

Agrupando Objetos

Instituto Metr pole Digital

Disciplina: IMD0040 - Linguagem de Programac o II

Docente: Emerson Alencar



A biblioteca de classes Java

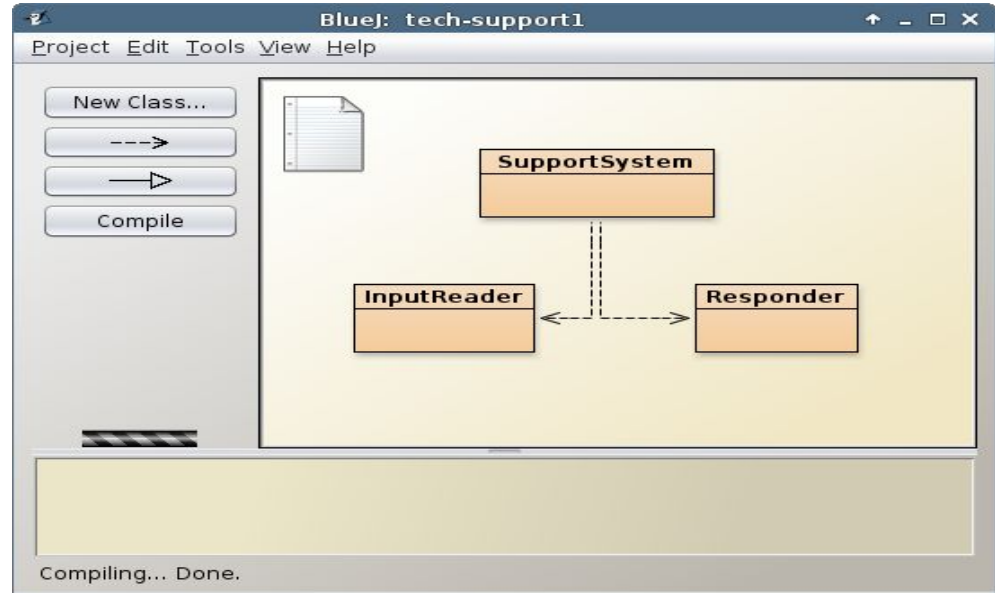
- Milhares de classes
- Dezenas de milhares de métodos
- Muitas classes úteis que tornam a vida muito mais fácil
- Não temos que escrever tudo do zero
- O Java chama suas bibliotecas de pacotes
- Um programador Java deve ser capaz de trabalhar com bibliotecas

Trabalhando com a biblioteca

- Você **NÃO** deve decorar todas as classes que existem
- Você deve:
- Conhecer algumas classes importantes por nome
- Saber como encontrar outras classes/métodos
- Programação OO
- Precisamos conhecer apenas a interface e não a implementação

Sistema de suporte técnico

- Projeto tech-support1
- Sistema de diálogo textual
- Baseado no 'Eliza' por Joseph Weizenbaum (MIT, 1960s)
- Abrir no BlueJ
- Crie um novo objeto
- SupportSystem
- Método start
- Crie outros objetos
- Outras classes



Loop principal

- Fazer o programa “ficar rodando” até o usuário escolher sair

```
boolean finished = false;
while (!finished) {
    //Faça algo
    if (condição de saída) { finished = true; }
    else { //Faça outra coisa }
}
```

- Lendo informações do usuário

```
String input = reader.getInput();
...
String response = responder.generateResponse();
System.out.println(response);
```

- Condição de saída

```
String input = reader.getInput();
if (input.startsWith("bye")) {
    finished = true;
}
```

Loop principal

- Fazer o programa “ficar rodando” até o usuário escolher sair

```
boolean finished = false;
while (!finished) {
    //Faça algo
    if (condição de saída) { finished = true; }
    else { //Faça outra coisa }
}
```

- Lendo informações do usuário

```
String input = reader.getInput();
...
String response = responder.generateResponse();
System.out.println(response);
```

- Condição de saída

```
String input = reader.getInput();
if (input.startsWith("bye")) {
    finished = true;
}
```

- De onde vem 'startsWith'?
- O que é? O que faz?
- Como podemos localizá-lo?

