Análise e Projeto de Desenvolvimento de Sistemas

APDS

Aula 3

Universidade Federal do Rio Grande do Norte
Unidade Acadêmica Especializada em Ciências Agrárias
Escola Agrícola de Jundiaí
Tecnologia em Análise e Desenvolvimento de Sistemas
Profa. Alessandra Mendes

Modificadores de Acesso

Profa. Alessandra Mendes – UFRN/EAJ/TI/APDS

Modificadores de Acesso

- Os modificadores de acesso controlam a forma como os membros de uma classe são visíveis por outras classes e/ou instâncias de outras classes.
- Um membro de uma classe pode ter um ou nenhum modificador de acesso na sua declaração.
  - Quando não é especificado um dos três modificadores de acesso, dizemos que o membro tem acesso *friendly*.

Public

- O mais abrangente de todos os tipos de acesso, o modificador *public* declara que elementos com esse modificador são acessíveis de qualquer classe Java;
- Este modificador é aplicável a variáveis, métodos, construtores e classes;
  - Quem tem acesso à classe acessa também qualquer membro *public*. Assim, uma declaração com este modificador pode ser acessada de qualquer lugar e por qualquer entidade que possa visualizar a classe à qual ela pertença.

```
public void setName(String name){ //método público
       this.name = name; // armazena o nome
}
```

Public

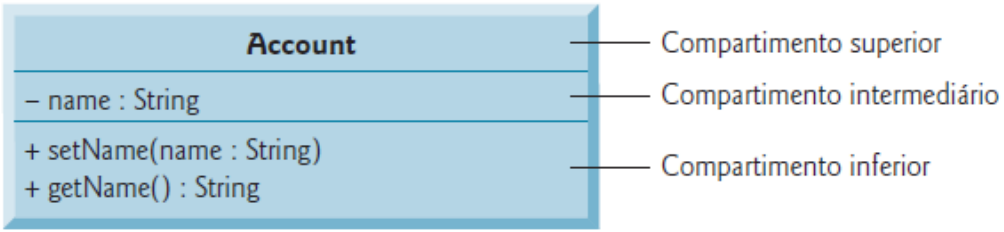
- O principal objetivo dos métodos *public* é apresentar aos clientes da classe uma visualização dos serviços que a classe fornece (isto é, a interface *public* dela);
- Os clientes não precisam se preocupar com a forma como a classe realiza suas tarefas. Por essa razão, as variáveis *private* e os métodos *private* da classe (isto é, seus detalhes de implementação) não são acessíveis aos clientes.

Private

- O modificador de acesso *private* é o mais restritivo;
- Variáveis, métodos e construtores com esse modificador são visíveis somente dentro da própria classe;
- O modificador *private* pode ser usado em variáveis, métodos e construtores;
- A declaração de variáveis de instância com o modificador de acesso *private* é conhecida como ocultamento de dados ou ocultamento de informações.

Private

- Exemplo: Quando um programa cria (instancia) um objeto de classe *Account*, a variável *name* é **encapsulada** (ocultada) no objeto e pode ser acessada apenas por métodos da classe do objeto.




```
public class Account {
	private String name; // variável de instância
	public Account(String name){ // construtor
		this.name = name;    }
	public void setName(String name){
		this.name = name;    }
	public String getName() {
		return name;  }
}
```

Protected

- O modificador de acesso *protected* oferece um nível intermediário de acesso entre *public* e *private*;
- Os membros *protected* de uma superclasse podem ser acessados por membros dessa superclasse, de suas subclasses e de outras classes no mesmo pacote;
- O modificador *protected* torna o membro acessível às classes do mesmo pacote ou através de herança;
- Membros *protected* também têm acesso de pacote.

