Comando (padrão de desenho)/Banco, Conta, Titular (aplicação bancária)

*From Wiki**3*

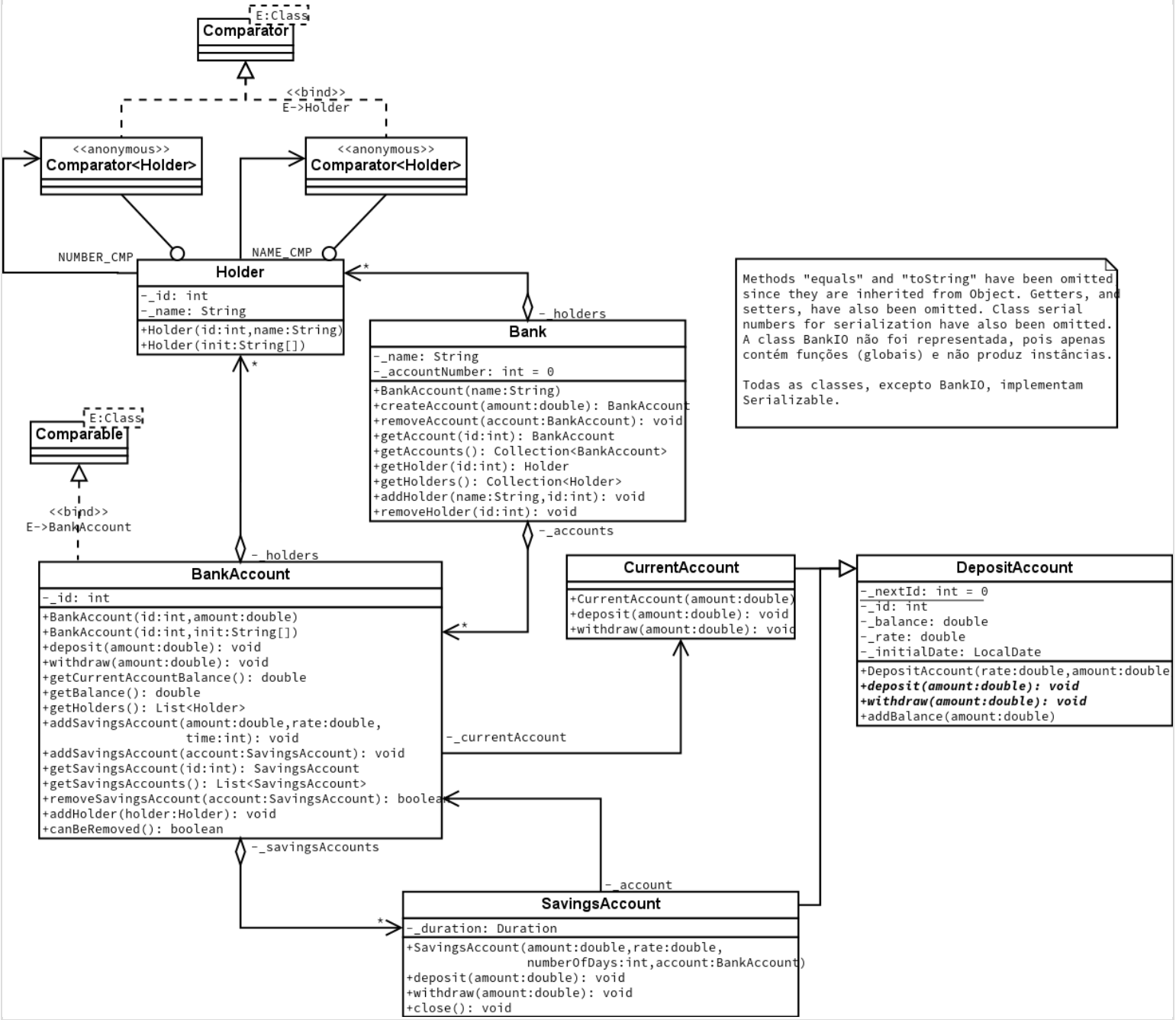
< Comando (padrão de desenho)

Contents

A aplicação bancária representa a funcionalidades das classes Dialog, Menu e Command.

