SENAI CIMATEC

SISTEMA FIEB

TECNOLOGIA E INOVAÇÃO EM PROL DA INDÚSTRIA

SENAI CIMATEC

SISTEMA FIEB

Curso Técnico em Informática

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Vetores, Composição e Agregação

Prof: Diego Corrêa

Vetores no JAVA

- O tipo **array** no JAVA é um objeto
- Faz parte do pacote java.util
- São objetos de recipientes que contém um número fixo de valores de um único tipo
- Cada item em um **array** é chamado de elemento, e cada elemento é acessado pelo número, o índice

Exemplo

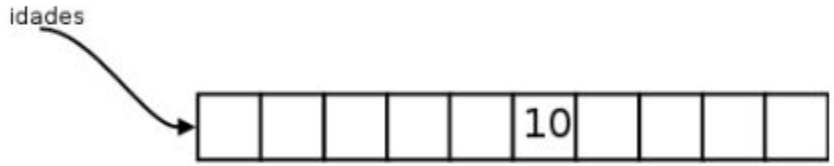
```
int[] idades;
idades = new int[10];
```

- Observe a que é instanciado um objeto idades

- E para atribuir valores para alguma destas posições do vetor?

- E para atribuir valores para alguma destas posições do vetor?

idades[5] = 10



Os índices do array vão de 0 a n-1, onde n é o tamanho dado no momento em que você criou o array.

Populando array

```java
package vetorinteiro;

public class VetorInteiro {
    public static void main(String[] args) {
        int[] valores = new int[10];

        for(int i = 0; i < valores.length; i++){
            valores[i] = i;
        }
        for(int i = 0; i < valores.length; i++){
            System.out.println(valores[i]);
        }
        System.out.println("**********************************************");
        for(int i = 0; i < valores.length; i++){
            valores[i] *= 2;
        }
        for(int i = 0; i < valores.length; i++){
            System.out.println(valores[i]);
        }
    }

}
```

Atividade 1/2

Construa um programa que calcule o **Fibonacci Interativo** e o **Fibonacci Recursivo**

Salvando em um vetor os valores do Fibonnaci para cada posição

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Atividade 1/2

```java
public static int[] interativo(int tamanho) {
    int[] valores = new int[tamanho];
    valores[0] = 0;
    valores[1] = 1;
    for(int i = 2; i < valores.length; i++){
        valores[i] = valores[i-2] + valores[i-1];
    }
    return valores;
}
```

Atividade 1/2

```java
public static int recursivo(int n) {
    if (n <= 1) return n;
    return Fibonnaci.recursivo(n-1) + Fibonnaci.recursivo(n-2);
}

public static void imprimirVetor(int[] valores){
    for(int i = 0; i < valores.length; i++){
        System.out.println(valores[i]);
    }
}
```

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Atividade 1/2

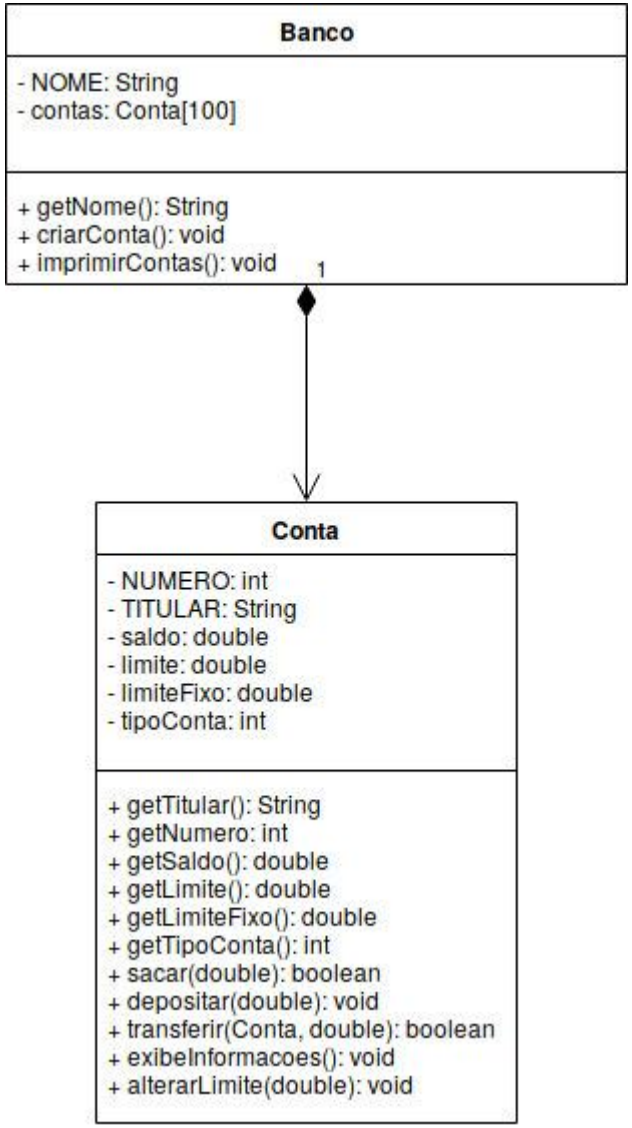
```
public static void main(String[] args) {
    // TODO code application logic here
    int tamanho = 10;

    int[] valores = Fibonnaci.interativo(tamanho);
    Fibonnaci.imprimirVetor(valores);
    System.out.println("-----------------------------------");
    System.out.println(Fibonnaci.recursivo(tamanho-1));
}
```

