INF319 — Object-Oriented Design and Implementation
Mathematical Progressions
Progressões

Luiz E. Busato

Institute of Computing – UNICAMP
buzato@ic.unicamp.br

Specialization in Software Engineering

Contents

Especificação

Progressão Matemática

- Uma progressão (seqüência) é uma lista ordenada de inteiros não negativos, denominados termos da progressão.
- A determinação de um termo de uma progressão é feita por uma função inteira que mapeia o inteiro que representa a posição do termo na seqüência ao inteiro que corresponde ao termo.
- Freqüentemente, a função que define a progressão é recursiva e pode usar um, dois ou um número arbitrário de termos anteriores em sua definição.

Exemplos

Progressão Aritmética

$$a(i) = \begin{cases} 0 & \text{se } i = 0 \\ a(i-1) + \text{incremento} & \text{se } i \geq 1 \end{cases}$$

Progressão Geométrica

$$g(i) = \begin{cases} 1 & \text{se } i = 0 \\ g(i-1) \times \text{base} & \text{se } i \geq 1 \end{cases}$$

Especificação da Aplicação

Apoio ao Ensino

Uma universidade que utiliza sistemas de apoio ao ensino precisa de um componente de software que calcule a seqüência de termos de progressões matemáticas. Inicialmente, a empresa quer que o componente seja capaz de (re)-iniciar uma progressão, retornar o próximo valor de uma progressão, retornar o i-ésimo valor de uma progressão e imprimir os n primeiros valores de uma progressão. A versão do componente que será testada para aceitação deve calcular as progressões aritmética e geométrica. O componente deve ser capaz de acomodar incrementalmente novas séries matemáticas.

Especificação da Aplicação

Apoio ao Ensino

Após negociação com os desenvolvedores do software, o cliente, no papel de dono do produto, concordou que a primeira versão operacional da aplicação deveria ser capaz de computar as progressões aritmética e geométrica. A segunda versão deveria incluir outras progressões, que serão especificadas oportunamente.

Determinação das Classes

Onde estão os objetos?

- Progressão aritmética (ProgressaoAritmetica)
- Progressão geométrica (ProgressaoGeometrica)
- Valor corrente da progressão
- (Re)-iniciar
- Retornar o próximo valor
- Retornar o i-ésimo valor
- Imprimir n primeiros valores

Detalhamento dos Objetos

Atributos e Métodos: ProgressaoAritmetica

Atributos: valor corrente

Métodos: inicia(), proxTermo(), iesimoTermo(), imprimeProgressao()

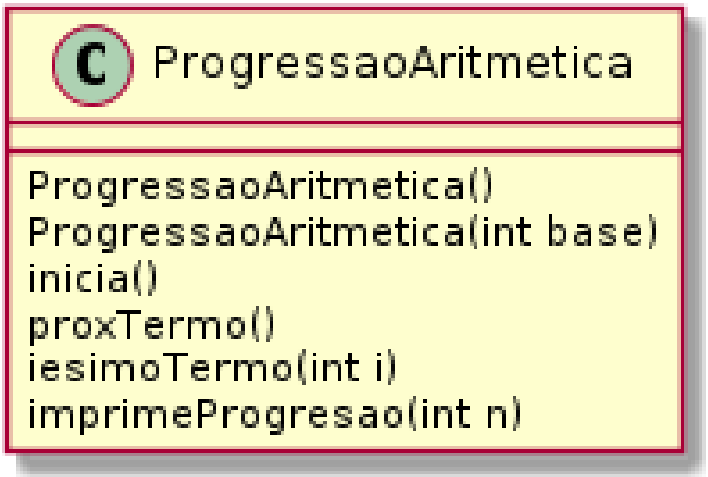
Atributos e Métodos: ProgressaoGeometrica

Atributos: valor corrente

Métodos: inicia(), proxTermo(), iesimoTermo(), imprimeProgressao()

Métodos são iguais! Definimos uma interface comum para progressões na análise.