A package principal ("core")

Diagrama UML do "core" da aplicação bancária (versão antiga) **[Collapse]**



Um banco tem zero ou mais contas e zero ou mais titulares e é identificado univocamente pelo seu nome. A classe Bank deve disponibilizar métodos que permitam guardar/remover contas e titulares. Do ponto de vista de um banco, tanto as contas como os titulares são identificados por um número. Quando é iniciada a aplicação bancária e se pretende criar um banco novo, é necessário

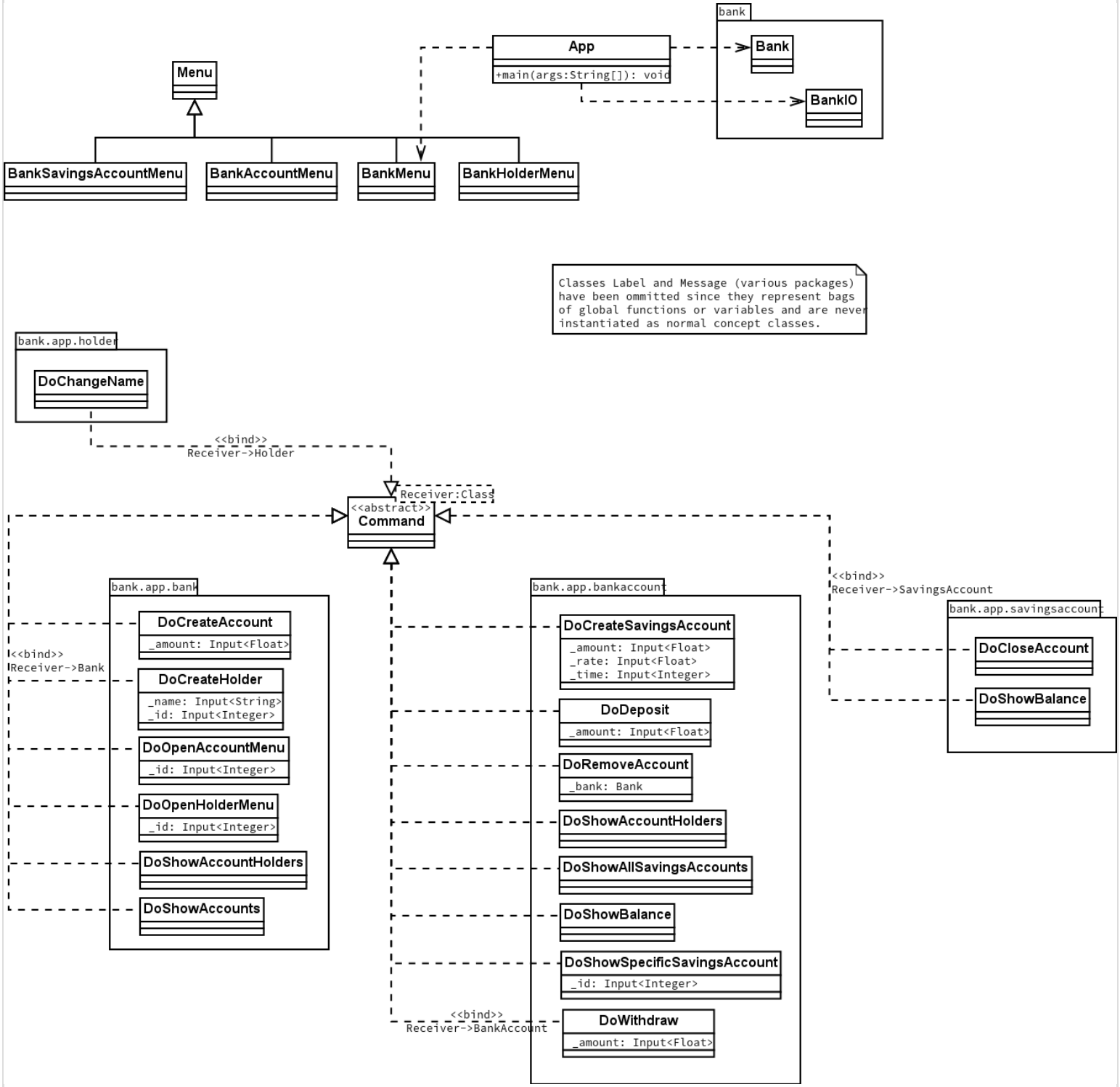
indicar o nome do banco que se pretende criar. Adicionalmente, as funcionalidades que um banco deve disponibilizar são as seguintes:

- Criação de contas: recebe o valor do saldo inicial e devolve o número da nova conta.
- Remoção de contas: recebe o número da conta a apagar. A conta só é apagada se o seu saldo total for igual a zero. Caso não seja possível apagar a conta, é apresentada uma mensagem justificativa.
- Criação de titulares: recebe o nome do titular e número de identificação.
- Remoção de titulares: recebe o número de identificação do titular.

As classes **Bank**, **SavingsAccount**, **BankAccount** e **Client** devem disponibilizar as operações necessárias para o funcionamento da aplicação.

A package de interacção ("app")

Diagrama UML das packages de interacção (versão antiga) **[Collapse]**



A interacção deve permitir o acesso à funcionalidade do "core". Cada operação vai estar encapsulada num comando (subclasse de **Command**) que tem como "receiver" uma entidade do "core". As instâncias dos comandos são agrupadas em menus.

