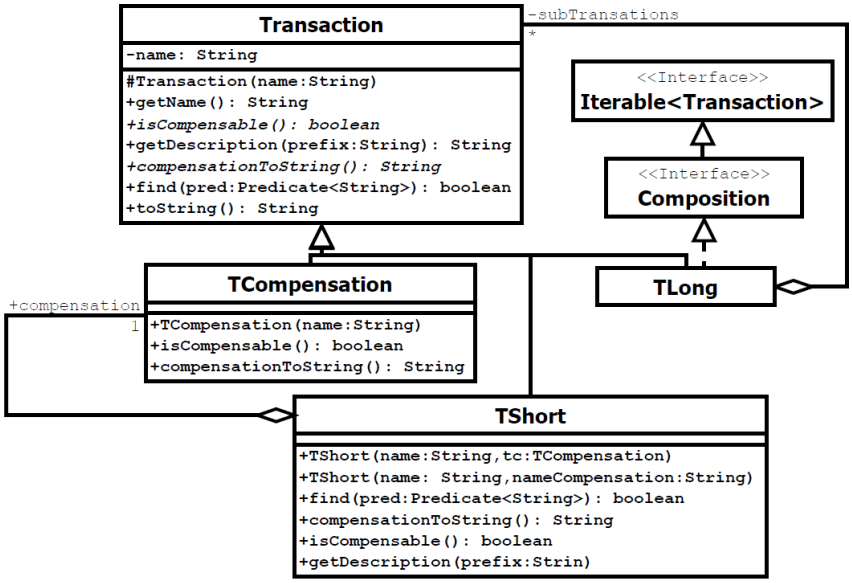
Instituto Superior de Engenharia de Lisboa
Programação III
1º Teste – 25 de Novembro de 2021 (Duração: 1:00)

ISEL ADEETC

Pretende-se implementar uma solução para a descrição de modelos de composição de transações. Considere por exemplo a atividade de uma agência de viagens: a compra de uma viagem implica marcação de hotel e de voo; a marcação, por sua vez, implica reserva e pagamento; a desistência da viagem implica o cancelamento das reservas e crédito da importância paga. Neste exemplo identificam-se as seguintes transações:

- a transação longa **`TLong`** (viagem) que é composta por um conjunto de sub-transacções (marcação de hotel e voo) que, por sua vez, também são compostas por um conjunto de sub-transacções (reservas e pagamentos);
- a transação curta **`TShort`** (reserva e pagamento) tem associada uma transação que a compensa, isto é, que reverte os seus efeitos (cancelamento e devolução respetivamente);
- a transação de compensação **`TCompensation`**, que para efeitos de simplificação se assume não estar associada a qualquer outra compensação, isto é, não é compensável.

Para o efeito chegou-se ao seguinte diagrama de classes.



Tendo em conta o diagrama estático de classes, os troços de código e os respetivos output:

1. [3] Defina a classe abstrata **`Transaction`**. Ao construtor é passado o nome. O método **`getName`** retorna o nome recebido por parâmetro no construtor e não pode ser redefinido. Os métodos **`isCompensable`** e **`compensationToString`** são abstratos. O método **`getDescription`** retorna o prefixo concatenado com o nome entre aspas. Os métodos **`find`** e **`toString`** têm a seguinte implementação e não necessitam de ser transcritos para o teste:

```
public boolean find(Predicate<String> pred){ return pred.test(name); }
public final String toString()              { return  name; }
```

2. [2] Defina a classe **`TCompensation`**. Ao construtor é passado o nome. Uma transação de compensação não é compensável. O método **`compensationToString`** lança a exceção de *runtime* **`UnsupportedOperationException`**. Exemplo de transação de compensação:

```
TCompensation cancel = new TCompensation("Cancelar voo");
System.out.println( cancel.getDescription("->") );
System.out.println( cancel.isCompensable());
```

```
->"Cancelar voo"
false
```