Diagrama de Classes



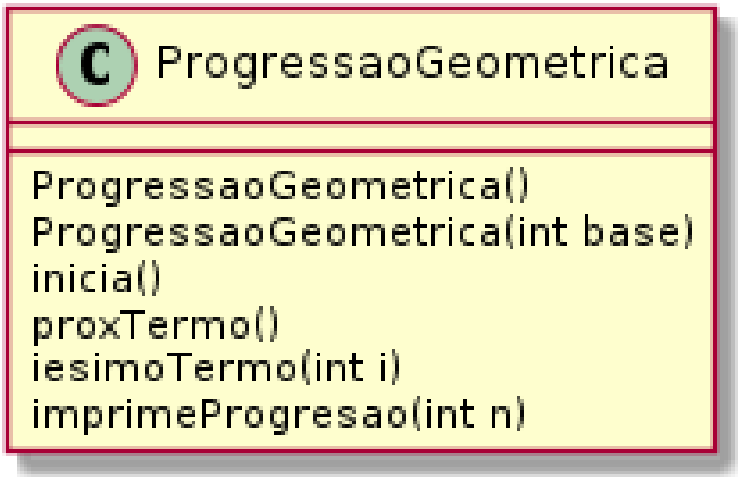
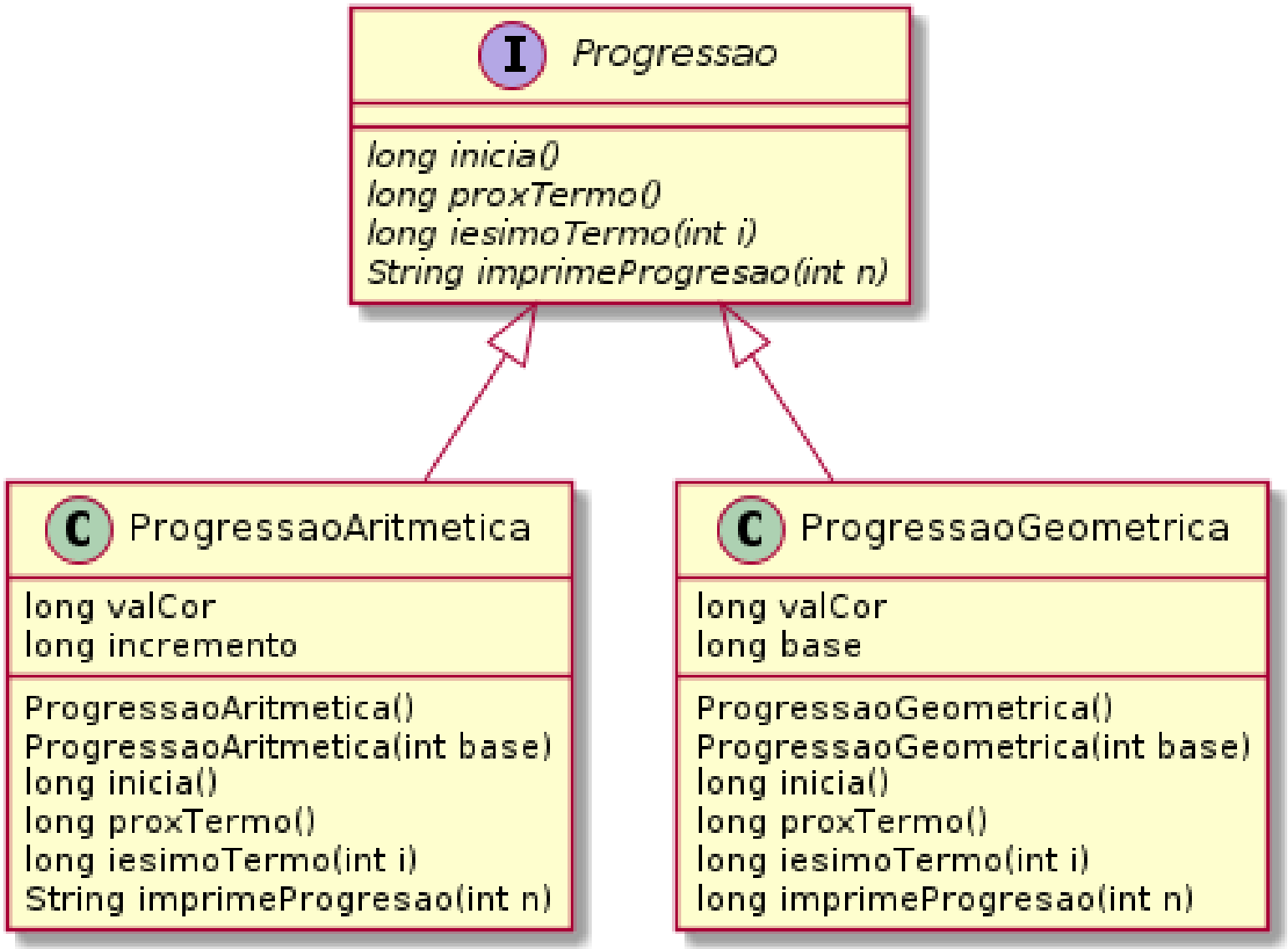