Lendo a documentação de classe

- Documentação das bibliotecas Java no formato HTML
- Abre em qualquer navegador Web
- *API: Application Programmers' Interface*
- Descrição de interface de todas as classes da biblioteca
- <http://docs.oracle.com/javase/8/docs/api/>

Interface versus implementação

- Documentação inclui:
 - Nome da classe
 - Descrição geral da classe
 - Lista de construtores e métodos
 - Valores de retorno e parâmetros para construtores e métodos
 - Descrição do propósito de cada construtor e método
- A interface da classe

Interface versus implementação

- Documentação **NÃO** inclui:
 - Campos privados
 - Maioria dos campos
 - Métodos privados
 - Corpo de cada método
 - Código fonte que implementa o método
- A implementação da classe

Utilizando classes de biblioteca

- Classes de biblioteca devem ser importadas
- `import`
- Exceto classes do pacote `java.lang`
- Pode-se importar uma única classe
 - `import java.util.ArrayList;`
- Pode-se importar pacotes inteiros
 - `import java.util.*;`
- Podem ser usadas como classes do seu projeto

Coleções

- Framework Java Collections
- Classes para agrupamento de objetos
- Tamanho dinâmico
- Aumenta sua capacidade de acordo com a necessidade
- Mantém contagem de itens
- Método de acesso `size()`
- Detalhes de como isso tudo é feito são ocultos
- Você não precisa saber como funciona para poder usar

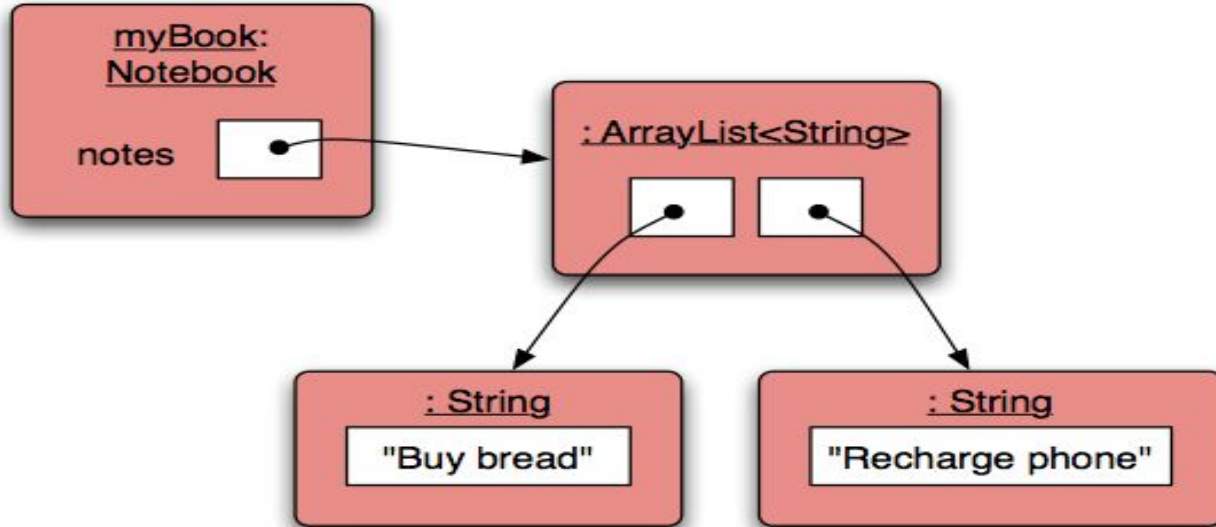
Classes genéricas

- Coleções são conhecidas como tipos parametrizados ou genéricos
- Na criação especificamos
- O tipo da coleção: ArrayList
- O tipo de objetos que ela conterá: String
- ArrayList de String

