Linguagem de Programação orientada a objetos

Linguagem de Programação II

Lista e coleções

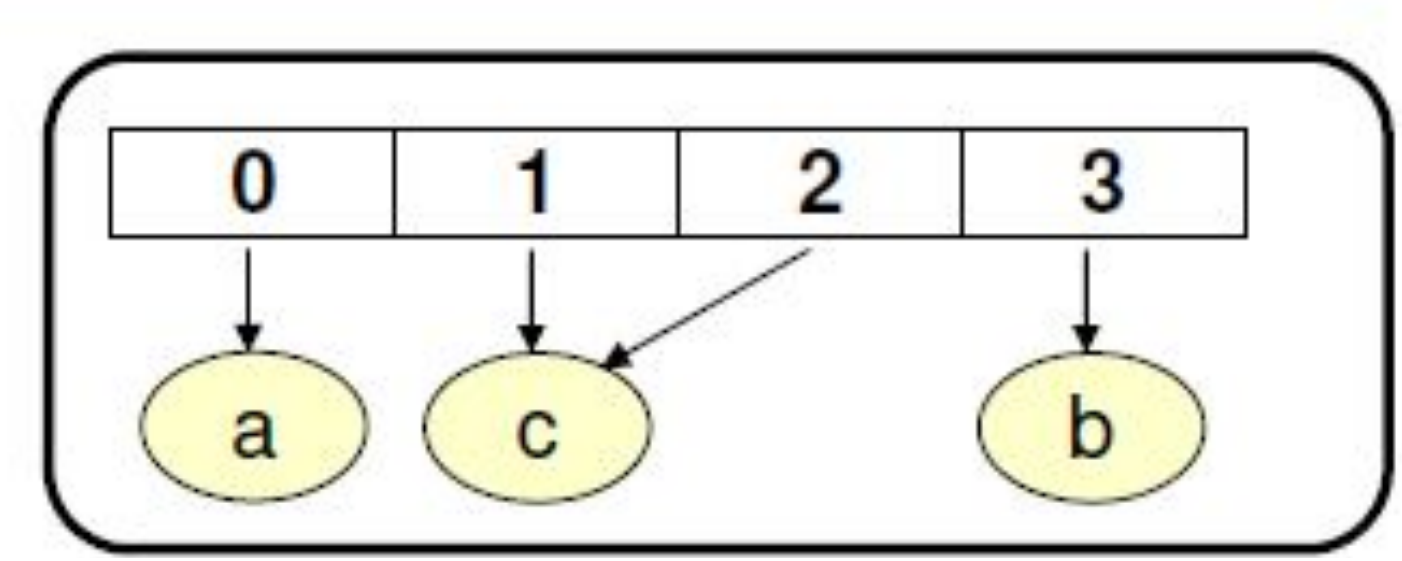
Drª. Alana Morais

alanamm.prof@gmail.com

Principais Coleções

Array Tradicional

- uma coleção de objetos *indexada por 0, 1, ...,N-1 com tamanho fixo.*



Coleções

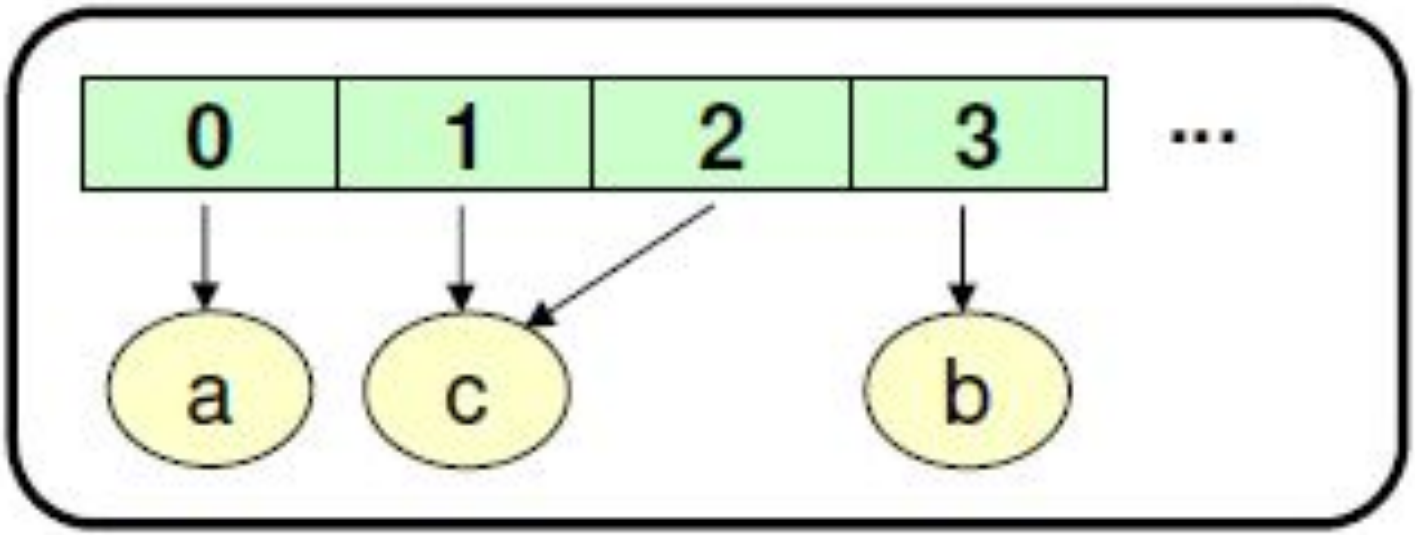
- Coleções permitem que um número arbitrário de objetos seja armazenado.