Composição

Composição

Uma composição acontece quando uma **classe A** tem instanciada dentro dela um objeto (ou lista de objetos) de uma outra **classe B**, quando a **classe A** for destruída, os objetos da **classe B** instanciados dentro da **classe A** também serão destruídos

UML


Banco
- NOME: String
- contas: Conta[100]
+ getNome(): String
+ criarConta(): void
+ imprimirContas(): void
1
Conta
- NUMERO: int
- TITULAR: String
- saldo: double
- limite: double
- limiteFixo: double
- tipoConta: int
+ getTitular(): String
+ getNumero: int
+ getSaldo(): double
+ getLimite(): double
+ getLimiteFixo(): double
+ getTipoConta(): int
+ sacar(double): boolean
+ depositar(double): void
+ transferir(Conta, double): boolean
+ exibeInformacoes(): void
+ alterarLimite(double): void

Código

```java
public class Banco {
    private String NOME;
    private Conta[] contas;
    private static int qtdeContas = 0;

    Banco(String nome, int maxQtdeContas) {
        this.NOME = nome;
        this.contas = new Conta[maxQtdeContas];
    }

    public Conta criarConta(Conta nova_conta) {
        this.contas[Banco.qtdeContas] = nova_conta;
        Banco.qtdeContas++;
        return nova_conta;
    }

    public void imprimirContas() {
        for(int i = 0; i < Banco.qtdeContas; i++) {
            this.contas[i].exibeInformacoes();
        }
    }

}
```

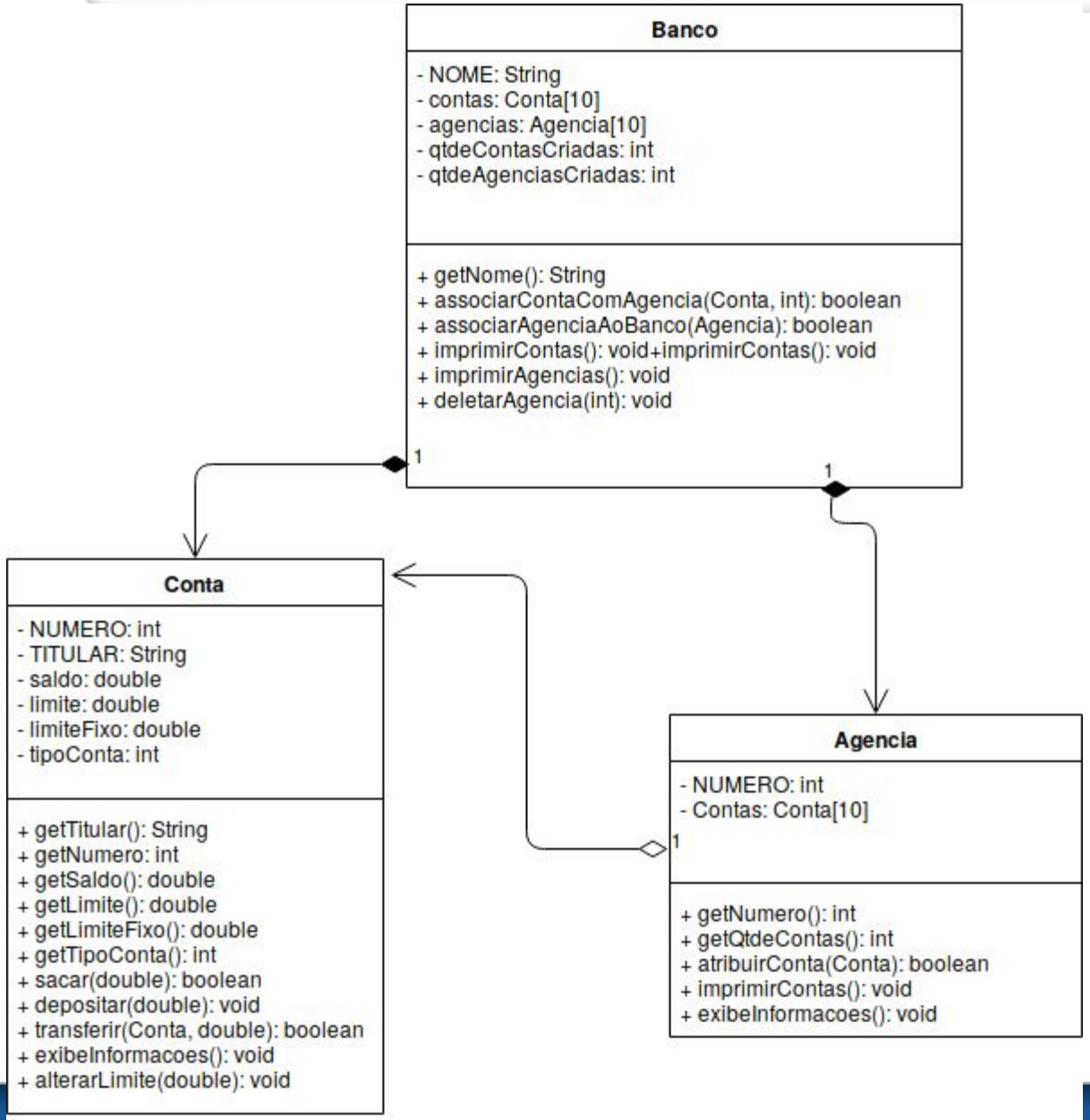
SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Agregação