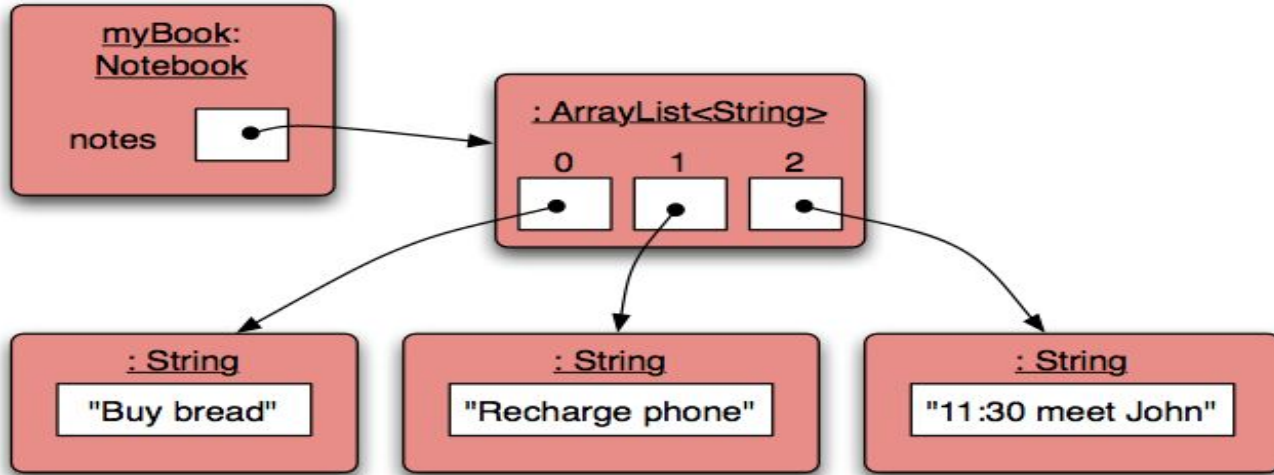
```
import java.util.ArrayList;  
...  
ArrayList<String> lista = new ArrayList<String>();
```

- ArrayList implementa a funcionalidade de lista
- add, get, remove, size, ...

Estruturas de objeto com coleções



Adicionando uma terceira anotação



Manipulando coleções

- Recuperando um objeto

```
public void showNote(int noteNumber){
    if (noteNumber < 0) {
        //numero invalido
    }
    else if (noteNumber < numberOfNotes()) {
        System.out.println(notes.get(noteNumber));
    }
    else {
        //numero invalido
    }
}
```

- Removendo um objeto

```
public void removeNote(int noteNumber){
    if (noteNumber < 0) {
        //numero invalido
    }
    else if (noteNumber < numberOfNotes()) {
        notes.remove(noteNumber);
    }
    else {
        //numero invalido
    }
}
```

Iteração em Java

- Instruções de loop em Java
- for, while, do-while – Similar a C++
- for-each
- Cabeçalho
- Para cada elemento element da coleção collection, faça
- Atenção para o tipo do elemento
- Corpo
- Ações a serem realizadas

```
for (elementType element : collection) {  
    loop body  
}
```

```
public void listNotes()  
{  
    for (String note : notes) {  
        System.out.println(note);  
    }  
}
```


Iteradores

- Objetos que permitem percorrer uma coleção

Utilizando um objeto Iterator

- Forma geral

```
Iterator<ElementType> it = myCollection.iterator();  
while (it.hasNext()) {  
    //chama it.next() para obter o próximo objeto  
    //faz algo com esse objeto  
}
```

```
public void listNotes(){  
    Iterator<String> it = notes.iterator();  
    while (it.hasNext()) {  
        System.out.println(it.next());  
    }  
}
```

```
public void listNotes(){  
    Iterator<String> it = notes.iterator();  
    while (it.hasNext()) {  
        String minhaNota = it.next();  
        System.out.println(minhaNota);  
    }  
}
```

Arrays em Java

```
private int[] hourCounts;  
private String[] names;  
...  
  
HourCounts = new int[24];  
...  
hourCounts[i] = 0;  
hourCounts[i]++;  
System.out.println(hourCounts[i]);
```

hourCounts



:int[]

