INSTITUTO FEDERAL
São Paulo
Campus Caraguatatuba

PROGRAMAÇÃO DE COMPUTADORES 2

PROF. EDERSON / PROFª. JULIANA

INSTITUTO FEDERAL
São Paulo
Campus Caraguatatuba

COLEÇÕES

Aula 02 – 05/08/2025

Curso Superior de TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

INTRODUÇÃO

- Collections é um objeto que pode agrupar vários outros objetos (conjunto de objetos).

- Uma coleção tem por objetivo facilitar a inserção, busca e recuperação destes objetos agrupados sistematicamente.

COLLECTION FRAMEWORK

- Um arcabouço lógico para este tipo abstrato de dados, que se compõe de:
  - Interfaces que representam e manipulam a coleção independentemente dos detalhes de implementação;
  - Implementações que são as materializações, a codificação das interfaces; e
  - Algoritmos que são os métodos usados para realizar as tarefas propostas pelas interfaces, como ordenação, inserção e recuperação dos objetos armazenados.

BENEFÍCIOS

- redução do esforço de programação;
- redução dos custos de implementação;
- aumento da eficiência da geração de códigos;
- interoperabilidade de codificação;
- redução do esforço de aprendizagem;
- entre outros.

INTERFACE COLLECTION

- Possui métodos para adicionar, remover e buscar elementos (e mais alguns outros)
  - `size()` – retorna a quantidade de objetos na coleção.
  - `isEmpty()` - retorna verdadeiro (true), se a coleção está vazia; e falso (false), caso contrário.
  - `add (Objeto o)` – adiciona um elemento à coleção.
  - `remove (Objeto o)` – remove um objeto da coleção.

TIPOS DE COLEÇÕES

- Lista (`List`): uma sequência de elementos. Mantém dados a respeito de ambos, a ordem e a contagem.
- Conjunto (`Set` e `SortedSet`): uma coleção de elementos que não mantém uma ordem nem uma contagem dos elementos. Cada elemento ou está no conjunto ou não (não há elementos repetidos)
- Mapa (`Map` e `SortedMap`): uma associação entre chaves e valores. Ele mantém um conjunto de chaves e mapeia cada chave para um único valor.
- Fila (`Queue`): um tipo de coleção para manter uma lista de prioridades, onde a ordem dos seus elementos, definida pela implementação determina essa prioridade.

INTERFACE `java.util.List`

- Estrutura de dados semelhante a `array`.
- Possui uma quantidade de elementos variável;
- Permite elementos duplicados;
- Mantém uma ordenação específica dos elementos.
- Exemplo de classes que implementam a interface `List`: `ArrayList` e `LinkedList`.

MÉTODOS PARA MANIPULAR OBJETOS `List`

- `get()`: para recuperar um Object de um determinado índice.
- `add()`: adiciona um elemento na lista, reorganizando ela.
- `remove()`: retira um elemento da lista, reorganizando ela.
- `contains()`: verifica se um Object passado como parâmetro existe na lista.
- `size()`: informa o tamanho da lista.
- `isEmpty()`: informa se a lista está vazia.
- `iterator()`: responsável por buscar novos elementos nas coleções.

INSTITUTO FEDERAL
São Paulo
Campus Caraguatatuba

EXEMPLO

```java
import java.util.Arrays;

/**...4 lines */
public class Aula08 {

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

        ArrayList<String> bandas = new ArrayList<>();

        bandas.add("Rush");
        System.out.print("Adicionando a banda Rush: ");
        System.out.println(Arrays.toString(bandas.toArray()));

        bandas.add("Beatles");
        System.out.print("Adicionando a banda Beatles: ");
        System.out.println(Arrays.toString(bandas.toArray()));

        bandas.add("Iron Maiden");
        System.out.print("Adicionando a banda Iron Maiden: ");
        System.out.println(Arrays.toString(bandas.toArray()));

        System.out.print("Quem está na índice 0: ");
        System.out.println( bandas.get(0) );

        System.out.print("Adicionando Tiririca onde estava o Rush: ");
        bandas.add(bandas.indexOf("Rush"), "Tiririca");
        System.out.println(Arrays.toString(bandas.toArray()));

        System.out.print("Número de elementos na lista: ");
        System.out.println(bandas.size());

        System.out.print( "Removendo o Tiririca: " );
        bandas.remove("Tiririca");
        System.out.println(Arrays.toString(bandas.toArray()));

        System.out.print("Removendo tudo: ");
        bandas.clear();
        System.out.println(Arrays.toString( bandas.toArray()));
    }
```

Curso Superior de TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

INTERFACE `java.util.Set`

- Estrutura de dados semelhante a conjuntos (matemática).
- Possui uma quantidade de elementos variável;
- Não permite elementos duplicados;
- Não garante que os elementos vão ser resgatados na ordem em que foram inseridos;
- Exemplo de classes que implementam a interface `Set`: `HashSet`, `LinkedHashSet` e `TreeSet`.

MÉTODOS PARA MANIPULAR OBJETOS `Set`

- `add(Object o)`: adiciona um Object passado como parâmetro no conjunto.
- `remove(Object o)`: retira um Object passado como parâmetro do conjunto.
- `contains(Object o)`: verifica se um Object passado como parâmetro existe no conjunto.
- `size()`: informa o tamanho do conjunto.

INSTITUTO FEDERAL
São Paulo
Campus Caraguatatuba

EXEMPLO

```java
import java.util.HashSet;

/**...4 lines */
public class ExemploHashSet {

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

        HashSet digits = new HashSet();

        digits.add(0);
        digits.add(1);
        digits.add(2);
        digits.add(3);
        digits.add(4);
        digits.add(5);
        digits.add(null);
        System.out.println("Todos os elementos do hashset:\t" + digits);

        digits.add(5);
        digits.add(2);
        System.out.println("Depois de adicionar dados duplicados: \t" + digits)

        System.out.println("\n-------Usando o Métodos Contains -------");
        System.out.println("digits.contains(0) : " + digits.contains(0));
        System.out.println("digits.contains(2) : " + digits.contains(2));
        System.out.println("digits.contains(3) : " + digits.contains(7));
        System.out.println("digits.contains(null) : " + digits.contains(null));

    }
```

INTERFACE `java.util.Map`

- Estrutura de dados semelhante a índices.
- Possui um identificador (valor chave) para cada objeto.
- Exemplo de classes que implementam a interface Map: `HashMap`, `TreeMap` e `HashTable`.

MÉTODOS PARA MANIPULAR OBJETOS `Map`

- `put(Object k, Object o)`: insere um objeto o, com chave k, ao mapa.
- `remove (Object k)`: remove um objeto com a chave k do mapa.
- `containsKey (Object k)`: retorna verdadeiro, para o caso em que a chave k verificada esteja presente no mapa; e falso, caso contrário.
- `containsValue (Object o)`: retorna verdadeiro, para o caso em que o objeto o verificado esteja presente no mapa; e falso, caso contrário.
- `get(Object o, Object k)`: recupera o objeto o especificado. Retorna o valor contido na chave k especificada.
- `size()`: retorna o tamanho do mapa.
- `entrySet()`: itera todas as entradas de um mapa.
- `values()`: cria uma coleção de valores do mapa.

INSTITUTO FEDERAL
São Paulo
Campus Caraguatatuba

EXEMPLO

```java
import java.util.HashMap;

/**...4 lines */
public class ExemploHashMap {

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

        HashMap<Integer, String> chaveNome = new HashMap<>();

        chaveNome.put (212133, "Bridget Logan");
        chaveNome.put (162348, "Ivan the Great");
        chaveNome.put(8082771, "Donald John Trump");

        System.out.println(chaveNome.containsKey(11111));
        System.out.println(chaveNome.containsValue("Donald John Trump"));
    }
}
```

INTERFACE `java.util.Queue`

- Estrutura de dados semelhante a fila
- Consiste em uma coleção de coisas munida de duas operações:
  - `enqueue`, que insere uma coisa na coleção, e
  - `dequeue`, que remove uma coisa antiga da coleção.

MÉTODOS PARA MANIPULAR OBJETOS `Queue`

- `enqueue(Object o)`: insere um objeto o na fila.
- `dequeue()`: remove o objeto mais antigo da fila.
- `isEmpty()`: verifica se a fila está vazia.
- `size()`: retorna o tamanho da fila.

INSTITUTO FEDERAL
São Paulo
Campus Caraguatatuba

EXEMPLO

```
import java.util.LinkedList;
import java.util.Queue;

/**...4 lines */
public class ExemploQueue {

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

        Queue<Integer> queueL = new LinkedList<>();
        for (int i = 5; i > 0; i--) {
            queueL.add(i);
        }
        System.out.println("Imprimindo sua Fila (FIFO) Lista Ligada: " + queueL);

        queueL.remove();

        System.out.println("Imprimindo sua Fila (FIFO) Lista Ligada: " + queueL);

    }
}
```

Curso Superior de TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

EXERCÍCIO PRÁTICO

INSTITUTO FEDERAL São Paulo Campus Caraguatatuba

GERENCIADOR DE CONTAS BANCÁRIAS COM COLEÇÕES

Expandir o gerenciador de contas básico para suportar o cadastro, visualização, saque e depósito de múltiplas contas bancárias, utilizando coleções Java para armazenar e gerenciar as contas em memória.

1. **Armazenamento das contas.** Utilize uma coleção Java adequada (por exemplo, `ArrayList<ContaCorrente>`) para armazenar todas as contas em memória.
2. **Leitura de contas do arquivo.** O arquivo `contas.txt` conterá múltiplas linhas, cada uma representando uma conta no formato: numero, titular, saldo. Implemente um método para carregar todas as contas do arquivo para a coleção.
3. **Interface gráfica.** Atualize a GUI para: Exibir uma lista ou tabela com todas as contas carregadas. Permitir ao usuário selecionar uma conta para visualizar seus detalhes (número, titular, saldo). Permitir operações de saque e depósito na conta selecionada. Exibir mensagens apropriadas para sucesso ou erros (ex: saldo insuficiente). Atualizar o saldo exibido após cada operação.
4. **Operações com contas.** Implementar funcionalidades para: Sacar um valor da conta selecionada (tratando exceção de saldo insuficiente). Depositar um valor na conta selecionada. Adicionar uma nova conta via GUI (opcional para extensão). Salvar as contas atualizadas de volta em um arquivo `contas_atualizadas.txt` com o mesmo formato.
5. **Persistência.** Após operações de saque e depósito, o arquivo com as contas deve ser atualizado para refletir os novos saldos.

INSTITUTO FEDERAL
São Paulo

Campus
Caraguatatuba