Diagrama de Classes



Atividades

Estratégia

- Devido a simplicidade do problema, o modelo de classes obtido na análise parece ser apropriado.
- Detalharemos o modelo com os atributos necessários e realizaremos uma primeira implementação.
- ProgressaoAritmetica e ProgressaoGeometrica serão testadas para validar o seu comportamento contra a especificação.

Revisão da Análise

Herança

- Uma análise comparativa das implementações de progressão aritmética e geométrica nos permite avaliar o uso de herança.
- Existem muitas semelhanças na implementação destas classes, o que indica que eles compartilham mais do que uma interface.

Análise ou Projeto?

As classes advindas da análise já incluiam herança da interface, então a "subida" de atributos e métodos das classes mais especializadas é um refino da análise ou do projeto? Ambas as abordagens são corretas, mas neste caso específico prefiro considerar como refino do projeto porque a razão desta mudança está mais associada a reaproveitamento de código do que a uma re-definição do comportamento das classes concretas.

Refatoração de Atributos

```
class ProgressaoAritmetica {
    private int incremento; private int valCor;


class ProgressaoGeometrica {
    private int base; private int valCor;
```

Atributos

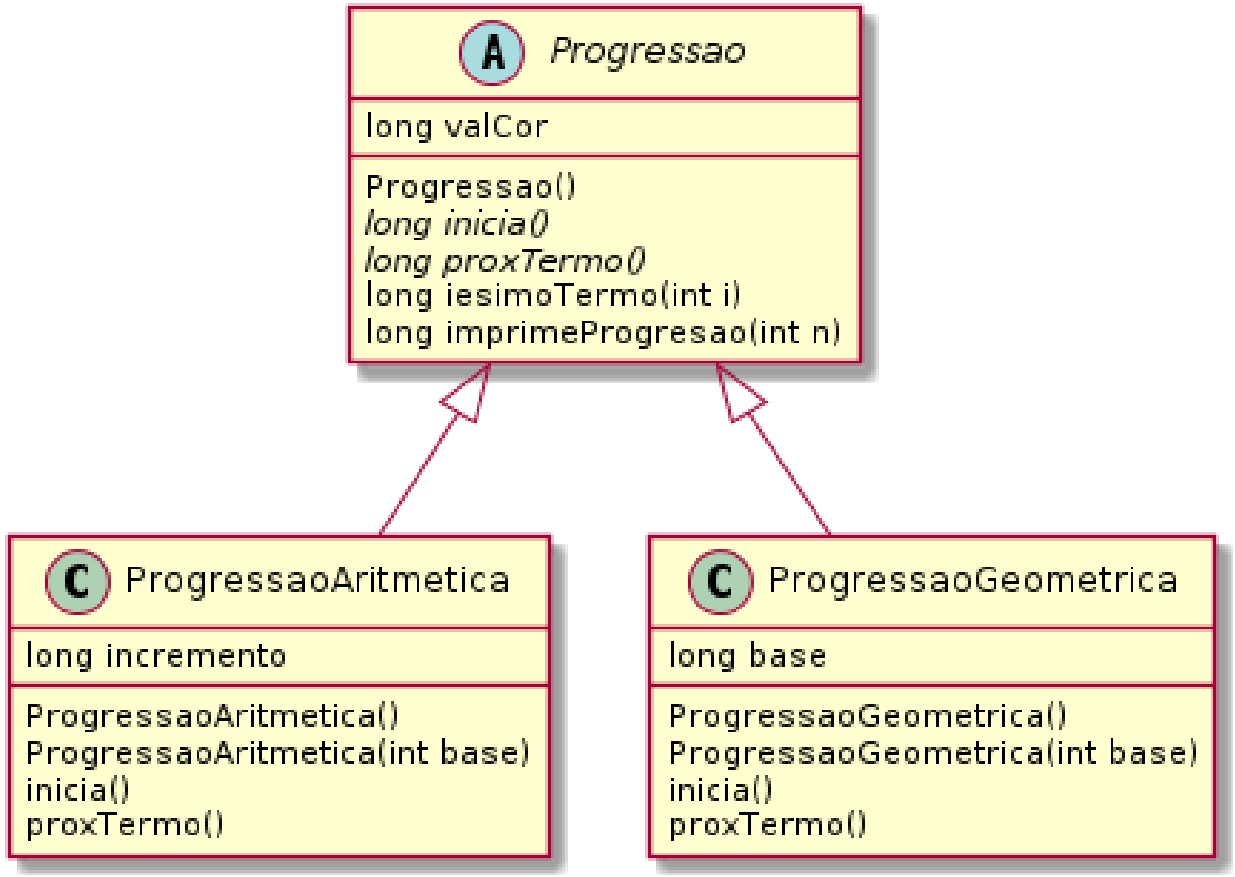
- Os atributos `base` e `incremento` são específicos de cada progressão, portanto, permanecem nas classes `ProgressaoGeometrica` e `ProgressaoAritmetica`.
- O atributo `valCor` é comum e pode ser deslocado para a superclasse `Progressao`.