Agregação

Uma Agregação acontece quando uma **classe A** agrega dentro dela um objeto (ou lista de objetos) de uma outra **classe B**, quando a **classe A** for destruída, o objeto da **classe B** permanece intacto, ou seja, os objetos contidos podem existir sem serem parte do objeto que os contém

UML


Banco
- NOME: String
- contas: Conta[10]
- agencias: Agencia[10]
- qtdeContasCriadas: int
- qtdeAgenciasCriadas: int
+ getNome(): String
+ associarContaComAgencia(Conta, int): boolean
+ associarAgenciaAoBanco(Agencia): boolean
+ imprimirContas(): void+imprimirContas(): void
+ imprimirAgencias(): void
+ deletarAgencia(int): void
1
1
Conta
- NUMERO: int
- TITULAR: String
- saldo: double
- limite: double
- limiteFixo: double
- tipoConta: int
+ getTitular(): String
+ getNumero: int
+ getSaldo(): double
+ getLimite(): double
+ getLimiteFixo(): double
+ getTipoConta(): int
+ sacar(double): boolean
+ depositar(double): void
+ transferir(Conta, double): boolean
+ exibeInformacoes(): void
+ alterarLimite(double): void
Agencia
- NUMERO: int
- Contas: Conta[10]
1
+ getNumero(): int
+ getQtdeContas(): int
+ atribuirConta(Conta): boolean
+ imprimirContas(): void
+ exibeInformacoes(): void

