Generics

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Prof. Edwar Saliba Júnior
Setembro / 2023

Genéricos

- Programação genérica consiste na criação de estruturas de programação que podem ser usadas com tipos de dados diferentes;
- exemplo: a classe `ArrayList<?>` é genérica;
- estrutura de dados que pode ser instanciada para coleções de diferentes tipos de dados e
- o tipo é verificado durante a compilação do programa.

Convenção

- São usadas letras maiúsculas para especificação de parâmetros de tipos:
  - `E` – (*element*) elemento de uma coleção;
  - `K` – (*key*) chave em um mapa;
  - `V` – (*value*) valor em um mapa;
  - `T` – (*type*) tipo genérico e
  - `S`, `U` – tipos adicionais.

Instanciação

- Como instanciar?

```
NomeClasse<Tipo1, Tipo2, ...> n = new NomeClasse<>();
```

- do Java 7 em diante, não é necessário repetir os parâmetros de tipo do lado direito da atribuição;
- usa-se `<>` (válido para quaisquer quantidade de parâmetros de tipo) e
- a ausência do parâmetro de tipo em uma classe genérica implica na utilização do tipo `Object` como *default*.

Classe Que Armazena Uma String

```
public class Dado {
  private String dado;
  public Dado(String d) {
    dado = d;
  }
  public String getDado() {
    return dado;
  }
}
...
Dado d = new Dado(“Um texto qualquer.”);
String x = d.getDado();
...
```

Classe Que Armazena Qualquer Tipo

```
public class Dado<E> {
  private E dado;
  public Dado(E d) {
    dado = d;
  }
  public E getDado() {
    return dado;
  }
}
...
Dado<String> d = new Dado<>(“Ola”);
String x = d.getDado();
...
Dado<Pessoa> d = new Dado<>(new Pessoa(“Carla”,20));
Pessoa x = d.getDado();
...
```

Considerações Sobre Genéricos

- Java não cria um tipo específico para cada instância de uma estrutura genérica.

- Durante a compilação as anotações entre < e > são apagadas e ocorre uma substituição para código Java tradicional com os tipos e *casts* adequados.

Exemplo 01 - Produto

- Exemplo Produto

Restrições

- É possível criar genéricos limitados a uma certa "família" de classes;
- Exemplo:

```
public class MinhaLista<E extends Produto> { ... }
```

- o exemplo define uma classe `MinhaLista` que pode conter quaisquer elementos cujo tipo seja subclasse ou implementação de `Produto`;
- este tipo de restrição é chamado de "limite superior" (*upper bounds*);
- não importando se `Produto` é uma classe ou interface usa-se a palavra reservada `extends`.

Restrições 2

- É possível criar também restrições de limite inferior (*lower bounds*) que são de utilização mais rara

- Exemplo:

```
public class MinhaLista<E> {
    ...
    public void metodo(List<? super Enlatado> li) {
        ...
    }
}
```

- o exemplo apresenta uma lista cujos elementos devem ser `Enlatado` ou superclasses de `Enlatado`.

- Exemplo: se `Enlatado` é derivado de `Produto`, então `Produto` é um tipo de elemento aceito na coleção.

Bibliografia

- COHEN, M. **Genéricos em Java**. Disponível em: <https://www.inf.pucrs.br/flash/alpro2/present/U03_projeto/01b-genericos/handout.html>. Acesso em: 18 set. 2023.
- DEVMEDIA. Usando Generics em Java. Disponível em <https://www.devmedia.com.br/usando-generics-em-java/28981>. Acesso em: 21 set. 2023.