Acesso de Pacote

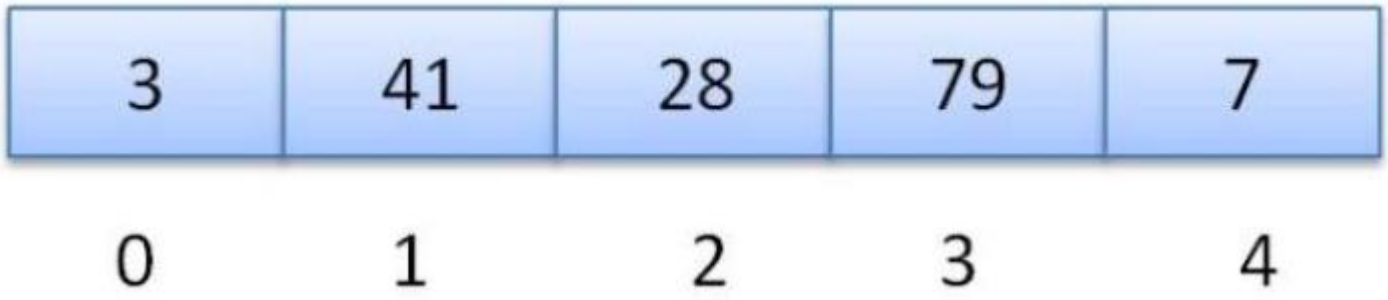
- Se nenhum modificador de acesso (*public*, *protected* ou *private*) for especificado para um método ou variável quando esse método ou variável é declarado em uma classe, o método ou variável será considerado como tendo acesso de pacote.
  - Outros nomes também usados para designar esse tipo de acesso são: "*friendly*", "*package*" e "*default*".
- Os membros com acesso de pacote somente podem ser acessados por classes do mesmo pacote;
- O acesso de pacote é raramente usado.

Arrays

Profa. Alessandra Mendes – UFRN/EAJ/TI/APDS

Array (vetores)

- Fazem parte do pacote java.util;
- São objetos de recipientes que guardam valores do mesmo tipo e tem um comprimento fixo (estabelecido quando criado).
- Cada item em um array é chamado de elemento e cada elemento é acessado pelo índice.
  - Exemplo – array de 5 elementos:



Declaração de arrays

- Na declaração de um array, cada elemento recebe um valor padrão: 0 (zero) para números de tipo primitivo, falso (false) para elementos booleanos e nulo (null) para referências.

```
public class Declaracao_Array {
    public static void main(String[] args) {
        //[] - são inseridos em uma variável que referecia um array
        int[] a = new int[4];
        //OUTRA MANEIRA DE FAZER UMA DECLARAÇÃO DE ARRAY
        int[] b;
        b = new int[10];
        //DECLARANDO VÁRIOS ARRAYS
        int[] r = new int[44], k = new int[23];

        //{} - inicializar valores em um array sua declaração
        int[] iniciaValores = {12,32,54,6,8,89,64,64,6};

        //DECLARA UM ARRAY DE INTEIROS
        int[] meuArray;
```

Continua no próximo slide...

Acessando elementos de um array

... continuando slide anterior.

```
        //DECLARA UM ARRAY DE INTEIROS
        int[] meuArray;
        //ALOCA MEMÓRIA PARA 10 INTEIROS
        meuArray = new int[10];
        //INICIALIZA O PRIMEIRO ELEMENTO
        meuArray [0] = 100;
        meuArray [1] = 85;
        meuArray [2] = 88;
        meuArray [3] = 93;
        meuArray [4] = 123;
        meuArray [5] = 952;
        meuArray [6] = 344;
        meuArray [7] = 233;
        meuArray [8] = 622;
        meuArray [9] = 8522;
        //meuArray [10] = 564; //ESTOURA A PILHA POIS NÃO EXISTE O ÍNDICE 10
        System.out.println(meuArray[9]);
        System.out.println(meuArray[2]);
    }
}
```

Tamanho do array

- Por padrão, cada array sabe seu próprio tamanho, independente de quantos valores forem inseridos.
- O array armazena na variável de instância o método *length*, que retorna o tamanho do array especificado.

