

Interface em Java

Exercícios

Vinicius Takeo Friedrich Kuwaki

Universidade do Estado de Santa Catarina

Seções

Exemplo

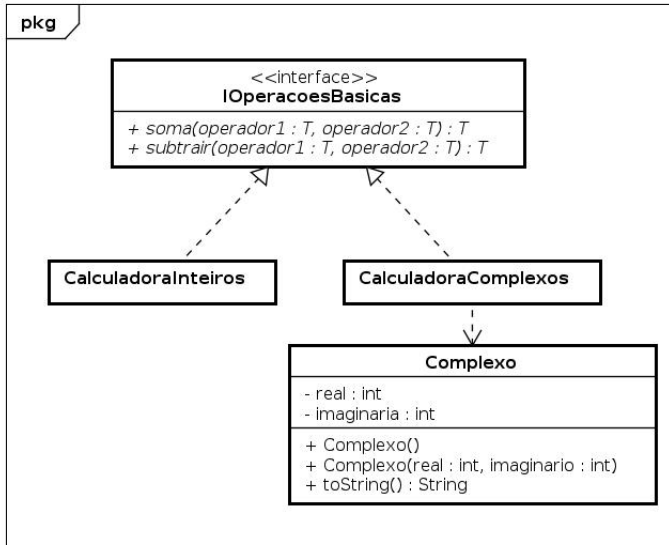
Resolução

Exercício

Exemplo

- A partir do Diagrama de Classes UML que será apresentado a seguir, implemente duas classes que realizam as operações de soma e subtração de números inteiros e números complexos.

Exemplo - Diagrama



Exemplo - Descrição

- Não é necessário implementar um sistema em três camadas, apenas crie uma classe contendo um método **main()** que gere números aleatórios e apresente a soma deles;
- A classe **Complexo** possui dois construtores, o primeiro deles não pede nenhum parâmetro, ele gera dois inteiros aleatórios para representar as partes real e imaginária de um número complexo;
- O segundo construtor pede como parâmetro a parte real e imaginária, e as seta nos atributos da classe;
 - Ex: $(2 + 3i)$;
 - Onde o 2 representa a parte real;
 - E o 3 representa a parte imaginária;
- A interface **IOperacoesBasicas** utiliza um tipo genérico **T**, definido dentro das classes que a realizam;

Seções

Exemplo

Resolução

Exercício

Interface IOperacoesBasicas

- A interface será de um tipo genérico T;
- E possuirá dois métodos:
 - T soma(T operador1, T operador2);
 - T subtracao(T operador1, T operador2);

```
package dados;  
  
public interface IOperacoesBasicas<T> {  
    public T soma(T operador1 , T operador2);  
  
    public T subtracao(T operador1 , T operador2);  
}
```

Classe CalculadoraInteiros

- Agora iremos implementar a primeira realização da interface;
- A classe que realiza a soma e subtração de números inteiros;
- Utilizaremos o Wrapper Integer para substituir nosso tipo genérico T;
- Para realizar uma interface utilizamos a palavra reservada **implements** antes do nome da classe;

```
package dados;

public class CalculadoraInteiros implements IOperacoesBasicas<Integer> {

    @Override
    public Integer soma(Integer operador1, Integer operador2) {
        return operador1 + operador2;
    }

    @Override
    public Integer subtracao(Integer operador1, Integer operador2) {
        return operador1 - operador2;
    }

}
```


Classe Complexo

- Antes de implementarmos a próxima realização da interface, precisamos criar a classe Complexo, para representar os números complexos;
- Todo número complexo possui uma parte real e uma parte imaginária que serão definidos como atributos da classe;

```
package dados;  
  
import java.util.Random;  
  
public class Complexo {  
    private int real;  
    private int imaginaria;  
}
```

Classe Complexo

- A classe terá dois construtores, o primeiro gera um número complexo aleatório e o segundo constrói o número com os valores recebidos como parâmetro;
- Além dos getters para as partes reais e imaginária;

```
public Complexo(int real, int imaginaria) {  
    this.real = real;  
    this.imaginaria = imaginaria;  
}  
  
public Complexo() {  
    Random r = new Random();  
    real = r.nextInt(100);  
    imaginaria = r.nextInt(100);  
}  
  
public int getReal() {  
    return this.real;  
}  
  
public int getImaginaria() {  
    return this.imaginaria;  
}
```

Classe Complexo

- A classe também irá possuir um método **toString()** que retornará uma string contendo o número representado na sua forma algébrica $Z = x + yi$, onde x é a parte real e y a parte imaginária;

```
public String toString() {  
    return "(" + real + " + " + imaginaria + "i" + ")";  
}  
  
}
```

