Sumário

[Versões de linguagem 2](#_Toc448831664)

[Premissas 2](#_Toc448831665)

[IDE Utilizada 2](#_Toc448831666)

[Dependências do projeto 2](#_Toc448831667)

[Decisões Arquiteturais 2](#_Toc448831668)

[Outras informações 3](#_Toc448831669)

# Versões de linguagem

O código desenvolvido está compatível a partir da versão da JDK 1.5.

# Premissas

Para configurar o projeto em sua IDE de desenvolvimento, será necessário possuir o Mavem. Responsável por gerir as dependências do projeto.

# IDE Utilizada

Para o desenvolvimento do projeto foi utilizada a IDE Eclipse Mars. Porém, o projeto pode ser configurado em qualquer IDE de sua preferência.

# Dependências do projeto

Seguindo as orientações passadas, foi utilizado o mínimo possível de utilitários focando somente no desenvolvimento da solução do problema.

Para o desenvolvimento do projeto em questão foi necessário a utilização das seguintes dependências:

* **Apache-commons-lang (3.0):** utilitário para geração de equals e hash code do modelo;
* **JUnit (4.11):** utilitário para realização dos testes unitários;
* **HsqlDB (1.8.0.10):** utilitário que realiza a criação de um servidor de banco de dados;

# Decisões Arquiteturais

Seguindo as recomendações da prova foi decidido que:

* Para construir um objeto Transferência, foi imposto a necessidade de utilização do design pattern Builder fazendo com que a classe “TransferênciaBuilder” seja necessária para a criação do objeto. Assim esta classe se torna responsável pela validação dos atributos e cálculo da taxa específica em questão;
* Como foi utilizado um servidor de banco de dados, foi utilizado o design pattern de DAO visando agrupar todas as operações com o banco de dados em questão;
* Para fazer o desenvolvimento do cálculo das taxas foi utilizado o design pattern de template method abstraindo algumas validações e fazendo com que a classe específica de um determinado tipo de taxa fique com foco apenas no desenvolvimento da lógica do cálculo da taxa;
* Como nos requisitos do problema está descrito que não é necessário produzir uma aplicação funcional, apenas a implementação do problema proposto, esta implementação foi desenvolvido na classe “AgendamentoTransferenciaService” onde disponibiliza exatamente os serviço solicitados pelo enunciado que é o de agendamento da transferência e a consulta as transferências agendadas.
* Foi utilizado a metodologia TDD. No pacote específico para testes estão todas as classes de testes contendo todos os cenários que consegui pensar neste curto período de tempo.

# Outras informações

O tempo total de desenvolvimento da prova foi de aproximadamente 12 horas.

Além das informações neste documento. Há algumas observações no comentário de alguns métodos e/ou classes.