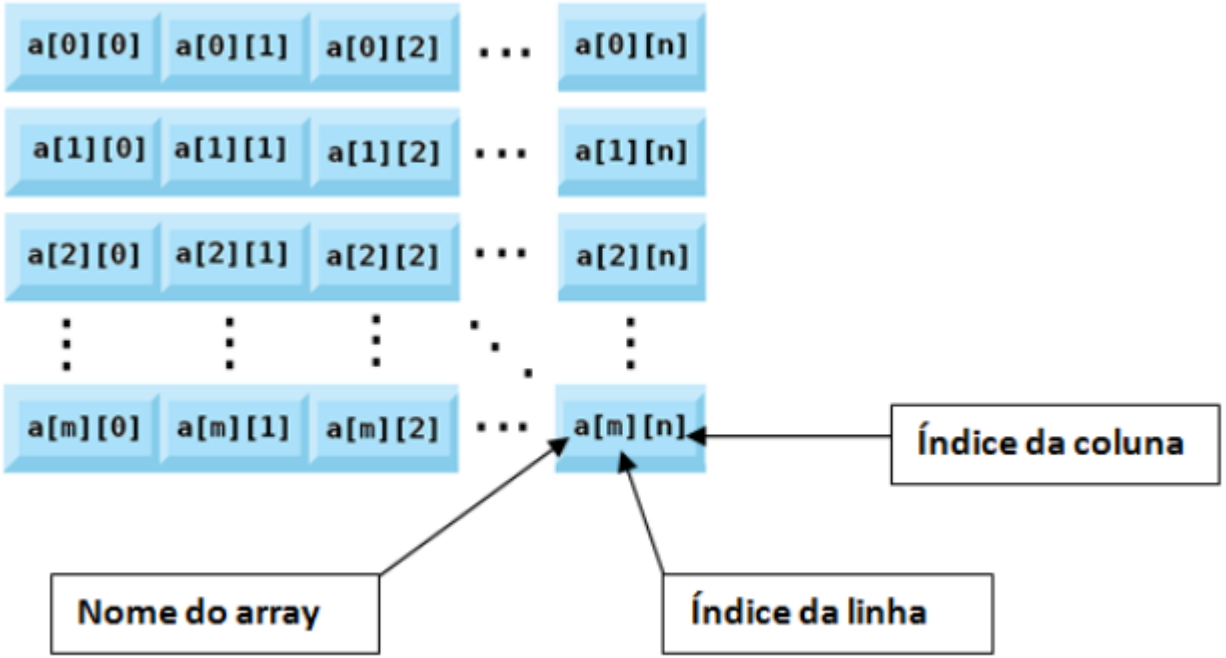
```
public class TamanhoArray {

    public static void main(String[] args) {
        int[] arrayUm = {12,3,5,68,9,6,73,44,456,65,321};
        int[] arrayDois = {43,42,4,8,55,21,2,45};

        if(arrayDois.length > 8){
            System.out.println("Tamanho do ArrayDois - Maior que 8!");
        }else{
            System.out.println("Tamanho do ArrayDois - Menor que 8!");
        }
        System.out.println("\nTamanho do ArrayUm = "+arrayUm.length);
    }
}
```

Arrays bidimensionais (matrizes)

- Os arrays bidimensionais precisam de dois índices para identificar um elemento particular.
- Exemplo de declaração: int [][] a = { { 1, 2 }, { 2, 2 } };



Arrays bidimensionais (matrizes)

- Os arrays bidimensionais precisam de dois índices para identificar um elemento particular.
- Declaração: int [][] a = {{ 1, 2 },{ 2, 2 }}; int[][] a;
- Para percorrer os elementos do array bidimensional são utilizados normalmente dois laços de repetição.

```
public static void outputArray(int[][] array)
{
    //FAZ UM LOOP PELAS LINHAS DO ARRAY
    for(int linha = 0; linha < array.length; linha++)
    {
        //FAZ LOOP PELAS COLUNAS DA LINHA ATUAL
        for( int coluna = 0; coluna < array[linha].length; coluna++)
            System.out.printf("%d ", array[linha][coluna]);
        System.out.println();
    }
}
```

ArrayLists

Profa. Alessandra Mendes – UFRN/EAJ/TI/APDS

A classe ArrayList

- O Java, por padrão, possui uma série de recursos prontos (APIs) para que possamos tratar de estrutura de dados, também chamados de coleções (collections).
- Uma coleção é uma estrutura de dados (na realidade um objeto) que pode armazenar ou agrupar referências a outros objetos (um contêiner).
- Pode-se dizer que ArrayList é uma classe para coleções.
- Pode-se dizer ainda que o ArrayList é um array sem tamanho pré-definido (array dinâmico).
- As classes e interfaces da estrutura de coleções são membros do pacote java.util.