Refatoração de Métodos

Métodos

- Os construtores e os métodos `inicia()` e `proxTermo()` são específicos de cada progressão, portanto, suas implementações concretas permanecem nas classes `ProgressaoGeometrica` e `ProgressaoAritmetica`. Eles implementam a progressão. A superclasse `Progressao` oferecerá implementações concretas genéricas para esses métodos.
- Observa-se que os métodos `iesimoTermo()` e `imprimeProgressao()` podem ser implementados somente na superclasse `Progressao`. Esses métodos utilizarão as implementações específicas dos métodos polimórficos `inicia()` e `proxTermo()` na sua implementação.

Diagrama de Classes, Detalhado



Projeto revisado: Código e Teste

Implementação

- Como resultado da 2ª iteração, temos um modelo de classes que utiliza herança e polimorfismo para aumentar o reuso.
- Os testes continuam os mesmos da 1ª iteração. Isto permite fazer a reestruturação não só do projeto mas do código com mais segurança.

Mais Uma Progressão

Objetivo

Na 3ª iteração, incluiremos a progressão de Fibonacci no sistema de progressões.

Progressão de Fibonacci

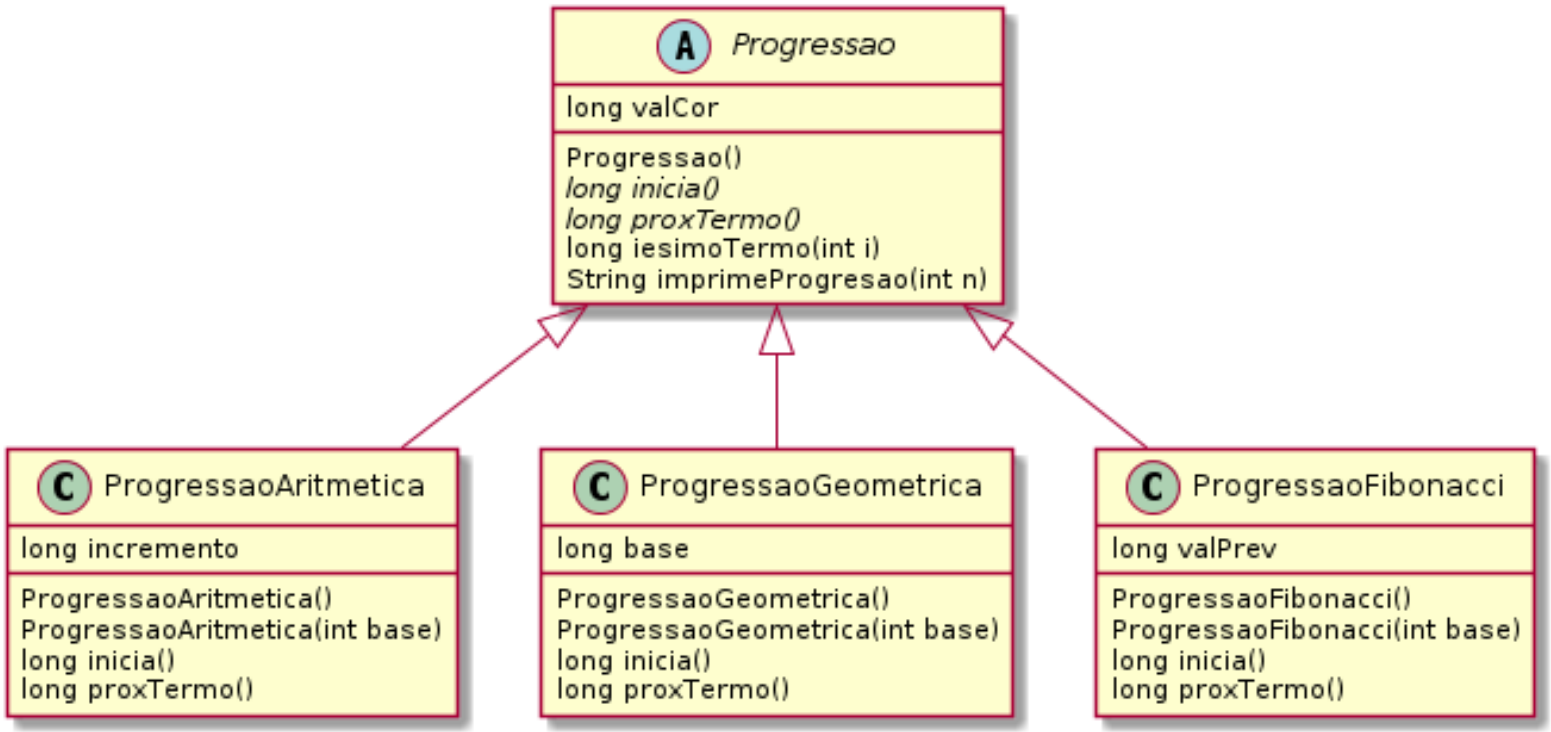
$$f(i) = \begin{cases} 0 & \text{se } i = 0 \\ 1 & \text{se } i = 1 \\ f(i-1) + f(i-2) & \text{se } i > 1 \end{cases}$$

Revisão do Projeto

O Que Muda?

- A análise de Fibonacci mostra que pela primeira vez uma progressão requer a manutenção do valor corrente do termo e de seu valor anterior. Como se adaptará o projeto?