- Várias aplicações envolvem coleções de objetos:
  - Agendas Pessoais
  - Catálogos de bibliotecas
  - Sistema de registro de alunos

- O número de itens armazenados varia:
  - Itens podem ser adicionados.
  - Itens podem ser excluídos.

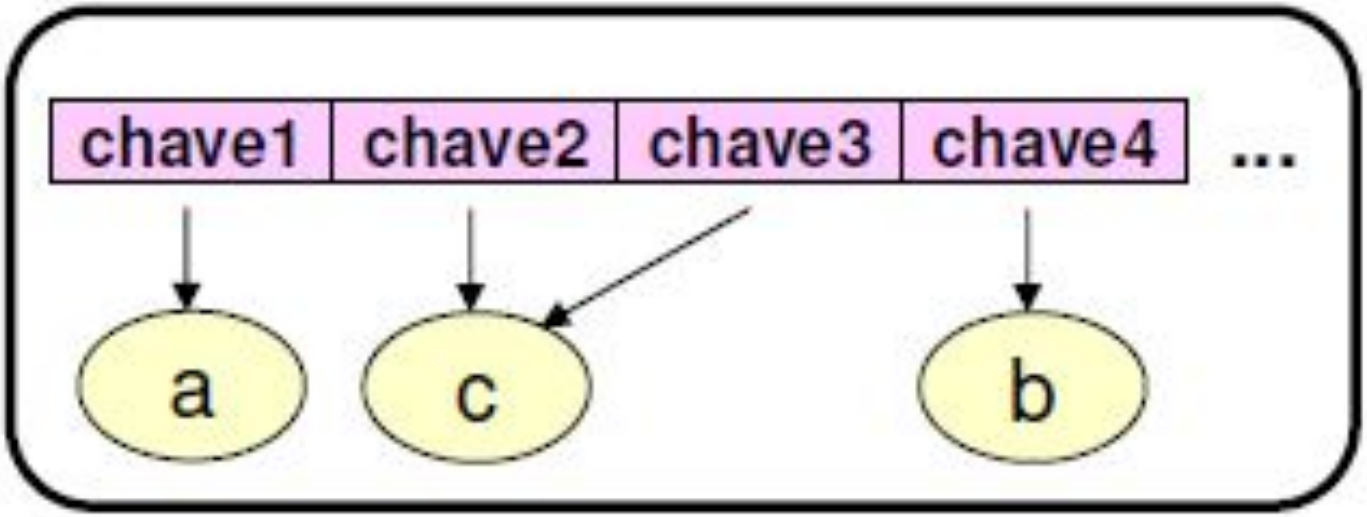
Principais Coleções

- Classe ArrayList
  - uma coleção de objetos *indexada por 0, 1, ..., com tamanho* variável.
  - Pode adicionar objetos de diversos tipos



Principais Coleções

- Classe HashMap
  - uma coleção de objetos *indexada por <chave>, com tamanho* variável.



ARRAYLIST

- Ações:
  - **Declaração**
  - **Adicionar valores**
  - **Operações**

ARRAYLIST

- Declaração
  - Coleções de objetos do tipo T
  - T pode ser uma classe encapsulada (Integer, String, Double, Float, etc) ou uma classe própria (Aluno, Livro, etc).

  - Sintaxe

ArrayList<Tipo> nome = new ArrayList<Tipo>();

ou

ArrayList nome = new ArrayList();

Mas antes precisamos importar a classe Arraylist ...
Adicione!

```
Import java.util.arraylist;
```

ARRAYLIST

- Ações:
  - **Declaração**
  - **Adicionar valores**
  - **Operações**

ARRAYLIST

- Adicionar Valores
  - Método add( )

```
lista.add(“alguma coisa");
lista.add(new Integer(10));
lista.add(new Aluno());

lista.add(1, “elemento”);
```

CLASSES ENCAPSULADORAS:
Integer, Long, Double, Boolean, etc...

ARRAYLIST

- Ações:
  - **Declaração**
  - **Adicionar valores**
  - **Operações**

ArrayList

- Operações
  - **Incluir** um objeto numa posição específica ou em qualquer posição.
  - **Obter** a quantidade de objetos.
  - **Percorrer** (iterar) a coleção.
  - **Localizar** um determinado objeto pela chave ou referência.
  - **Recuperar** um objeto pela posição ou chave ou referência.
  - **Atualizar** e **remover** um objeto pela posição ou referência.

ArrayList - API

- Principais métodos:

boolean add(Object o)
boolean add(int index, Object o)

Object get(int index)
Object set(int index, Object element)

boolean remove(int index)
Object remove(Object o)

ArrayList - API

- Percorrendo usando o tamanho físico

```
for (int i=0; i<alunos.size(); i++) {
 System.out.println(alunos.get(i).toString());
}
```

- Percorrendo usando o FOR (Java 1.5):

```
for (Aluno a: alunos) {
    System.out.println(a.imprima());
}
```

ArrayList - API

- **boolean add(Object o):**
  - Insere o objeto no final da lista

- **Exemplo:**

```
alunos.add(a1); // a1
alunos.add(a2); // a1, a2
alunos.add(a3); // a1, a2, a3

nomes.add("joao");
nomes.add("maria"); //joao, maria

numeros.add(new Double(100) );
numeros.add(new Integer(200)); //100, 200
```

ArrayList -API

- **boolean add(int index, Object o):**
  - Insere o objeto na posição (0,...,n-1), deslocando os demais objetos para direita.

- **Exemplo:**

```
alunos.add(a1); // a1
alunos.add(0, a2); // a2, a1
alunos.add(0, a3); // a3, a2, a1
```

ArrayList - API

- **Object get(int index):**
  - Retorna a referência do objeto da posição indicada

- **Exemplo:**

```
alunos.add(a1); // a1
alunos.add(a2); // a1, a2
alunos.add(a3); // a1, a2, a3

Aluno a = alunos.get(2); // retorna referencia a3
```

ArrayList - API

- **Object set(int index, Object o):**
  - Atualiza o objeto da posição indicada

- **Exemplo:**

```
alunos.add(a1); // a1
alunos.add(a2); // a1, a2
alunos.set(0,a3); // a3, a2

numeros.add(100);
numeros.set(0,101);
```

ArrayList - API

- **boolean remove(int index):**
  - Remove o objeto, fechando o "buraco" automaticamente.

- **Exemplo:**

```
alunos.add(a1);
alunos.add(a2); // a1, a2
alunos.remove(0); //a2

numeros.add(new Integer(100));
numeros.add(new Integer(200)); //100, 200
numeros.remove(1); //100
```

ArrayList - API

- **Object remove(Object o):**
  - Remove o objeto fechando o "buraco" automaticamente.

- **Exemplo:**

```
alunos.add(a1);
alunos.remove(a1);

Integer i = new Integer(100) ;
numeros.add(i);
numeros.remove(i);
```

ArrayList - API

- **int indexOf(Object o)**
  - Localiza um objeto e retorna índice i, tal que:
    - o.equals(get(i)) == true

- **Exemplo:**

  i = alunos.indexOf(umaluno) ;

  i = nomes.indexOf(“maria”) ;

  i = numeros.indexOf(a1) ;

ArrayList - API

- **boolean contains(Object o)**
  - If (nomes.contains (“joao”)) …

- Outros métodos
  - **boolean isEmpty( )**
    if (nomes.isEmpty( )) …
  - **int size( )**
    i = alunos.size( )
  - **void clear( )**
  - **int lastIndexOf (Object elem)**

Exercício resolvido

- Faça um programa que catalogue uma coleção de Livros e mostre suas informações ao final.
- Saiba que os livros devem ter nome(String), autor(String) e preço (double).

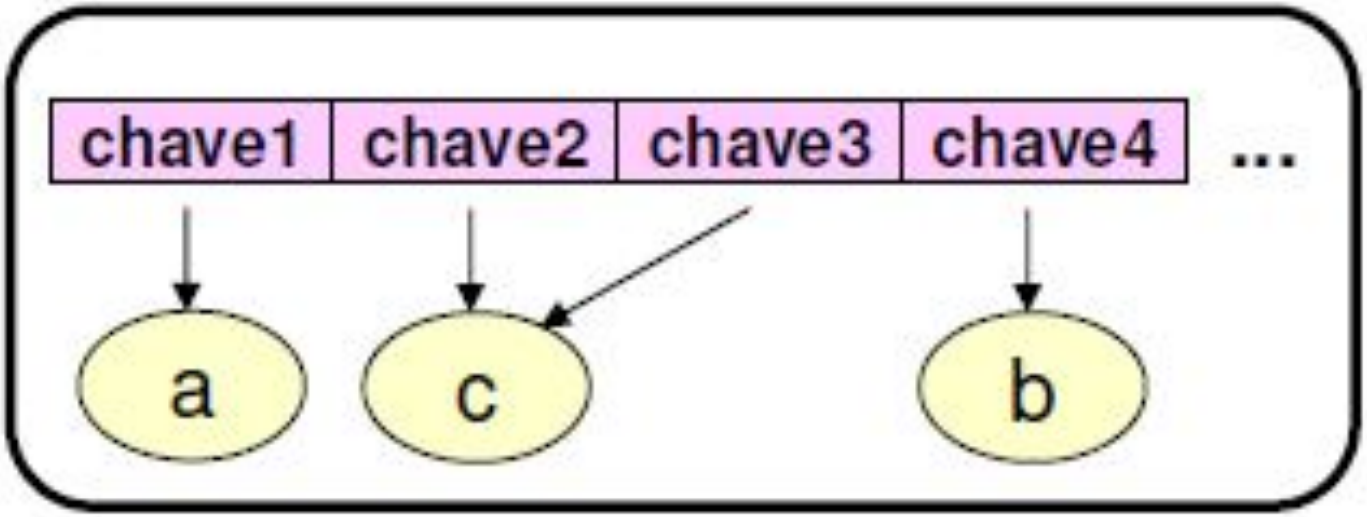
TED 4: Exercício

- Crie um projeto que utilize um ArrayList de objetos da **classe Conta** e determine o número da conta de maior saldo. Apresente o número e saldo da Conta com o maior saldo.

- Conta deve ter os atributos: numero (int) e saldo (double), além de seu construtores, gets, sets, toString.

Principais Coleções

- Classe HashMap
  - uma coleção de objetos *indexada por <chave>, com tamanho* variável.



HashMap

- Ações:
  - **Declaração**
  - **Adicionar valores**
  - **Operações**

HashMap

- Declaração
  - Coleções de objetos do tipo V
  - V pode ser uma classe encapsulada (Integer, String, Double, Float, etc) ou uma classe própria (Aluno, Livro, etc).
  - C representa a chave de acesso e deve ser uma classe.

HashMap<C,V> c HashMap = new HashMap<C,V>( );

ou

HashMap c = new HashMap();

Novamente não esqueça!

Adicione a classe ....

Import java.util.hashmap;

HashMap

- Ações:
  - **Declaração**
  - **Adicionar valores**
  - **Operações**

HashMap

- Adicionar Valores
  - Método put()

c.put(“chave1", new Integer(1));
c.put(“chave2", new Double(11));

c.put(new Aluno, new Integer(1));
c.put(new Float(33), new Double(11));

HashMap

- Ações:
  - **Declaração**
  - **Adicionar valores**
  - **Operações**

HashMaP

- Operações
  - **Incluir** um objeto numa posição específica ou em qualquer posição.
  - **Obter** a quantidade de objetos.
  - **Percorrer** (iterar) a coleção.
  - **Localizar** um determinado objeto pela chave ou referência.
  - **Recuperar** um objeto pela posição ou chave ou referência.
  - **Atualizar** e **remover** um objeto pela posição ou referência.

HashMap

- Object get(Object key)
  - Object valor = listaHash.get(“chave1”);
- void replace(K key, V value)
- int size()
- boolean containsValue(Object value)

HashMap

- Exercício

1) Faça um exemplo de lista em HashMap usando como chave-conteúdo a estrutura nome-data de nascimento dos usuários.

HashMap

- Exercício
  - Faça um sistema de matrícula de Alunos.
  - Crie um HashMap composto por chave String e conteúdo String para representar a matrícula e nome do aluno.
  - Insira, remova e consulte elementos desta lista em uma classe teste.

Dúvidas?