A classe ArrayList

- Podem ser criados objetos - através de uma classe - e agrupados utilizando um ArrayList;
- No ArrayList podem ser realizadas várias operações, como: adicionar e retirar elementos, ordená-los, procurar por um elemento específico, apagar um elemento específico, limpar o ArrayList, entre outras.

- Importando:
  - *import java.util.ArrayList;*
- Declarando e instanciando:
  - *ArrayList<Classe> nomeDoArrayList = new ArrayList<Classe>();*

A classe ArrayList

- Exemplo:
  - *ArrayList<String> agenda = new ArrayList<String>();*
- Adicionando elementos:
  - agenda.add(“João”);
- Removendo elementos:
  - agenda.remove(“João”);
- Qual o tamanho do ArrayList?
  - agenda.size()
- Buscando elementos do ArrayList:
  - agenda.get(índice);
- Limpando o ArrayList:
  - agenda.clear()

A classe ArrayList

- Exemplo de código – Agenda de contatos

```
public class Pessoa {
    private String nome;
    private String telefone;
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getTelefone() {
        return telefone;
    }
    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }
}
```

Continua no próximo slide...

A classe ArrayList

... continuando...

```
import java.util.Scanner;
import java.util.ArrayList;

/**...4 linhas */
public class Principal {

    /**...3 linhas */
    public static void main(String[] args) {
        // TODO code application logic here

        Scanner ler = new Scanner(System.in);
        int resp, i;
        String nomeBusca;
        boolean achou = false;

        ArrayList<Pessoa> agenda = new ArrayList<Pessoa>(); //declaração
```

A classe ArrayList

... continuando...

```
System.out.println("\nOperação 1: Inserindo contatos...");
do{
    Pessoa p = new Pessoa();
    System.out.println("Nome: ");
    p.setNome(ler.nextLine());
    System.out.println("Telefone: ");
    p.setTelefone(ler.nextLine());
    agenda.add(p); //inserindo o contato no ArrayList
    System.out.println("Deseja inserir outro contato? (1-SIM/2-NÃO) ");
    resp = ler.nextInt();
    ler.nextLine();
}while(resp==1);

System.out.println("\nOperação 2: Buscando um contato pelo nome...");
System.out.println("Digite o nome para a busca: ");
```

A classe ArrayList

... continuando...

```
System.out.println("Digite o nome para a busca: ");
nomeBusca = ler.nextLine();
for(i=0; i<agenda.size(); i++){
    if(agenda.get(i).getNome().equals(nomeBusca)){
        System.out.println("Contato localizado!");
        System.out.println("Telefone: "+agenda.get(i).getTelefone());
        achou = true;
        System.out.println("\nOperação 3: Alterando...");
        System.out.println("Digite o novo telefone: ");
        agenda.get(i).setTelefone(ler.nextLine());
        System.out.println("\nOperação 4: Excluindo...");
        System.out.println("Deseja excluir o contato? (1-SIM/2-NÃO) ");
        if(ler.nextInt()==1)
            agenda.remove(agenda.get(i));
    }
}
if(!achou)
    System.out.println("Contato não localizado!");
```

A classe ArrayList

... final do código.

```
        if(!achou)
            System.out.println("Contato não localizado!");

        System.out.println("\nOperação 5: Exibindo todos os contatos");
        for(i=0; i<agenda.size(); i++){
            System.out.println("Contato "+i);
            System.out.println("Nome: "+agenda.get(i).getNome());
            System.out.println("Telefone: "+agenda.get(i).getTelefone());
            System.out.println("---------------------------");
        }

        System.out.println("Fim de programa");
    }

}
```

Vamos à prática...

Exercício

- Alterar a implementação da questão 2 da Lista 1 utilizando um ArrayList de Alunos.

Dúvidas?