Esta funcionalidade (de mostrar as operações disponíveis para uma dada entidade e executar a opção escolhida pelo utilizador) deve ser concretizada pelas classes **Menu** e **Command**. A classe **Command** representa uma operação genérica que pode ser executada sobre uma dada entidade. Cada comando sabe o título da operação que representa (é utilizado no menu para indicar os títulos das operações disponíveis). Esta classe abstracta define o método abstracto **execute** que executa a operação. Cada operação a disponibilizar num menu terá que ser concretizada numa sua subclasse. A classe **Menu** recebe um vector de comandos e mostra os títulos destes comandos por forma a que os utilizadores saibam as operações disponíveis. Cabe ainda ao menu executar o comando correspondente à opção escolhida pelo utilizador.

- Criar um titular;
- Criar uma conta;
- Listar todos os titulares ordenados por ordem crescente de nome ou de número;
- Listar todas as contas;
- Aceder às operações de um determinado titular;
- Aceder às operações de uma determinada conta;
- Alterar a data actual.

Por forma a centralizar (e assim permitir a reutilização de código) a interface com o utilizador, existe uma classe, chamada **Dialog**, que é responsável pela interacção com o utilizador.

A leitura de dados do utilizador é realizada a partir de formulários (instâncias da classe **Form**), existindo um por comando. Esta classe agrega os campos de entrada (classe **Input** e especializações), concentrando todas as operações de leitura de dados (fornecidos pelo utilizador). Os dados lidos são validados, i.e., apenas são aceites dados correctos. No caso do utilizador inserir dados inválidos, é pedida a respectiva reintrodução.

A escrita de dados é realizada através da classe **Display**, existindo uma instância por comando. Esta classe permite escrever o resultado da execução da aplicação num ficheiro em vez de escrever no standard output. O nome destes ficheiros são indicados através das propriedades in e out . Caso não sejam indicadas, são utilizados os valores por omissão (standard input e standard output).

Código e Exercícios

A estrutura desta aplicação é semelhante à do projecto a desenvolver.

- po-uilib-202209081626.tar.bz2 - Base interaction package, a.k.a. **po-uilib**
- po-bank-core-202109221024.tar.bz2 - Core functionality, a.k.a. **bank-core**
- po-bank-app-202109221024.tar.bz2 - Bank application and corresponding multimodal user interface, a.k.a. **bank-app**

Each package includes a makefile to compile and build the corresponding JAR archive. Note that the makefiles assume that all JAR dependencies reside in /usr/share/java (linux installation -- other environments may, and indeed should work, but are not covered in this text).

Ver com atenção o código da aplicacão bancária (acima) já com a introdução da interacção com o utilizador.

- Terminar o comando **DoChangeName**.
- Alterar o comando **DoShowAccounts**, por forma a escrever uma conta por linha, em que os dois primeiros caracteres da linha são dois caracteres '\t' e os restantes são os retornados pelo método **toString** da classe **BankAccount**.
- Finalmente, acrescentar duas novas funcionalidades à aplicação bancária:
  - A primeira permite adicionar um titular a uma conta;
  - A segunda permite a remoção de um titular, com a ressalva de que se uma conta só tiver um titular então esse titular não poderá ser removido.

⚠ A partir de 2020, é necessário usar Java 14 ou superior (recomendado 17 ou superior).

Como executar a aplicação

Assumindo que a aplicação já está disponível nos vários JARs (**po-bank-core**, **po-bank-app**, **po-uilib**), para a execução, basta definir a variável **CLASSPATH** apropriadamente:

```
export CLASSPATH=/caminho/para/po-bank-core.jar:/caminho/para/po-bank-app.jar:/caminho/para/po-uilib.jar
```

Seguidamente, basta invocar o método **main**:

```
java bank.app.App BPO
```

Neste comando, "BPO" é simplesmente o argumento do programa principal, sendo específico desta aplicação.

Alternativamente (embora menos legível e menos conveniente), pode ser indicado o valor de **CLASSPATH** directamente ao comando que executa a aplicação (opção **-cp**):

```
java -cp /caminho/para/po-bank-core.jar:/caminho/para/po-bank-app.jar:/caminho/para/po-uilib.jar bank.app.App BPO
```

Note-se que, neste último comando, a classe que contém o método **main** a executar deve vir depois das opções do comando **java** (que executa a máquina virtual do Java).

Ver também

A seguinte página descreve o padrão Command (Comando), que está subjacente à estrutura da interacção textual da aplicação.

- Comando (padrão de desenho)

Categories: **Ensino** **PO** **PO Exemplos** **Java**