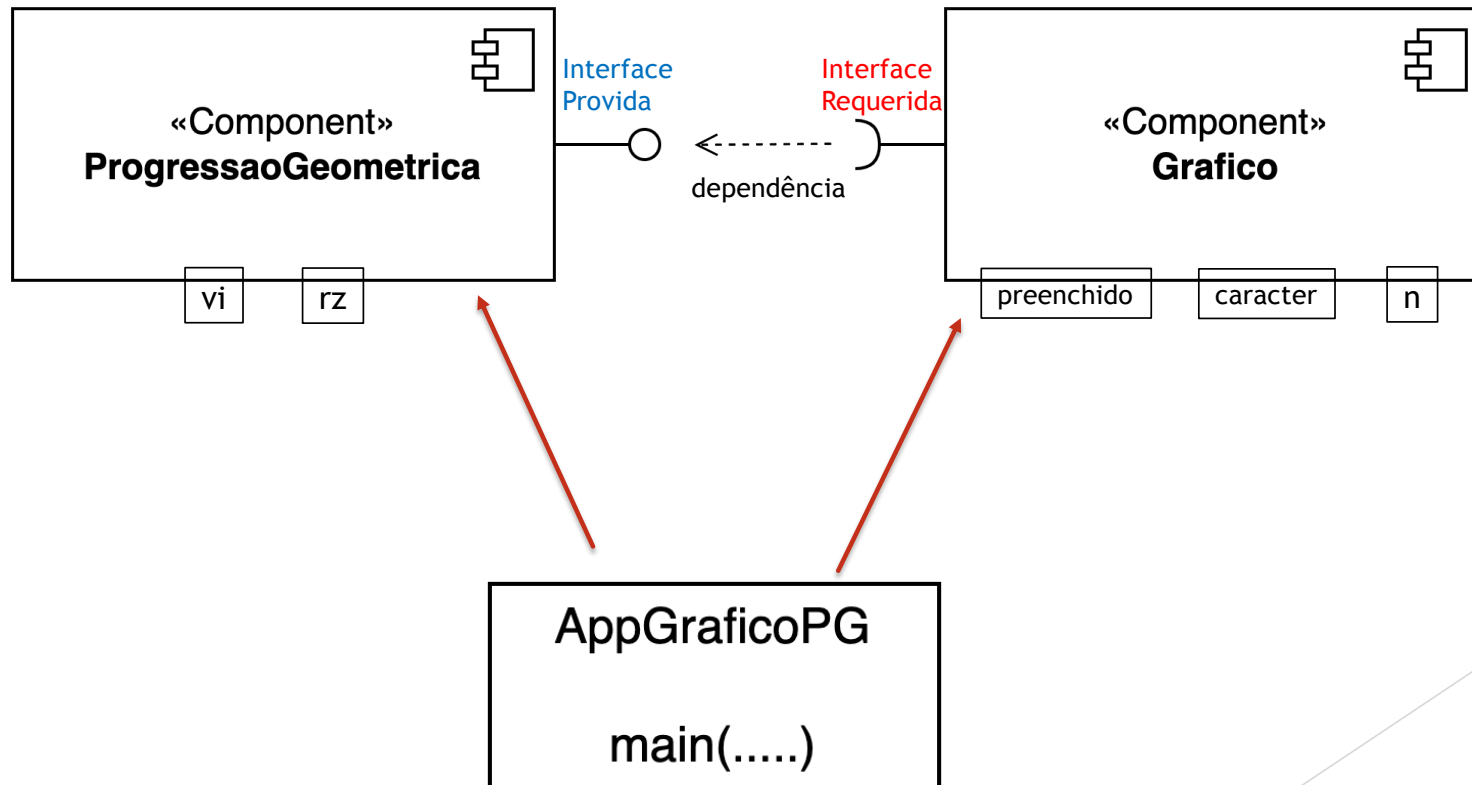


Componente Grafico

```
package A18Componentes.graficoPG.grafico;
```

- ▼  A18Componentes.graficoPG.grafico
 - >  Grafico.java
 - >  IGrafico.java
 - >  IGraficoPlot.java
 - >  IGraficoPropriedades.java
 - >  IRSequencia.java

Conectando Componentes



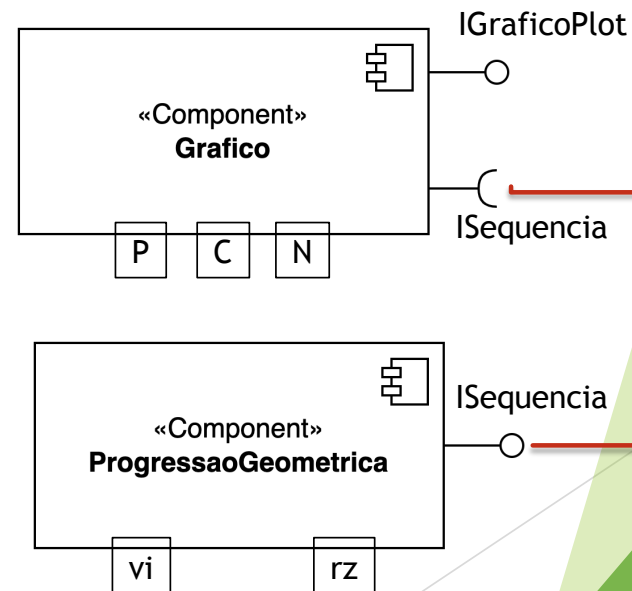
Conectando Componentes

```
IProgressaoGeometrica pg = new ProgressaoGeometrica();  
pg.setVi(1);  
pg.setRz(3);
```

```
IGrafico g = new Grafico();  
g.setPreenchido(true);  
g.setCaracter(*);  
g.setN(5);
```

```
g.connect(pg);
```

```
g.plot();
```



Referências

FURGERI, Sérgio. **Java 8: Ensino Didático**. 1ª ed., São Paulo: Érica, 2015.

DEITEL, Harvey M.; DEITEL, Paul J. **Java: como programar**. 10ª ed. Editora Pearson Education do Brasil, 2017.

JUNIOR, Peter Jandl. **Java Guia do Programador**. 4a Edição: Atualizado para Java 16. Novatec Editora, 2021.