Abstrações e Funções de Alta Ordem - Polimorfismo Universal Paramétrico

Kleber Jacques F. de Souza

Polimorfismo Universal Paramétrico

- **Polimorfismo Universal Paramétrico**: É o polimorfismo que permite que você escreva código genérico para servir para os subtipos.
- Serve como Tipos abstratos de dados **parametrizados.**

Polimorfismo Universal Paramétrico

- É uma forma de se tornar uma linguagem mais expressiva.
- Mantém toda sua **tipagem estática segura**.
- O conceito de parâmetros de tipos é aplicado em C# por meio de **Generics.**

Generics

- Tornam possíveis a estruturação de classes e métodos que **adiam a especificação de um ou mais tipos** até que a classe ou método seja declarada e instanciada pelo código.

Generics

- Usando um **parâmetro de tipo genérico T** você pode escrever uma única classe que outro código poderá usar **sem aumentar o custo ou risco de conversões** (*cast*) em tempo de execução (*runtime*) ou operações de **boxing.**

Generics

- O **boxing** é um processo de conversão de um tipo de valor para o tipo object. É implícito!
- **Unboxing** extrai o tipo de valor do objeto. É explícito!

```
int i = 123;
object o = i; //BOXING
o = 123;
i = (int)o;  // UNBOXING
```

Generics

- Dentro da linguagem C#, existem 3 situações correspondentes à maneira como o compilador trata a conversão entre tipos genéricos (mais abstratos) e tipos mais específicos (mais especializados): **Covariância**, **Contravariância** e **Invariância**.

Covariância

- Trata-se da **conversão** de um **tipo especializado** (específico) para um **tipo mais genérico.**

```
string a = "Teste";
object obj1 = a;
```

Contravariância

- É a situação inversa à Covariância - um **tipo mais genérico** é convertido para um **tipo mais específico.**

```
object obj1 = 13;
int num = (int)obj1;
```

Invariância

- **Não há necessidade de conversão dos tipos**, pois se trata da mesma tipagem.

```
int numero = 13;
```

Generics

- O C#, em sua versão atual, consegue fazer estas operações em interfaces, *delegates* e tipos genéricos.
- Usamos tipos genéricos para maximizar **reutilização de código**, **segurança de tipo**, e **desempenho**.

Generics

```
public class TipoGenerico<T>{

        private T dado;

        public T getDado(){
            return dado;
        }
        public void setDado(T _dado){
            dado = _dado;
        }
}
```

Generics

```
static void Main()
{
        TipoGenerico<int> dadoInt = new TipoGenerico<int>();
        dadoInt.setDado(13);
        Console.WriteLine(dadoInt.getDado());

        TipoGenerico<string> dadoString = new TipoGenerico<string>();
        dadoString.setDado("Teste");
        Console.WriteLine(dadoString.getDado());
}
```

Referências Bibliográficas

Microsoft, 2018. **Generics**. Disponível em: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/