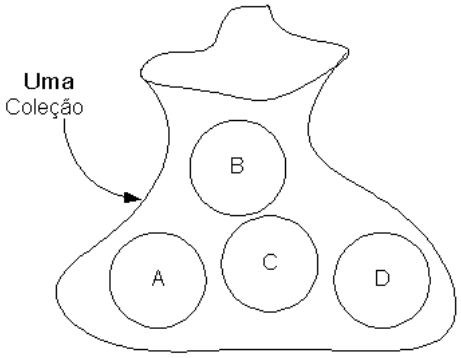
Orientação a Objeto - Coleções de Objetos

Objetivos da seção

- Apresentar o conceito de uma coleção de objetos
- Caracterizar o comportamento de uma coleção
- Ver como iteração numa coleção usando índices
- Entender a necessidade de cast
- Ver como iteração numa coleção usando um iterador
- Pesquisa em coleções

Coleções de Objetos: Iteração Usando Índices

- O que é uma coleção: apresentando o saco de objetos



- Nós somos clientes (usuários) da coleção: o que queremos fazer com ela?
- Nem sempre queremos fazer as mesmas coisas, mas as possibilidades úteis são:
    - Adicionar um objeto dentro da coleção
    - Remover um objeto da coleção
    - Pesquisar (achar a referência a) um objeto particular da coleção, dada uma chave
    - Iterar (ou varrer) os objetos da coleção
        - Isso significa fazer um loop tratando de cada objeto da coleção, um de cada vez
- Nem toda coleção permite todas as operações acima
- O programa Cartas4.java mostra que Baralho é uma coleção
    - Contém outros objetos (Cartas)
    - Porém, é uma coleção peculiar, porque:
        - Já vem cheia de objetos
        - Não tem método para adicionar Carta
        - O método pegaCarta() permite fazer uma iteração dos objetos, conjuntamente com a remoção dos objetos
- Normalmente, queremos coleções com comportamento mais completo

O Array como Coleção

- O saco de objetos não poderia ser um array?
- Sim, como mostra o programa Cadastro1.java

```
/*
 * Uso de arrays
 */

import p1.io.*;
import java.util.*;

public class Cadastro1 {
        public static void main(String[] args) {
                final int MAX_PESSOAS = 10;
                final String prompt = "Digite o nome de uma pessoa: ";
```

```
                String[] cadastro = new String[MAX_PESSOAS];
                String nome;
                // entrada dos dados de cadastro
                int numPessoas = 0;
                while ((nome = Entrada.in.lerLinha(prompt)) != null) {
                        cadastro[numPessoas++] = nome;
                }

                // imprime o cadastro antes da ordenação
                System.out.println();
                for (int i = 0; i < numPessoas; i++) {
                        System.out.println(cadastro[i]);
                }

                // ordena o cadastro
                String[] cadOrdenado = new String[numPessoas];
                for (int i = 0; i < numPessoas; i++) {
                        cadOrdenado[i] = cadastro[i];
                }
                Arrays.sort(cadOrdenado);

                // imprime o cadastro ordenado
                System.out.println();
                for (int i = 0; i < numPessoas; i++) {
                        System.out.println(cadOrdenado[i]);
                }
        } // main
} // Cadastro1
```

- A saída do programa:

```
Digite o nome de uma pessoa: roberto
Digite o nome de uma pessoa: jacques
Digite o nome de uma pessoa: camilo
Digite o nome de uma pessoa: peter
Digite o nome de uma pessoa: aninha
Digite o nome de uma pessoa: jean
Digite o nome de uma pessoa: juliana
Digite o nome de uma pessoa: bruno
Digite o nome de uma pessoa: rodrigo
Digite o nome de uma pessoa: alexandre
Digite o nome de uma pessoa: ^Z

roberto
jacques
camilo
peter
aninha
jean
juliana
bruno
rodrigo
alexandre

alexandre
aninha
bruno
camilo
jacques
```

```
jean
juliana
peter
roberto
rodrigo
```

- Observe que ^Z (CTRL+Z) é a forma de indicar fim de arquivo no teclado
- Tente rodar o programa tendo o cadastro pronto num arquivo de texto

```
java -classpath .;packagep1\p1.jar Cadastro1 < cadastro.txt
```

- Pergunta: Usando um array como coleção, como implementar as 4 operações?
  - Adicionar
  - Remover
  - Pesquisar
  - Iterar
- O que ocorre se digitar mais do que 10 nomes na entrada?
  - Tente!
- Como solucionar?
  - Ver Cadastro2.java
- Existem arrays que crescem automaticamente?
  - Sim, Java tem várias coleções que fazem isso
  - Um exemplo é a coleção ArrayList que se comporta como um array que ajusta seu tamanho
  - Ver o programa Cadastro3.java

```
/*
 * Cadastro com ArrayList em vez de array.
 * Mostra como ArrayList é essencialmente um array que cresce sob demanda.
 */

import p1.io.*;
import java.util.*;

public class Cadastro3 {
        public static void main(String[] args) {
                final String prompt = "Digite o nome de uma pessoa: ";

                List cadastro = new ArrayList();
                String nome;
                // entrada dos dados de cadastro
                while ((nome = Entrada.in.lerLinha(prompt)) != null) {
                        cadastro.add(nome);
                }

                // imprime o cadastro antes da ordenação
                // (deveria usar um iterador, mas queremos mostrar
                // como o ArrayList é semelhante a um array)
                System.out.println();
                for (int i = 0; i < cadastro.size(); i++) {
                        System.out.println(cadastro.get(i));
                }

                // ordena o cadastro
                Collections.sort(cadastro);

                // imprime o cadastro ordenado
                System.out.println();
                for (int i = 0; i < cadastro.size(); i++) {
```

```
                        System.out.println(cadastro.get(i));
                }
        } // main
} // Cadastro3
```

- Observe a declaração de cadastro
  - O tipo genérico é List
  - O objeto que criamos é da classe ArrayList
  - ArrayList "é uma" List
  - Tem outros tipos de objetos que se comportam como List
  - Falaremos disso mais na frente quando falarmos de interfaces
- Observações sobre ArrayList
  - Qualquer Objeto pode ser colocado dentro de um ArrayList
    - Aqui, colocamos objetos do tipo String
    - Em Java, um objeto genérico que pode ser qualquer coisa é um Object
  - O número de objetos no ArrayList é size()
  - Para acessar o i-ésimo elemento de um ArrayList, use get(i)

Coleções de Objetos: Iteração Usando um Iterador

- Tem outra forma de iterar uma coleção sem usar índices
- Usando um "iterador", podemos dizer: "me dê o próximo, me dê o próximo, me dê o próximo, ..." até acabar
- O primeiro exemplo usa um ArrayList
- Ver a solução em Cartas6.java

```
/*
 * Laço: Coleção genérica List
 */

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import p1.aplic.cartas.Baralho;
import p1.aplic.cartas.Carta;
import p1.io.Entrada;

public class Cartas6 {
        public static void main(String[] args) {

                Baralho baralho = new Baralho();
                baralho.baralhar();
                // List é uma "Coleção" de objetos genéricos
                List<Carta> aMao = new ArrayList<Carta>();

                int n = Entrada.in.lerInt("Quantas cartas na mao? ");

                for (int i = 0; i < n; i++) {
                        aMao.add(baralho.pegaCarta());
                }

                // dedo server para varrer (iterar) as cartas na mao
                Iterator<Carta> dedo = aMao.iterator();
                Carta maiorCarta = null;
                while (dedo.hasNext()) {
                        Carta proximaCarta = dedo.next();
                        if (maiorCarta == null || proximaCarta.compareTo(maiorC
                                maiorCarta = proximaCarta;
                        }
```

```
                }
                // List tem um toString bonitinho também! :-)
                System.out.println("A mao: " + aMao);
                System.out.println("A maior carta: "
                                + (maiorCarta == null ? "nao tem" : maiorCarta)
        }
}
```

- A saída do programa:

```
Quantas cartas na mao? 5
A mao: [SETE de ESPADAS, CINCO de ESPADAS, TRES de OUROS, NOVE de PAUS, SEIS de
A maior carta: NOVE de PAUS
```

- Um iterador é um objeto que nos ajuda a varrer (iterar) uma coleção, como um dedo nos ajuda a varrer as cartas de uma mão de cartas

Coleções de Objetos: Pesquisa

- Uma operação importante que fazemos com coleções é a pesquisa
- Examine o programa Pesquisa1.java

```
/*
 * Laços aninhados
 */

import java.util.ArrayList;
import java.util.List;

import p1.aplic.geral.Pessoa;
import p1.io.Entrada;

public class Pesquisa1 {
        public static void main(String[] args) {
                final String prompt1 = "Digite o nome de uma pessoa (\"fim\" pa
                final String prompt2 = "Digite o CPF dessa pessoa: ";
                final String prompt3 = "Digite o CPF a pesquisar (\"fim\" para
                List cadastro;
                String nome;
                String cpf;

                cadastro = new ArrayList();

                // entrada dos dados de cadastro
                while ((nome = Entrada.in.lerLinha(prompt1)) != null
                                && !nome.equals("fim")) {
                        cpf = Entrada.in.lerLinha(prompt2);
                        if (cpf != null) {
                                cadastro.add(new Pessoa(nome, cpf));
                        }
                } // while

                // pesquisa de dados por cpf
                // observe o aninhamento de laços
                // Seria possivel usar o método indexOf de List, mas
                // queremos mostrar como fazer pesquisa sequencial num array
                while ((cpf = Entrada.in.lerLinha(prompt3)) != null
                                && !cpf.equals("fim")) {
                        boolean achei = false;
                        for (int i = 0; i < cadastro.size(); i++) {
```

```
                                Pessoa p = (Pessoa) cadastro.get(i);
                                if (p.getCPF().equals(cpf)) {
                                        System.out.println(p.getNome());
                                        achei = true;
                                }
                        } // for
                        if (!achei) {
                                System.out.println("Nao achei CPF " + cpf + " n
                        }
                } // while
        } // main
} // Pesquisa1
```

- Saída do programa

```
Digite o nome de uma pessoa ("fim" para terminar): jacques
Digite o CPF dessa pessoa: 123456789-10
Digite o nome de uma pessoa ("fim" para terminar): biluca
Digite o CPF dessa pessoa: 012345678-91
Digite o nome de uma pessoa ("fim" para terminar): fim
Digite o CPF a pesquisar ("fim" para terminar): 123456789-10
jacques
Digite o CPF a pesquisar ("fim" para terminar): 123456789-11
Nao achei CPF 123456789-11 no cadastro.
Digite o CPF a pesquisar ("fim" para terminar): 012345678-91
biluca
Digite o CPF a pesquisar ("fim" para terminar): fim
```

- Observações
  - Veja como incluir um " dentro de um string: usando \"
  - Que coleção estamos usando? para quê?
  - Objetos de qual classe são colocados na coleção?
  - Qual é - sentido da linha
    - if(cpf != null) {
  - Também estamos vendo um exemplo de loops aninhados
  - Há coleções melhores do que List para fazer pesquisa
    - Veremos isso adiante (vejam Map aqui)
    - Exercício: refaça o programa de pesquisa acima usando um Map

Os Comandos break e continue

- Vamos agora ver um exemplo do comando "break" para sair de um loop
- Examine o programa Pesquisa2.java

```
/*
 * Uso de break
 */

import java.util.ArrayList;
import java.util.List;

import p1.aplic.geral.Pessoa;
import p1.io.Entrada;

public class Pesquisa2 {
        public static void main(String[] args) {
                final String prompt1 = "Digite o nome de uma pessoa (\"fim\" pa
                final String prompt2 = "Digite o CPF dessa pessoa: ";
                final String prompt3 = "Digite o CPF a pesquisar (\"fim\" para
                List cadastro;
```

```
                String nome;
                String cpf;

                cadastro = new ArrayList();

                // entrada dos dados de cadastro
                while ((nome = Entrada.in.lerLinha(prompt1)) != null
                                && !nome.equals("fim")) {
                        cpf = Entrada.in.lerLinha(prompt2);
                        if (cpf != null) {
                                cadastro.add(new Pessoa(nome, cpf));
                        }
                } // while

                // pesquisa de dados por cpf
                // observe o aninhamento de laços
                // Seria possivel usar o método indexOf de List, mas
                // queremos mostrar como fazer pesquisa sequencial numa List
                while ((cpf = Entrada.in.lerLinha(prompt3)) != null
                                && !cpf.equals("fim")) {
                        Pessoa p = null;
                        for (int i = 0; i < cadastro.size(); i++) {
                                p = (Pessoa) cadastro.get(i);
                                if (p.getCPF().equals(cpf)) {
                                        System.out.println(p.getNome());
                                        break; // cai fora do laço de pesquisa
                                }
                        } // for
                        if (p == null || !p.getCPF().equals(cpf)) {
                                System.out.println("Nao achei CPF " + cpf + " n
                        }
                } // while
        } // main
} // Pesquisa2
```

- Pergunta: no último "if", por que testar p==null?
- Um outro exemplo para ver o uso do comando "continue"
- Examine o programa Pesquisa3.java

```
/*
 * Uso de continue
 * Cadastro onde linha iniciando com # são comentários
 */

import java.util.ArrayList;
import java.util.List;

import p1.aplic.geral.Pessoa;
import p1.io.Entrada;

public class Pesquisa3 {
        public static void main(String[] args) {
                final String prompt1 = "Digite o nome de uma pessoa (\"fim\" pa
                final String prompt2 = "Digite o CPF dessa pessoa: ";
                final String prompt3 = "Digite o CPF a pesquisar (\"fim\" para
                final String INICIO_COMENTÁRIO = "#";
                List cadastro;
                String nome;
                String cpf;
```

```
                cadastro = new ArrayList();

                // entrada dos dados de cadastro
                while ((nome = Entrada.in.lerLinha(prompt1)) != null
                                && !nome.equals("fim")) {
                        if (nome.startsWith(INICIO_COMENTÁRIO)) {
                                continue;
                        }
                        // agora, processa informação de verdade
                        cpf = Entrada.in.lerLinha(prompt2);
                        if (cpf != null) {
                                cadastro.add(new Pessoa(nome, cpf));
                        }
                } // while

                // pesquisa de dados por cpf
                // observe o aninhamento de laços
                // Seria possivel usar o método indexOf de List, mas
                // queremos mostrar como fazer pesquisa sequencial numa List
                while ((cpf = Entrada.in.lerLinha(prompt3)) != null
                                && !cpf.equals("fim")) {
                        Pessoa p = null;
                        for (int i = 0; i < cadastro.size(); i++) {
                                p = (Pessoa) cadastro.get(i);
                                if (p.getCPF().equals(cpf)) {
                                        System.out.println(p.getNome());
                                        break; // cai fora do laço de pesquisa
                                }
                        } // for
                        if (p == null || !p.getCPF().equals(cpf)) {
                                System.out.println("Nao achei CPF " + cpf + " n
                        }
                } // while
        } // main
} // Pesquisa3
```

- Saída do programa

```
Digite o nome de uma pessoa ("fim" para terminar): jacques
Digite o CPF dessa pessoa: 123456789-01
Digite o nome de uma pessoa ("fim" para terminar): aninha
Digite o CPF dessa pessoa: 012345678-90
Digite o nome de uma pessoa ("fim" para terminar): # isso eh um comentario
Digite o nome de uma pessoa ("fim" para terminar): #isso tambem
Digite o nome de uma pessoa ("fim" para terminar): fim
Digite o CPF a pesquisar ("fim" para terminar): 123456789-01
jacques
Digite o CPF a pesquisar ("fim" para terminar): 012345678-90
aninha
Digite o CPF a pesquisar ("fim" para terminar): #isso tambem
Nao achei CPF #isso tambem no cadastro.
Digite o CPF a pesquisar ("fim" para terminar): fim
```

Programas Adicionais para o Aluno Estudar:

- Cartas8.java
- Cartas9.java
- Cartas10.java

oo-2 anterior próxima