# RESUMO

## Herança

### Reutilização de Código

-Deve diminuir a repetição de código.
-A ideia é reaproveitar o máximo do código já criado.
-Essa ideia está diretamente relacionada ao conceito Don't Repeat Yourself (DRY).
-Em outras palavras, devemos minimizar ao máximo a utilização do "copiar e colar".
-O aumento da produtividade e a diminuição do custo de manutenção são as principais motivações do DRY.

### Uma classe para todos os serviços

-A classe genérica é denominada super classe, classe base ou classe mãe.
-As classes específicas são denominadas sub classes, classes derivadas ou classes filhas.

### Reescrita de Método

-Os métodos das classes específicas têm prioridade sobre os métodos das classes genéricas.
-Em outras palavras, se o método chamado existe na classe filha ele será chamado, caso contrário o método será procurado na classe mãe.
-Quando definimos um método com a mesma assinatura na classe base e em alguma classe derivada, estamos aplicando o conceito de Reescrita de Método.

### Fixo + Específico

-Dessa forma, quando o valor padrão do preço dos serviços é alterado, basta modificar o método na classe Servico.

```
1  class  Servico {
2    public double calculaTaxa() {
3      return  5 ;
4    }
5  }
```

*Código Java 8.16: Servico.java*

```
1  class Emprestimo extends Servico {
2    // ATRIBUTOS
3
4    public double calculaTaxa() {
5      return  super.calculaTaxa() + this.valor * 0.1;
6      }
7  }
```

*Código Java 8.17: Emprestimo.java*

# Polimorfismo

## É UM (extends)

-Objetos criados a partir das classes específicas sejam tratados como objetos da classe genérica.
-Aplicando a ideia do polimorfismo facilita manutenção da classe.

---

# Classes Abstratas

## Definição

-Classe abstrata faz restrição no código, que não pode ser instanciadas.
-A sub-classe é uma classe concreta pois criaremos objetos a partir dela.
-Para definir uma classe abstrata, basta adicionar o modificador abstract.
-O método da super classe seria reescrito nas classes específicas sem nem ser reaproveitado.