Instituto de computação uff

Coleções e Exceções

Leonardo Gresta Paulino Murta

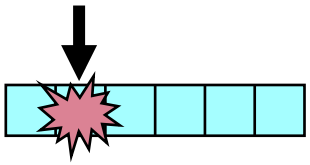
leomurta@ic.uff.br

Aula de hoje

- Estudaremos algumas das coleções disponíveis no Java
  - Lista
  - Conjunto
  - Dicionário
- Estudaremos também formas de tratamento de exceções
  - try...catch...finally

Coleções

- O **pacote java.util.*** define diversas estruturas de dados
- As estruturas implementam interfaces padrões:
  - Lista: List
  - Conjunto: Set
  - Dicionário: Map
- Cada interface tem uma implementação padrão (usualmente utilizada pelos programadores)
  - Lista: ArrayList
  - Conjunto: HashSet
  - Dicionário: HashMap

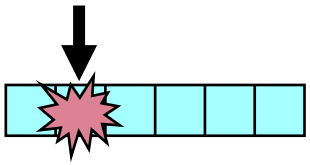


List

- A interface List (e a sua implementação padrão ArrayList) permite a criação de arrays dinâmicos
  - A lista pode conter qualquer tipo de objeto Java, em qualquer quantidade
  - Os elementos podem ser acessados em qualquer ordem
- Declarando um List e instanciando um ArrayList:

```
List<Pessoa> pessoas = new ArrayList<>()
```

Tipo que será guardado na lista



List

- Principais métodos:
  - **add(elemento)**: adiciona elemento no final da lista
  - **add(posição, elemento)**: adiciona elemento em uma posição da lista
  - **remove(elemento)**: remove um elemento da lista
  - **remove(posição)**: remove o elemento que está em uma posição da lista
  - **clear()**: remove todos os elementos da lista
  - **get(posição)**: retorna o elemento em uma posição da lista
  - **indexOf(elemento)**: retorna a posição de um elemento da lista
  - **isEmpty()**: informa se a lista está vazia
  - **size()**: informa o número de elementos da lista
- Ver demais métodos em http://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Exemplo

```
List<Pessoa> pessoas = new ArrayList<>();

pessoas.add(new Pessoa("João", 34));
pessoas.add(new Pessoa("Pedro", 14));
pessoas.add(new Pessoa("Paulo", 54));

for (Pessoa pessoa : pessoas) {
    System.out.print(pessoa.getNome() + " tem " +
                     pessoa.getIdade() + " anos.");
}

pessoas.clear();
```

Exemplo

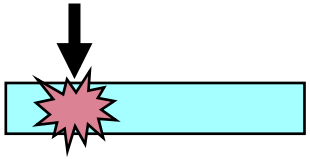
```
for (Pessoa pessoa : pessoas) {
    System.out.print(pessoa.getNome() + " tem " +
                     pessoa.getIdade() + " anos.");
}
```

é o mesmo que

```
for (int i = 0; i < pessoas.size(); i++) {
    Pessoa pessoa = pessoas.get(i);
    System.out.print(pessoa.getNome() + " tem " +
                     pessoa.getIdade() + " anos.");
}
```

Exercício

- Faça um programa que escreva a frase invertida (da última palavra para a primeira)
  - Use List para fazer a inversão

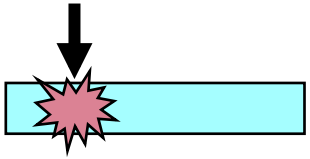


Set

- A interface Set (e a sua implementação padrão HashSet) permite a criação de conjuntos dinâmicos
  - Equivalente a lista, porém não impõe ordem aos elementos e não permite duplicata
- Declarando um Set e instanciando um HashSet:

```
Set<String> palavras = new HashSet<>()
```

Tipo que será guardado no conjunto



Set

- Principais métodos:
  - **add(elemento)**: adiciona elemento no conjunto
  - **remove(elemento)**: remove um elemento do conjunto
  - **clear()**: remove todos os elementos da lista
  - **contains(elemento)**: informa se o elemento está no conjunto
  - **isEmpty()**: informa se o conjunto está vazio
  - **size()**: informa o número de elementos do conjunto
- Ver demais métodos em http://docs.oracle.com/javase/8/docs/api/java/util/Set.html

Exemplo

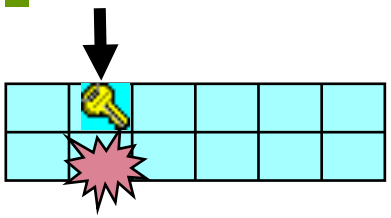
```
Set<String> palavras = new HashSet<>();

palavras.add("Flamengo");
palavras.add("Fluminense");
palavras.add("Botafogo");
palavras.add("Botafogo");

System.out.println(palavras.size());

for (String palavra : palavras) {
    System.out.println(palavra);
}
```

O que é mostrado aqui?

E aqui? Em qual ordem?

Exercício

- Faça um programa que leia uma frase e informe se há palavra repetida na frase
  - Use Set para fazer essa verificação



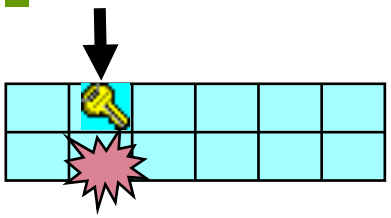
Map

- A interface Map (e a sua implementação padrão HashMap) permite a criação de dicionários dinâmicos
  - Um dicionário associa um objeto chave a um objeto valor (key → value)
- Declarando um Map e instanciando um HashMap:

```
Map<String,String> dddPorMunicipio = new HashMap<>()
```

Tipo que será valor do dicionário

Tipo que será chave do dicionário



Map

- Principais métodos:
  - **put(chave, valor)**: adiciona uma chave indexando um valor no dicionário
  - **get(chave):** retorna o valor indexado pela chave
  - **getOrDefault(chave, valor):** retorna o valor indexado pela chave ou o valor default informado
  - **keySet()**: retorna um conjunto com todas as chaves do dicionário
  - **remove(chave)**: remove o valor indexado pela chave no dicionário
  - **clear()**: remove todas as entradas do dicionário
  - **isEmpty()**: informa se o dicionário está vazio
  - **size()**: informa o número de entradas do dicionário
- Ver demais métodos em http://docs.oracle.com/javase/8/docs/api/java/util/Map.html

Exemplo

```
Map<String,String> dddPorMunicipio = new HashMap<>();

dddPorMunicipio.put("São Paulo", "11");
dddPorMunicipio.put("Rio de Janeiro", "21");
dddPorMunicipio.put("Belo Horizonte", "31");

for (String municipio : dddPorMunicipio.keySet()) {
    System.out.println("O DDD de " + municipio + " é " +
                       dddPorMunicipio.get(municipio));
}
```

Exercício

- Faça um programa que leia uma frase e informe o número de ocorrências de cada palavra da frase
  - Use Map para fazer essa contagem
- Dica: Java tem uma Classe para cada tipo primitivo, e faz a tradução automática entre ambos
  - Classe Integer para tipo int
  - Classe Double para tipo double
  - Classe Boolean para tipo boolean
  - Classe Character para tipo char
  - Etc.

Exceções

- Conceito
  - Exceções representam situações de erro, ocorridas durante a execução de um programa
  - Exemplos de exceções são divisão por zero ou incapacidade de ler dados de um arquivo

- Geradores de exceções
  - Interpretador Java: quando percebe uma situação de erro padrão (divisão por zero, falha de segurança, …)
  - Métodos do programa: quando percebe uma situação de erro interna do programa (informação inválida, …)

Tratamento de Exceções

- A palavra reservada *throws*, seguida pela classe de exceção gerada, deve ser indicada no cabeçalho de um método que gere uma exceção
- Os comandos *try-catch-finally* executam um código que pode gerar exceções de maneira segura, realizando o tratamento das exceções

```
public int gravaRegistro () throws IOException {
        …       // Código que gera a exceção
}
```

Tratamento de Exceções

```
try {
      // Código que pode disparar exceções
} catch (Excecao1 e) {
      // Código executado caso o código no bloco try
      // dispare uma exceção tipo Excecao1
}
...
catch (ExcecaoN e) {
      // Código executado caso o código no bloco try
      // dispare uma exceção tipo ExcecaoN
} finally {
      // Código executado sempre, mesmo que tenha ocorrido
      // uma exceção no bloco try
}
```

Propagação de Exceções

- A ocorrência de uma exceção transfere o fluxo de execução para o primeiro catch que trate a exceção
- Após o tratamento da exceção no catch, o fluxo é transferido para o finally do mesmo grupo try-catch-finally
- O finally é executado sempre, tendo ou não exceção, sendo útil para fechar arquivo, transação, etc.

Propagação de Exceções

```
getContent() {
    try {
        openConnection();
        readData();
    }
    catch (IOException e) {
        // Trata erro de E/S
    }
    ...
}
```

```
openConnection() throws IOException{
    openSocket();
    sendRequest();
    receiveResponse();
}
```

```
sendRequest() throws IOException{
    write(header);
    write(body); // Write Error!
}
```

Tipos de Exceção

- A biblioteca Java define alguns tipos de exceções:
  - Error: erro genérico
  - Exception: exceção genérica
  - RuntimeException: exceção detectada em tempo de execução
  - IOException: erros de entrada e saída
  - ArithmeticException: erro de cálculo algébrico (ex.: divisão por zero)
  - NullPointerException: erro de acesso a variável nula
  - SQLException: erro de acesso a banco de dados

- Diversos métodos das bibliotecas Java geram exceções
  - Estas exceções devem ser tratadas pelas classes que utilizem as classes das bibliotecas

Exercício

- Faça um código que leia dois valores do usuário e divida um pelo outro
- Duas situações lançam exceção:
  - Caso o usuário entre com valor não numérico
  - Caso o usuário entre com zero no segundo valor
- Teste as duas situações
- Escreva um tratamento de exceção para cada uma delas

Instituto de computação uff

Coleções e Exceções

Leonardo Gresta Paulino Murta

leomurta@ic.uff.br