- O projeto se comporta bem, os atributos extras necessários para computar Fibonacci podem ficar restritos à nova classe, sem afetar a implementação geral da classe `Progressao`.

Modelo de Classes



Construtores

Método: `inicia()`

A progressão de Fibonacci não é configurável, ela é sempre a mesma. O construtor simplesmente chama o método `inicia()`, que atribui valores iniciais apropriados para os atributos.

```
ProgressaoFibonacci() {
    inicia();
}

public int inicia() {
    valCor = 0; valPrev = 1;
    return valCor;
}
```

Cálculo da Progressão

Atributo: `valPrev`

O cálculo recursivo da seqüência de Fibonacci requer a manutenção do valor do termo anterior da progressão.

```
public int proxTermo() {
    valCor += valPrev;
    valPrev = valCor - valPrev;
    return valCor;
}
```

Mais Problemas

Aconteceu!

- Aconteceu o que todo projetista teme: o cliente gostou tanto do software entregue que pediu "melhorias".
- O sistema deverá ser modificado para acomodar a progressão de Josephus.

Progressão de Josephus

- É baseada no problema de Josephus, onde n pessoas esperam a execução em um círculo. Dado um passo k, a cada etapa da execução $k-1$ pessoas são poupadas e a k-ésima pessoa é morta. Eventualmente, todas as pessoas serão mortas. `https://www.geogebra.org/m/ExvvrBbR`
- A progressão de Josephus é a seqüência das posições onde estarão as pessoas a serem mortas.

O Que Muda?

