Generics em Java

1

Generics

- *Generics* são parâmetros de tipo, passados para uma classe, que permitem checagem de tipo em tempo de compilação;
- A programação com generics possibilita a escrita de códigos que podem ser reutilizados por objetos de vários tipos diferentes;
- O uso de generics auxilia na diminuição dos erros de casting durante a execução do programa;
  - Ex: `Pessal pessoa = (Pessoa) new Carro();`
- São interessantes porque deixam o código mais seguro e fácil de se entender.

2

2

Generics

- Uma classe com generics inclui uma variável de tipo entre "<" ">" depois do nome da classe;
  - Ex: `public class Dado<T> { ... }`
- Uma classe com generics pode ter mais de uma variável de tipo;
- As variáveis de tipo são usadas para especificar:
  - Os tipos de retorno dos métodos;
  - Os tipos dos atributos;
  - Os tipos das variáveis locais.

3

3

Generics – Exemplo

```
public class Par<T>
{
   private T primero = null;
   private T Segundo = null;

   public Par() {
   }
   public Par(T primeiro, T segundo) {
       this.primeiro = primeiro;
       this.segundo = segundo;
   }
   public T getPrimeiro() {
       return primeiro;
   }
   public T getSegundo() {
       return segundo;
   }
   public void setPrimeiro(T primeiro) {
        this.primeiro = primeiro;
   }
   public void setSegundo(T segundo) {
        this.segundo = segundo;
   }
}
```

4

4

Generics

- Como convenção de código, usam-se letras maiúsculas para definir as variáveis de tipo.
  - Ex: E, T, U, V, K
- No momento da declaração e instanciação do objeto o tipo generics as variáveis de tipo devem ser substituídas por tipo de objetos existentes.
  - Ex: `Par<String> parStrings = new Par<String>();`

5

5

Generics – Restrições

- **Tipo simples** não podem ser utilizados como tipos para os objetos a serem instanciados;
  - `Par<int> umPar = new Par<int>(); // ERRO`
- Não se pode lançar ou tratar **exceções** de objeto de classe generics;
- Não se pode declarar **arrays** de tipos parametrizados.
  - `Par<String>[] table = new Par<String>[10]; // ERRO`
- As variáveis de tipo generics devem ser sempre substituídas na instanciação do objeto que as contêm;
- Não se pode referenciar uma variável de tipo em um método ou atributo estático.

6

6

Classes Wrapper

- Tipos simples não podem ser utilizados para instanciar objetos generics;
- Para se instanciar objetos generics de tipos simples é necessário utilizar seus respectivos tipos de referência (wrappers);
- Existem um tipo referência (wrapper) para cada tipo simples da linguagem;
- Além de representar os tipos simples, eles têm métodos utilitários, tais como: os conversores de tipo, toString(), equals().

Classes Wrapper

- Exemplos de wrappers para tipos simples:

`Byte`	**`byte`**
`Integer`	**`int`**
`Boolean`	**`boolean`**
`Character`	**`char`**
`Float`	**`float`**
`Double`	**`double`**
`Short`	**`short`**
`Long`	**`long`**

Classes Wrapper – Métodos Utilitários

```
public class WrappersTeste {

   public static void main(String[]
   args) {
        Integer i = 5;
        Double d = Double.valueOf(i) ;
        String s = String.valueOf(d);
        Float f = Float.parseFloat(s);
        System.out.println(i);
        System.out.println(d);
        System.out.println(s);
        System.out.println(f);
   }
}
```

```
Resultado:
5
5.0
5.0
5.0
```

9

9

Autoboxing

- Permite conversão de tipos simples para suas classes *wrapper* (e vice-versa) de forma automática.

```
public void teste1()
{
    List<Integer> lista =
       new ArrayList<Integer>();

    lista.add(new Integer(2));
    lista.add(new Integer(4));
    lista.add(new Integer(6));
    lista.add(new Integer(8));

    int n = lista.get(3).intValue();
}
```

Sem autoboxing

```
public void teste1()
{
    List<Integer> lista =
       new ArrayList<Integer>();

    lista.add(2);
    lista.add(4);
    lista.add(6);
    lista.add(8);

    int n = lista.get(3);
}
```

Com autoboxing

10

10