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23



Literais e Comprimento de array

```
private int[] numbers = { 3, 15, 4, 5 };

int n = numbers.length;
System.out.println("Tamanho: " + n);  'length' não é um método
System.out.println(numbers[i]);
```

```
for (int hour = 0; hour < hoursCounts.length; hour++) {
    System.out.println(hour + ": " + hoursCounts[hour]);
}
```

```
int hour = 0;
while (hour < hoursCounts.length) {
    System.out.println(hour + ": " + hoursCounts[hour]);
    Hour++;
}
```

```
//Imprime múltiplos de 3 que estão abaixo de 40.
for (int num = 3; num < 40; num = num + 3) {
    System.out.println(num);
}
```

Utilizando Random

- Classe java.util.Random
- Geração de números aleatórios

```
import java.util.Random;
...
Random randomGenerator = new Random();
...
int index1 = randomGenerator.nextInt();
int index2 = randomGenerator.nextInt(100);
```

Gerando respostas aleatórias

```
import java.util.Random;

public Responder()
{
    randomGenerator = new Random();
    responses = new ArrayList<String>();
    fillResponses();
}

public String generateResponse()
{
    int index = randomGenerator.nextInt(responses.size());
    return responses.get(index);
}

public void fillResponses()
{ ... }
```

Outras Coleções

Mais classes de bibliotecas...

Utilizando conjuntos

```
import java.util.HashSet;

...
HashSet<String> mySet = new HashSet<String>();

mySet.add("one");
mySet.add("two");
mySet.add("three");

for (String element:mySet) {
    System.out.println(element);
}
```

- Um conjunto (set) é uma coleção que armazena cada elemento individual sem repetição. Ele não mantém qualquer ordem específica.
- Os elementos podem ser retornados pelo iterator em uma ordem diferente da que foram inseridos.

Mapas

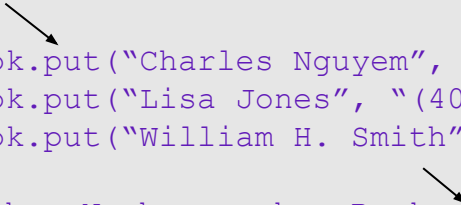
- Mapas são coleções que contêm pares de valores
- Chave (K)
- Valor (V)
- Tipos dos pares precisa ser definido pelo usuário
- Inserção funciona informando a chave e o valor
- Pesquisa funciona informando a chave e recuperando um valor
- Ex: uma lista telefônica