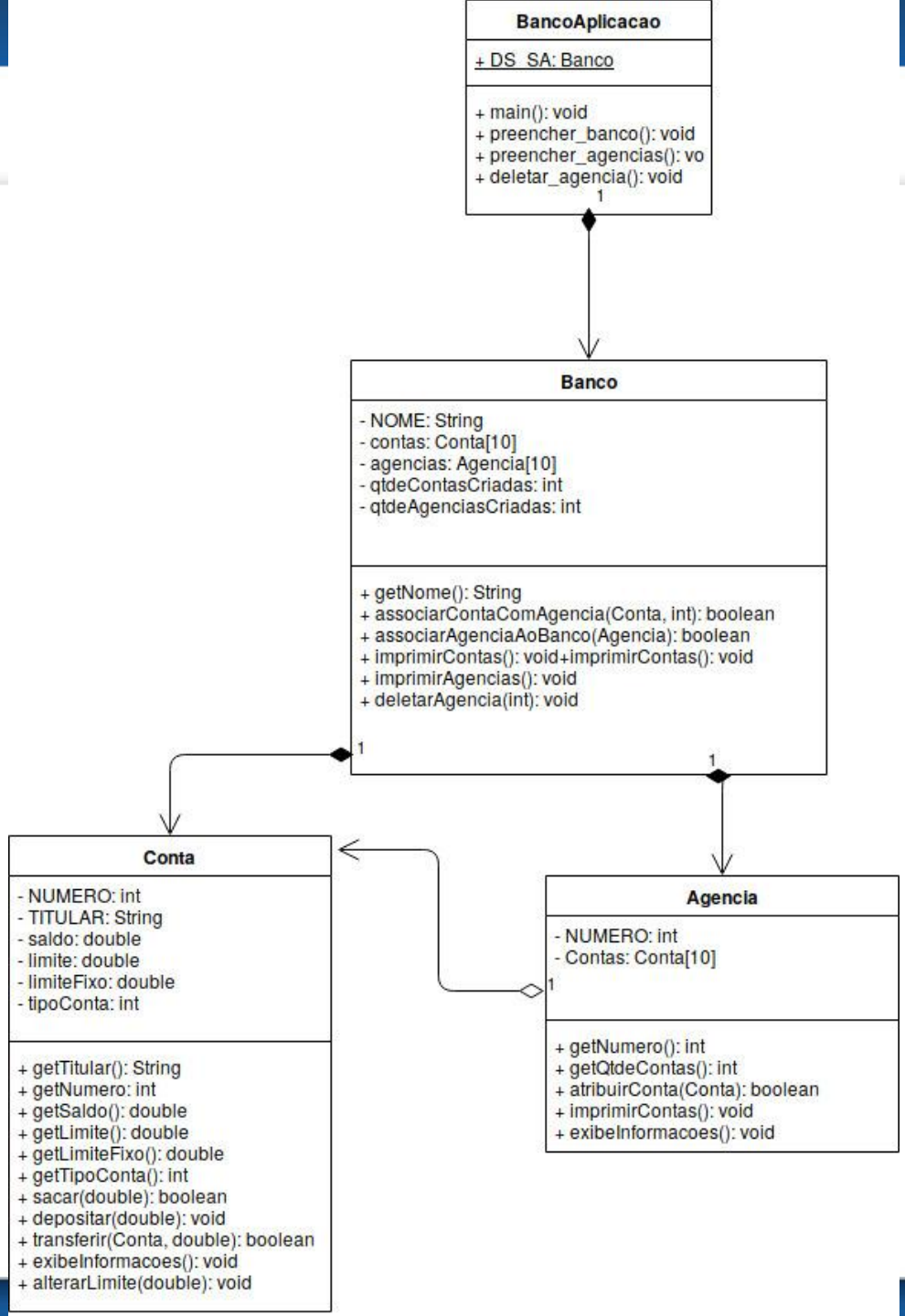
Código

```
public Boolean associarContaComAgencia(Conta nova_conta, int numero_agencia) {
    int i;
    for(i = 0; i < this.qtdeAgenciasCriadas && this.agencias[i].getNumero() != numero_agencia; i++){}
    if(this.agencias[i].getNumero() == numero_agencia && this.agencias[i].atribuirConta(nova_conta)) {
        this.contas[this.qtdeContasCriadas] = nova_conta;
        this.qtdeContasCriadas++;
        return true;
    }
    return false;
}
```

Atividade
2/2

Atividade 2/2

- Continuando a implementação do Banco
- Implementar o diagrama a seguir
- Instancie 3 Agências:
  - Uma com uma conta
  - Uma com duas contas
  - Uma **sem** conta
- Ao final delete a agência com duas contas


BancoAplicacao
+ DS_SA: Banco
+ main(): void
+ preencher_banco(): void
+ preencher_agencias(): vo
+ deletar_agencia(): void
1
Banco
- NOME: String
- contas: Conta[10]
- agencias: Agencia[10]
- qtdeContasCriadas: int
- qtdeAgenciasCriadas: int
+ getNome(): String
+ associarContaComAgencia(Conta, int): boolean
+ associarAgenciaAoBanco(Agencia): boolean
+ imprimirContas(): void+imprimirContas(): void
+ imprimirAgencias(): void
+ deletarAgencia(int): void
1
1
Conta
- NUMERO: int
- TITULAR: String
- saldo: double
- limite: double
- limiteFixo: double
- tipoConta: int
+ getTitular(): String
+ getNumero: int
+ getSaldo(): double
+ getLimite(): double
+ getLimiteFixo(): double
+ getTipoConta(): int
+ sacar(double): boolean
+ depositar(double): void
+ transferir(Conta, double): boolean
+ exibeInformacoes(): void
+ alterarLimite(double): void
Agencia
- NUMERO: int
- Contas: Conta[10]
1
+ getNumero(): int
+ getQtdeContas(): int
+ atribuirConta(Conta): boolean
+ imprimirContas(): void
+ exibeInformacoes(): void