

A thick black L-shaped frame is positioned on the left and bottom edges of the slide, framing the central text.

ALGORITMOS E PROGRAMAÇÃO II

Professor: Jânio Elias Teixeira Júnior



JAVA COLLECTIONS FRAMEWORK (JCF) E GENERICS



Java Collections Framework

■ Problema?

- *Arrays e Vetores;*
- *Alocação Estática;*
- *Difícil manipulação;*
 - Quantificar quantas posições estão preenchidas;
 - Remoção de um objeto;
 - Inserção de um objeto;
 - Buscar por um objeto;
 - Ordenação.

■ Solução ?

- *Estrutura de Dados*
 - Listas, Filas, Pilhas, etc.

Java Collections Framework

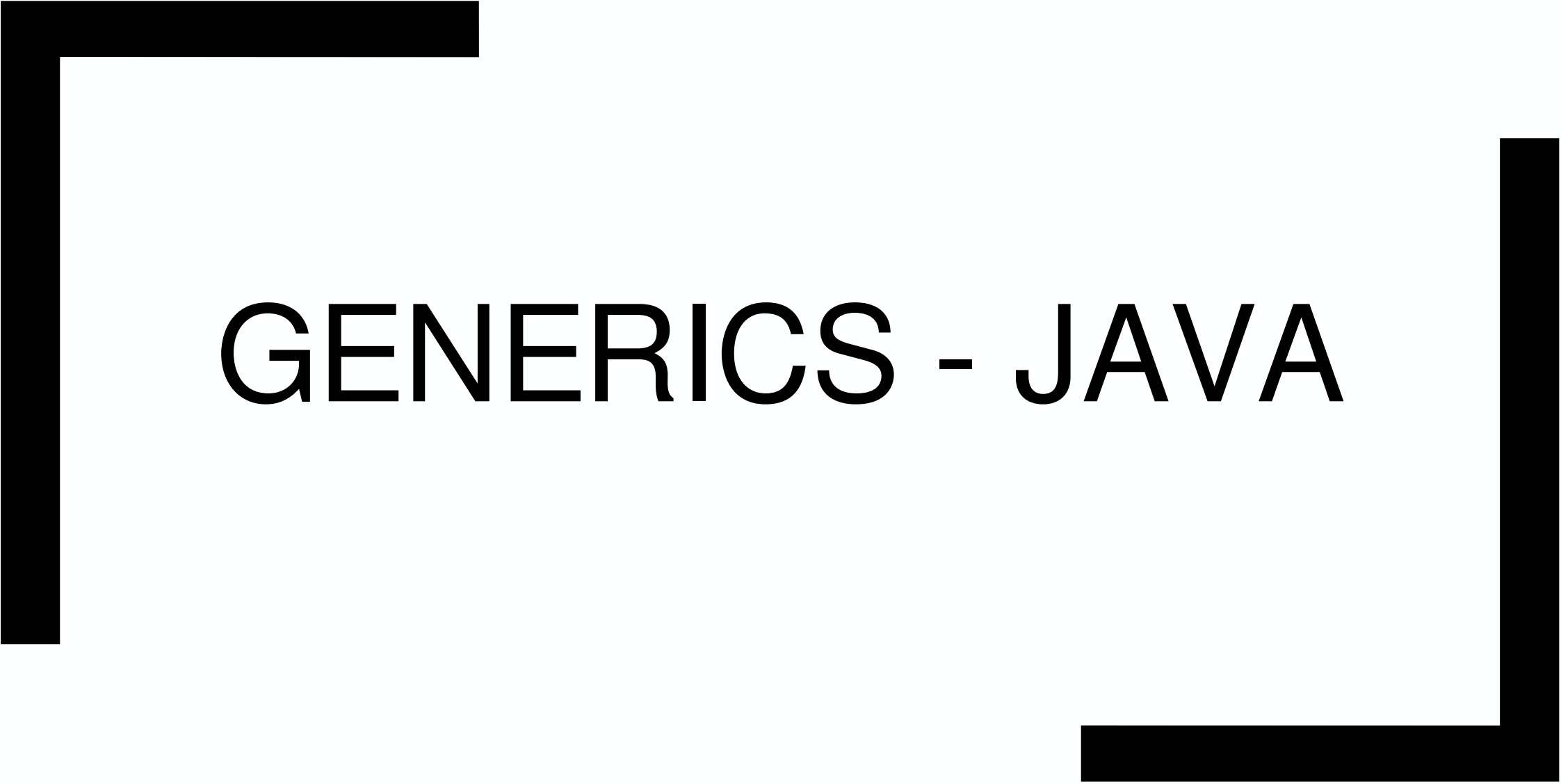
■ Listas:

- *java.util.List (Interface)*
- *É uma estrutura que permite **valores duplicados** e resolve todos os problemas levantados anteriormente de forma prática (na maioria das vezes);*
- *Existem diversas implementações para a interface List, sendo a ArrayList uma das mais utilizadas;*
- *Exemplo: **List lista = new ArrayList();***

Java Collections Framework

■ Principais métodos da interface List:

- ***add***, adiciona um novo objeto ao final da lista;
- ***isEmpty***, verifica se a lista está vazia;
- ***remove***, retira um objeto da lista (pela posição ou pelo valor*);
- ***clear***, remove todos os objetos da lista;
- ***contains***, verifica se existe um determinado objeto na lista;
- ***addAll***, adiciona uma coleção;
- ***get***, obtém um objeto através de seu índice;
- ***size***, retorna a quantidade de elementos da lista; e
- ***set***, altera o valor de uma determinada posição da lista.

The image features the text "GENERICS - JAVA" centered in a bold, black, sans-serif font. This text is enclosed within a decorative frame consisting of two thick, black L-shaped brackets. One bracket is positioned on the left side, with its vertical leg extending downwards and its horizontal leg extending to the right. The second bracket is on the right side, with its vertical leg extending upwards and its horizontal leg extending to the left. Together, they form a large, open rectangular shape that frames the central text.

GENERICS - JAVA

Problema?

■ O que é pior melhor:

- *Descobrir um **erro** ao **compilar** um código ou ao **executar** um programa ?*

Problema? Erro de compilação ou em tempo de execução?

```
public static void main(String[] args) {  
    01 Object o1 = 10;  
    02 Integer i1 = (int) o1;  
    03 Object o2 = (float) 20.02;  
    04 Integer i2 = (int) o2;  
    05 Object o3 = (double) 20.02;  
    06 Integer i3 = (int) o3;  
    07 Object o4 = "60";  
    08 Integer i4 = (int) o4;  
    09 System.out.println(i1 + i2 + i3 + i4);  
}
```


Erro em tempo de execução.

```
public static void main(String[] args) {
```

```
01 Object o1 = 10;
```

```
02 Integer i1 = (int) o1;
```

```
03 Object o2 = (float) 20.02;
```

```
04 Integer i2 = (int) o2;
```

```
05 Object o3 = (double) 20.02;
```

```
06 Integer i3 = (int) o3;
```

```
07 Object o4 = "60";
```

```
08 Integer i4 = (int) o4;
```

```
09 System.out.println(i1 + i2 + i3 + i4);
```

```
}
```

java.lang.Float cannot be cast to
java.lang.Integer

java.lang.Double cannot be cast to
java.lang.Integer

java.lang.String cannot be cast to
java.lang.Integer

Generics

- Objetivo: Adicionar estabilidade ao código, evitando possíveis erros que ocorreriam em tempo de execução;
- Generics permite que tipos (classes e interfaces) sejam parâmetros ao definir classes, interfaces e métodos.

Exemplo

■ Sem Generics:

```
List list = new ArrayList();  
list.add ("olá");  
String s = (String) list.get (0);
```

■ Com Generics:

```
List <String> list = new ArrayList <String>();  
list.add("olá");  
String s = list.get(0); // nenhum cast
```

Benefícios ao utilizar Generics

- Verificações de tipo em tempo de compilação;
 - *O compilador Java aplica a verificação de tipo forte ao código genérico e emite erros se o código violar a segurança do tipo. Corrigir erros de tempo de compilação é mais fácil do que corrigir erros de tempo de execução, o que pode ser difícil de encontrar.*
- Eliminação de ***casts***;
- Permite que os programadores implementem algoritmos genéricos que funcionam com diferentes tipos, de forma segura e fácil de ler.

Trabalho

- Desenvolva uma programa em Java que:
 - *Adicione 4 Pessoas em uma lista;*
 - Crie uma classe Pessoa contendo o **nome**, **cpf** e **sexo** como atributos;
 - Sobrescreva o método toString da classe Pessoa;
 - Sexo deve ser um enum;
 - Deve-se solicitar os dados ao usuário para o preenchimento da lista.
 - *Imprimir todos os dados da lista;*
 - Utilizar o seguinte método:
public static void imprimirTudo(List<Pessoa> lista);
 - *Após imprimir deve remover todas as pessoas do sexo masculino.*
 - *Imprimir todos os dados da lista novamente (Utilizando o método anterior).*

A thick black L-shaped frame is positioned on the left and bottom edges of the slide, framing the central text.

ALGORITMOS E PROGRAMAÇÃO II

Obrigado.