Tipos genéricos em Java

FURB
UNIVERSIDADE DE BLUMENAU

Programação genérica

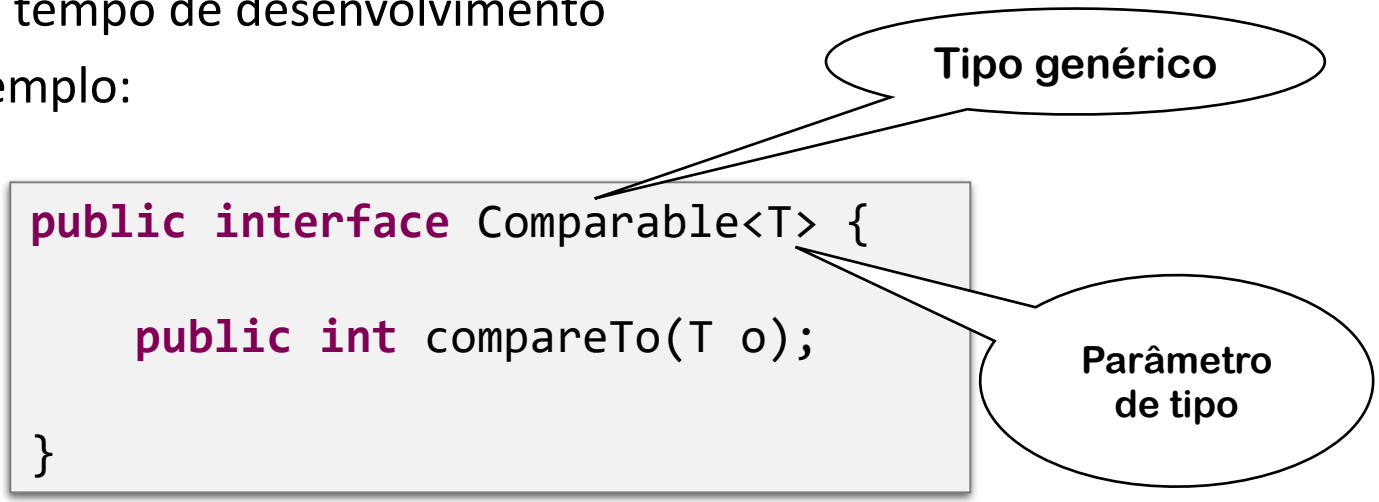
A programação genérica é um estilo de programação que permite que os algoritmos sejam escritos com tipos genéricos (classes/interfaces desconhecidas), mas que devem ser especificadas concretamente mais tarde.

Programação genérica

- Utilizado quando há algoritmos que diferem apenas pelo tipo de dado que manipulam.
  - Neste caso, aplicar a programação genérica reduz código duplicado
- Outros benefícios:
  - permitir o compilador detectar erros ao invés dos erros surgirem em tempo de execução.
  - Não é necessário realizar conversões (type casting) para tipos concretos

Tipo genérico

- É uma classe ou interface que manipula um tipo de dado desconhecido em tempo de desenvolvimento
- Exemplo:



- Os parâmetros de tipo são relacionados após o identificador da classe ou interface.
- Uma classe/interface genérica pode aceitar vários parâmetros de tipo
  - Portanto, chamamos de “classe genérica” as classes que são parametrizáveis
  - Igualmente, as interfaces genéricas são interfaces parametrizáveis

Exemplo

```
 1 public class Par <T,E> {
 2
 3     private T primeiro;
 4     private E segundo;
 5
 6     public void setPrimeiro(T primeiro) {
 7         this.primeiro = primeiro;
 8     }
 9
10     public T getPrimeiro() {
11         return primeiro;
12     }
13
14     public void setSegundo(E segundo) {
15         this.segundo = segundo;
16     }
17
18     public E getSegundo() {
19         return segundo;
20     }
21
22 }
```

Como informar um valor de parâmetro para um tipo genérico

FURB
UNIVERSIDADE DE BLUMENAU

Informando parâmetros para classes genéricas

- Quando o tipo genérico for utilizado, espera-se que seja fornecido um tipo concreto.
- Quando o tipo genérico é uma classe, a definição concreta do *parâmetro de tipo* ocorre na declaração da variável:

```
1 Par<String, Aluno> p1 = new Par<>();
2
3 p1.setPrimeiro("tecnologia");
4 p1.setSegundo(new Aluno(123,"Luiz"));
```