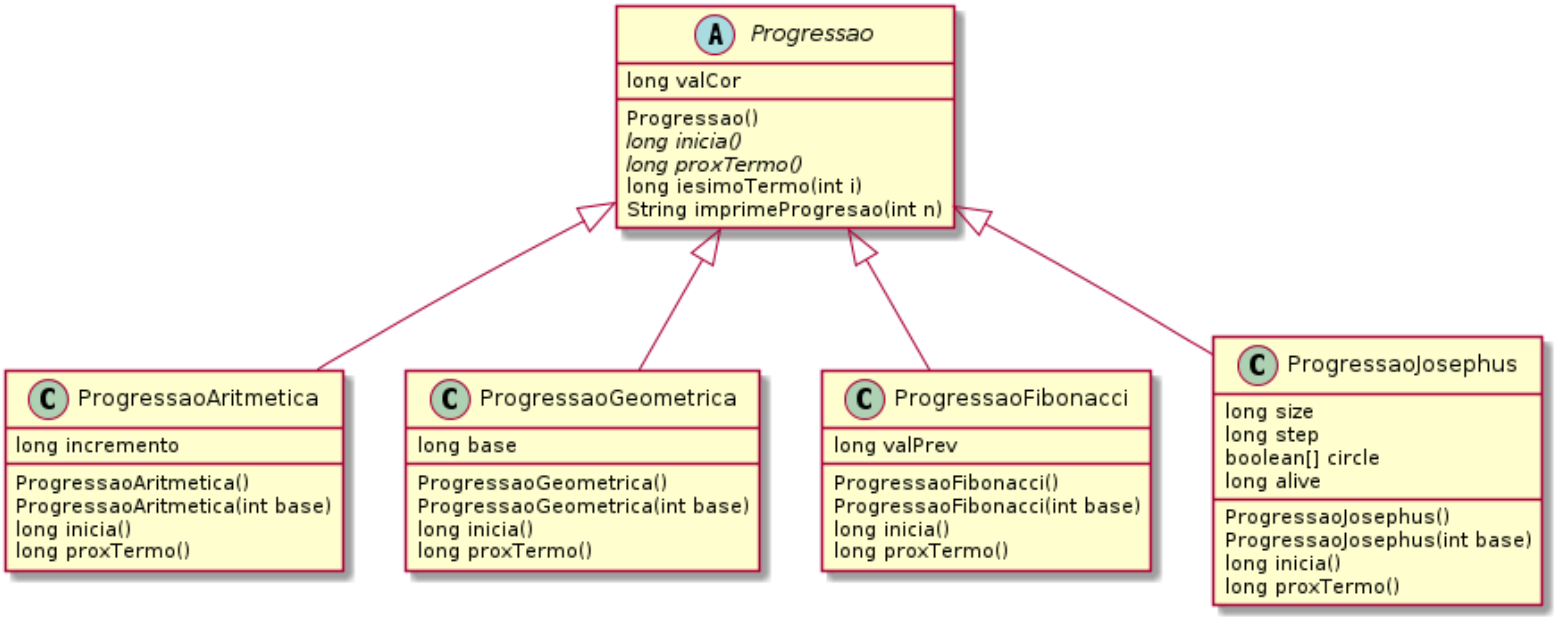
Josephus tem duas características importantes:

1. O número de termos da progressão é finito.
2. Não foi fornecida uma fórmula recursiva ou, melhor ainda, uma fórmula não recursiva. Temos apenas uma regra de como operar sobre um "círculo" de potenciais vítimas.

Adequação do Projeto

- É possível implementar esta progressão sem mudar a forma como as classes estão organizadas?
- Como ficam `inicia()` e `proxTermo()`?

Modelo de Classes



Projeto Refinado

Desempenho

- A progressão depende do valor anterior? Como em Fibonacci?
- Sim, na verdade ela depende de todos os valores dos termos requisitados.

Estrutura de Dados

- No projeto, decidiu-se pela utilização de um vetor auxiliar.
- Custo de "inserção" e "remoção" constantes no tamanho do círculo.
- Busca simplificada (índices fixos).

Progressões

Conclusão

Foi possível criar um componente (Progressões) que até o momento atendeu aos requisitos fundamentais de projeto orientado a objetos: aberto para extensão, fechado para mudanças; classes têm responsabilidade única; Progressoes depende de abstrações, não de implementações; as subclasses de Progressoes se comportam como subtipos (Liskov); a interface de Progressões separa, segrega, e expõe exatamente o comportamento de Progressões, nem mais, nem menos.