Classe CalculadoraComplexos

- Agora vamos implementar a classe CalculadoraComplexos;
- Seguindo a mesma lógica da calculadora de inteiros, iremos apenas substituir o tipo genérico T para a classe Complexo e alterar o retorno dos métodos;
- Sendo as operações de soma e subtração:
 - $Z_1 = a + bi$;
 - $Z_2 = c + di$;
 - $Z_1 + Z_2 = (a + c) + (b + d)i$;
 - $Z_1 - Z_2 = (a - c) + (b - d)i$;
- Construiremos o número que será retornado diretamente na declaração do retorno do método;
- Já realizando a operação ao mesmo tempo no construtor, utilizando os getters dos objetos recebidos;

Classe CalculadoraComplexos

```
package dados;

public class CalculadoraComplexos implements IOperacoesBasicas<Complexo> {

    @Override
    public Complexo soma(Complexo operador1, Complexo operador2) {
        return new Complexo(operador1.getReal() + operador2.getReal(),
                             operador1.getImaginaria() + operador2.getImaginaria());
    }

    @Override
    public Complexo subtracao(Complexo operador1, Complexo operador2) {
        return new Complexo(operador1.getReal() - operador2.getReal(),
                             operador1.getImaginaria() - operador2.getImaginaria());
    }

}
```

- Os códigos-fontes do exemplo estão disponíveis nesse [link](#);
- Prontamente com um método **main()** para a utilização das classes;

Seções

Exemplo

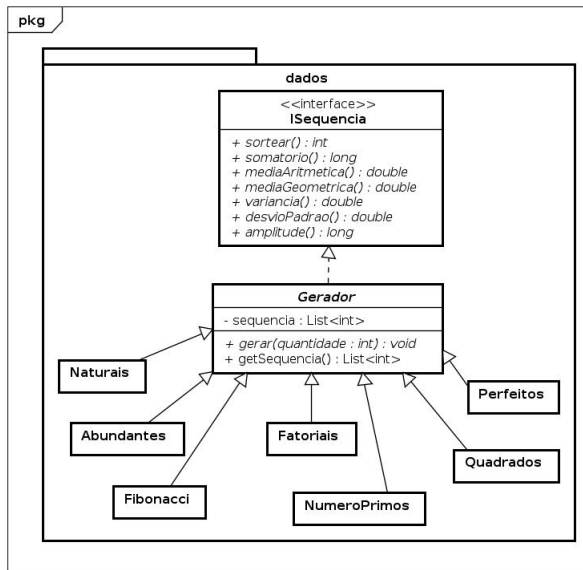
Resolução

Exercício

Exercício

- Continuando o exercício da Aula Prática 6: Classes Abstratas, implemente o Diagrama de Classes UML a seguir;
- Não é necessário implementar uma classe para realizar a interface via console com o usuário e nem uma classe que administre as funcionalidades do diagrama;

Exercício - Diagrama



Exercício - Descrição

- Agora a classe abstrata Gerador deve realizar a interface ISequencia. Portanto, ela precisa implementar todos os métodos definidos na interface;
- Implemente os métodos:

- **sortear()**: o método retorna um termo da sequência selecionado aleatoriamente;
- **somatorio()**: o método retorna o somatório dos termos presentes na sequência;
- **mediaAritmetica()**: o método retorna a média aritmética dos termos presentes na sequência, sendo calculada somando todos os termos da sequência, divididos pela quantidade de termos;
- **mediaGeometrica()**: o método retorna a média geométrica dos termos. Ela é calculada pela raiz n (quantidade de termos) do produtório dos mesmos;

Exercício - Métodos

- **variancia()**: o método retorna o valor de s^2 que é calculado pela seguinte fórmula:


$$s^2 = \frac{\sum_{i=1}^n (X_i - \bar{X})^2}{n - 1}$$

Figura 1: X_i representa o i ésimo termo da sequência, \bar{X} representa a média aritmética e n a quantidade de termos. Sendo \sum o somatório da expressão em parênteses

- **desvioPadrao()**: o método retorna o valor calculado pela $\sqrt{s^2}$ (s^2 é o cálculo da variância);
- **amplitude()**: o método retorna a diferença entre o maior e o menor número da sequência.

Exercício

- Instancie as 7 especializações da classe Gerador e para cada objeto, gere sequências contendo 10 números;
- Utilize os métodos implementados da interface Sequencia e determine para as sequências de 10 números qual delas possui maior:
 1. Somatorio;
 2. Media Aritmética;
 3. Média Geométrica;
 4. Variância;
 5. Desvio Padrão;
 6. Amplitude;

 KUWAKI, V. T. F. Modelo de slides udesc lattex. In: . [S.l.]: Disponível em: <<https://github.com/takeofriedrich/slidesUdescLattex>>. Acesso em: 24 jan. 2020.

Duvidas:
Vinicius Takeo Friedrich Kuwaki
vinicius.kuwaki@edu.udesc.br
github.com/takeofriedrich



UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA