UFMS DIGITAL

Programação Orientada a Objetos

Prof. Dr. Anderson V. de Araujo

UFMS DIGITAL

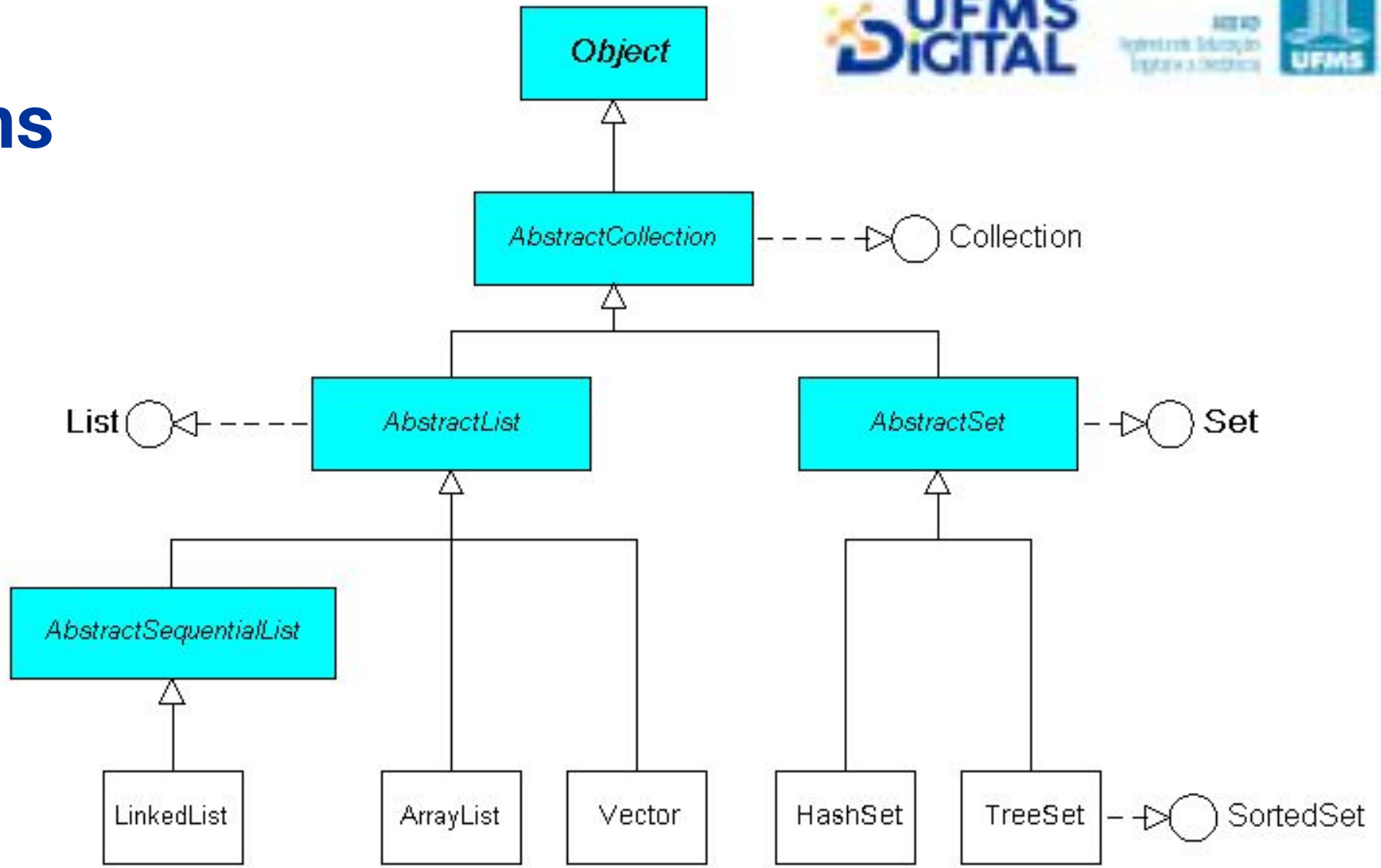
Módulo IV - Conceitos avançados

Unidade I - Coleções

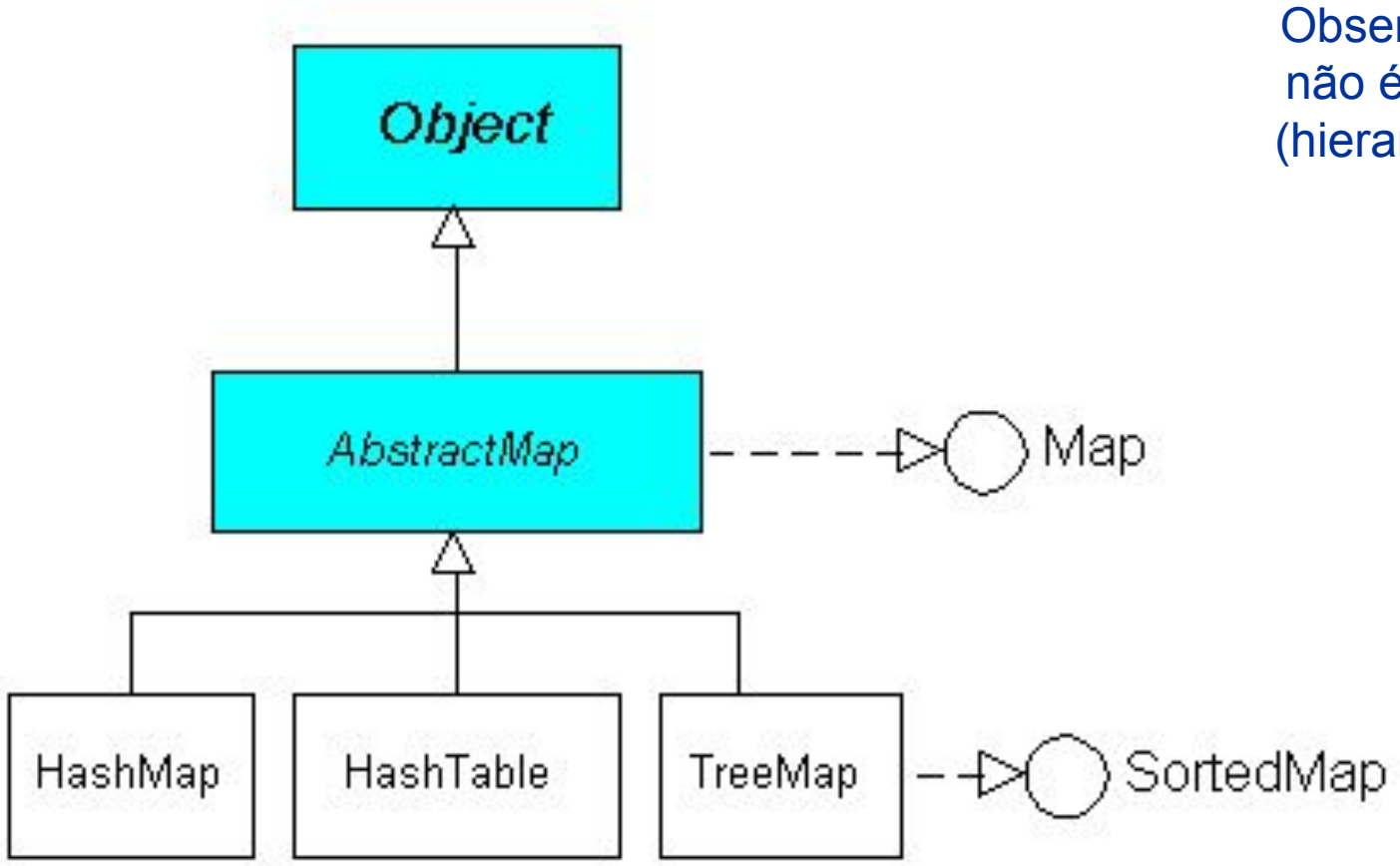
Introdução

- Java tem uma API específica para manipulação de objetos, denominada *Collections;*
- É uma solução flexível para armazenar objetos, o que facilita bastante a vida do desenvolvedor;
- A API Collections provê interfaces e classes para coleções;
- Numa coleção, a quantidade armazenada de objetos não é fixa, como ocorre com arrays;
  - Seu tamanho pode aumentar automaticamente conforme são adicionados mais elementos.
- Por meio destas coleções, é possível representar diferentes tipos de estruturas, como: listas, pilhas, filas, conjuntos e mapas.

Hierarquia das Collections (interfaces)



Hierarquia dos Maps (interfaces)



Observe que um Map não é uma *Collection* (hierarquia – herança)

A Interface List

- Uma lista é uma coleção de elementos arrumados em uma ordem linear:
  - É uma coleção em ordem (algumas vezes chamada de sequência);
  - Em geral, na ordem em que foram adicionados à lista;
- Isto é, cada elemento tem um antecessor (exceto o primeiro) e um sucessor (exceto o último);
- Normalmente implementada como Array ou Lista Encadeada.

List - Implementações

- **ArrayList**
  - A implementação mais utilizada da interface;
  - É implementada sobre um vetor:
    - Logo tem as vantagens e desvantagens associadas: acesso randômico, portanto é rápida para pesquisa.
  - Não é sincronizado, isto é não tem segurança para programação concorrente (*threads*)
- **Vector**: Ideal para acesso randômico, porém é sincronizado, logo, é mais lento.
- **LinkedList**: ideal para acesso sequencial:
  - É implementada sobre uma lista ligada. Logo tem as vantagens e desvantagens associadas;
  - Não é sincronizado, logo, não suporta *threads*.

Exemplo List (ArrayList)

```
public static void main(String[] args) {
   List list = new ArrayList();
   list.add(9);
   list.add(new Integer(8));
   list.add(new Integer(10));
   list.add(new Integer(6) + 3);
   list.add(7);
   for (int i = 0; i < list.size(); i++) {
      int x = (int)list.get(i);//cast necessário
      System.out.println(x);
   }
}
```

Saída: 9 8 10 9 7

For-each

- Sintaxe:
  - `for (Tipo nomeVar: Coleção)`
- Exemplo:
  - `for (Object obj: colecao_ou_vetor)`
    - Dá pra usar com *Generics*!
- Funciona para arrays também (já vimos isso!)

Exemplo 2 List (ArrayList)

```
public static void main(String[] args) {
   List list = new LinkedList();
   list.add(1);
   list.add("Strings");
   list.add("Teste");
   list.add(new Integer("1"));
   list.add(new ArrayList());
   System.out.println(list.get(2));
   for (Object obj: list) {
      System.out.println(obj);
   }
}
```

Não dá pra usar o for usando o índice (por exemplo, i), pois não existe a noção de posição.

Saída: Teste 1 Strings Teste 1 []

Licenciamento