Utilizando mapas

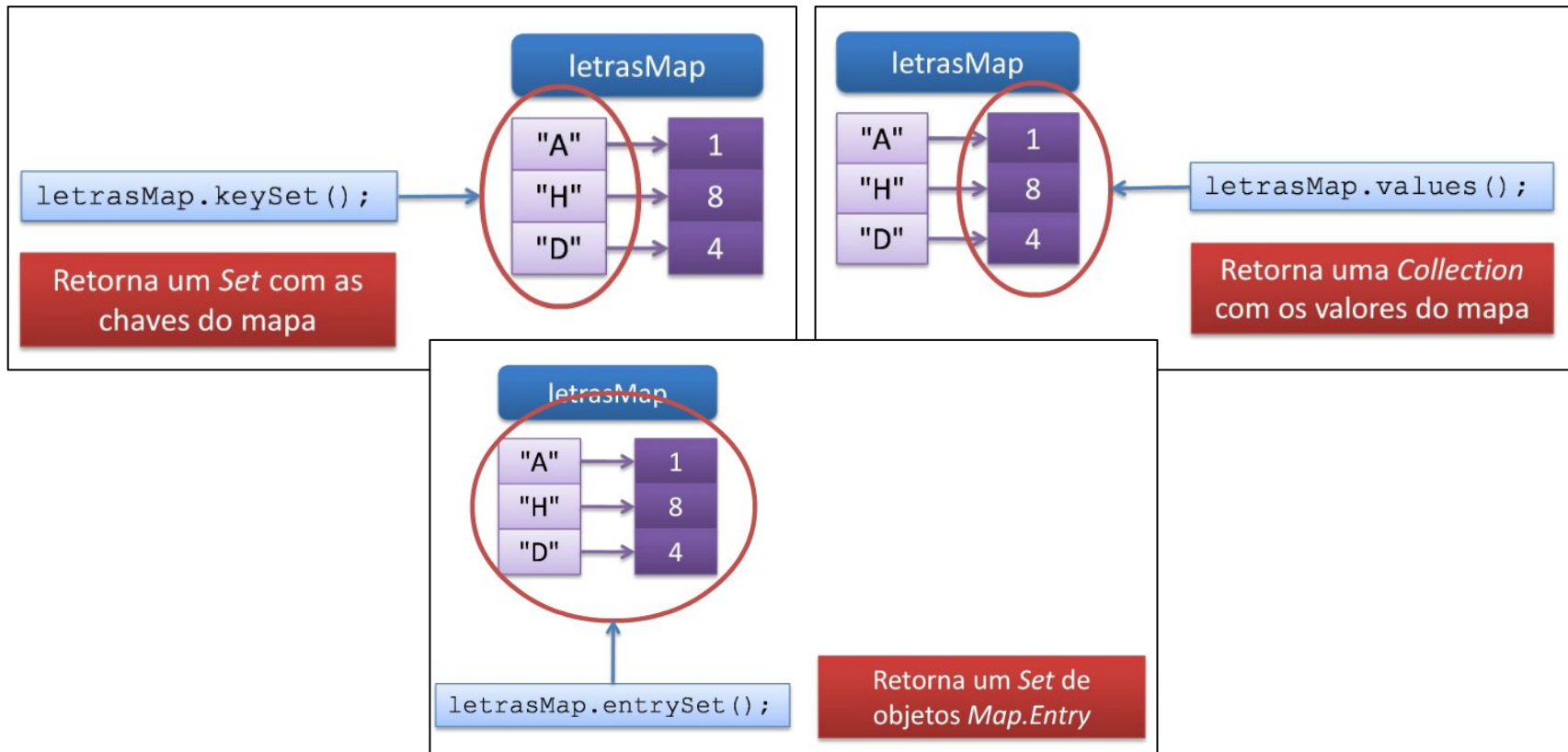
- Mapa com Strings como chaves e valores

"Charles Nguyen"	"(531) 9392 4587"
"Lisa Jones"	"(402) 4536 4674"
"William H. Smith"	"(998) 5488 0123"

```
HashMap <String, String> phoneBook =  
    new HashMap<String, String>();  
  
phoneBook.put("Charles Nguyem", "(531) 9392 4587");  
phoneBook.put("Lisa Jones", "(402) 4536 4674");  
phoneBook.put("William H. Smith", "(998) 5488 0123");  
  
String phoneNumber = phoneBook.get("Lisa Jones");  
System.out.println(phoneNumber);
```



Retornando coleções



Listas x Conjuntos x Mapas

- Listas
 - ArrayList
- Conjuntos
 - HashSet, TreeSet
- Mapas
 - HashMap, TreeMap

Tokenizando Strings

```
public HashSet<String> getInput()
{
    System.out.print(">");
    String inputLine =
        reader.nextLine().trim().toLowerCase();

    String[] wordArray = inputLine.split(" ");
    HashSet<String> words = new HashSet<String>();

    for (String word:wordArray) {
        words.add(word);
    }

    return words;
}
```

Público versus privado

- Atributos públicos (campos, construtores, métodos) são acessíveis a outras classes
- Campos não devem ser públicos (Em geral)
- Atributos privados são acessíveis apenas dentro de uma mesma classe
- Apenas métodos que são destinados a outras classes devem ser públicos

Ocultamento de informações

- Os dados que pertencem a um objeto são ocultados de outros objetos
- Saiba o que um objeto pode fazer, não como ele faz
- O ocultamento de informações aumenta o nível de independência
- Independência de módulos é importante para grandes sistemas e manutenção

Por hoje é só...