- Utiliza-se o operador “diamante” para definir a relação de tipos concretos
- No construtor, deve-se repetir o diamante novamente (neste caso, vazio)

Informando parâmetros para interfaces genéricas

- Quando o tipo genérico for uma interface, ao utilizá-lo, espera-se que seja informado um parâmetro (classe concreta).

```
public class MinhaClasse implements Comparable<Aluno> {

    @Override
    public int compareTo(Aluno o) {
        instruções
    }

}
```

Tipos genéricos

- Não é possível informar um tipo primitivo como argumento de tipo

```
Par<int, Aluno> p1 = new Par<>();
```

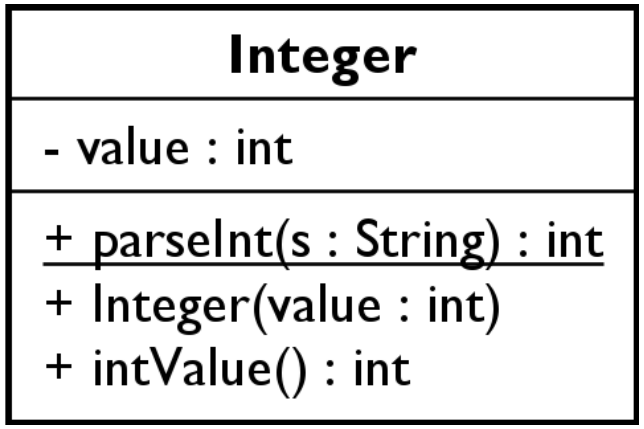
- Ao invés disso, deve-se utilizar as classes empacotadoras

```
Par<Integer, Aluno> p1 = new Par<>();
```

Classes Empacotadoras

- Uma classe empacotadora é uma classe que representa um dado primitivo.
- Java possui 8 classes empacotadoras:
  - Byte
  - Short
  - Integer
  - Long
  - Float
  - Double
  - Boolean
  - Character

Classe Empacotadora `Integer`



Para criar um objeto:

```
Integer idade = new Integer(21);
```

Autoboxing e Autounboxing

- Java permite que tipos primitivos e objetos de classes empacotadoras possam ser convertidos automaticamente.
- O compilador irá automaticamente empacotar (autoboxing) um valor primitivo quando estiver num contexto que exigir um objeto.

```
Integer idade = 21;
```

```
Integer idade = new Integer(21);
```

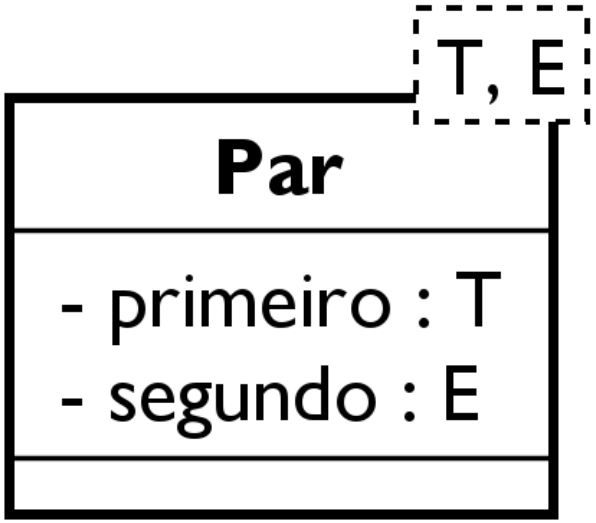
- O compilador irá automaticamente desempacotar (autounboxing) um objeto que estiver num contexto que exigir um valor primitivo

```
int filhos = new Integer(1);
```

```
int filhos = 1;
```

Tipo genérico na UML

- No diagrama UML, um tipo genérico é identificado através do uso do nome do *parâmetro de tipo* no canto superior direito.



Instanciação de arrays com tipo genérico

- Em Java, para criar um vetor com objetos cujo tipo é uma classe genérica

```
variavel = (T[]) new Object[tamanho];
```

- Onde **T** é parâmetro de tipo