3. [3] Defina a classe **`TShort`** que representa uma transação curta. Uma transação curta tem sempre associada uma transação de compensação. No construtor recebe por parâmetro o nome da transação e uma instância de **`TConpensation`**, ou o nome da transação e o nome da compensação. Uma transação curta é compensável. O campo público **`compensation`** só pode ser afetado no construtor. O método **`compensationToString`** retorna a descrição da transação que a compensa. O método **`getDescription`** acrescenta á descrição herdada a descrição da compensação. O método **`find`** retorna **`true`** se o nome da transação ou o nome da compensação obedecer ao predicado, `false` caso contrário. Exemplo de transações curtas:

```
TShort book=new TShort("Reservar voo",cancel);
System.out.println( book ) ;
System.out.println( book.getDescription(" >>" ));
System.out.println( book.isCompensable());
```

```
Reservar voo
  >>"Reservar voo" - "Cancelar voo"
true
```

```
System.out.println( book.find( s -> s.contains("Reserva")));
System.out.println( book.find( s -> s.contains("Cancela")));
```

```
true
true
```

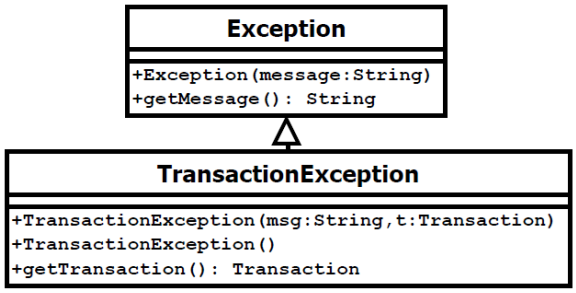
```
TShort pay = new TShort("Pagamento voo", "Devolver pagamento voo");
System.out.println( pay.getDescription(""));
System.out.println( pay.compensation ) ;
```

```
"Pagamento voo" - "Devolver pagamento voo"
Devolver pagamento voo
```

4. [2] Defina a classe **`TransactionException`** para que o método **`getMessage`** herdado de **`Exception`** retorne:

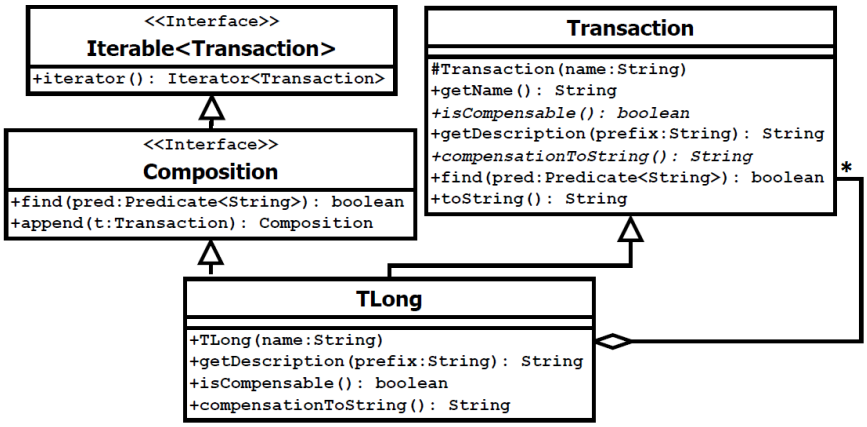
- A *string* "*Invalid transaction*", caso tenha sido instanciado com o construtor sem parâmetros;
- A *string* passado por parâmetro no construtor caso tenha sido instanciado com o construtor com dois parâmetros

O método **`getTransaction`** retorna a **`Transaction`** passado por parâmetro no construtor.



5. [2] Defina a interface **`Composition`**. O método **`append`** pode lançar a exceção **`TransactionException`**.

6. [8] Defina a classe **`TLong`**, que representa uma transação compensável, tendo em conta que:



- O método **`iterator`** retorna um **`Iterator`** para as sub-transações.
- O método **`append`** adiciona a transação ou lança a exceção **`TransactionException`** caso a transação a adicionar não seja compensável ou já exista uma transação com o mesmo nome (usar o método **`find`**). Retorna a própria transação.

```
TLong travel = new TLong("Viagem");
try {
  TLong flight= new TLong("Marcar voo");
  TShort book = new TShort("Reservar voo", "Cancelar voo"),
         pay = new TShort("Pagamento voo", "Devolver pagamento voo");
  flight.append(book).append( pay );
  TLong hotel = new TLong("Marcar hotel");
  hotel.append( new TShort("Reservar hotel", "Cancelar hotel") )
       .append( new TShort("Pagamento estadia","Devolver pagamento estadia"));

  travel.append( flight ).append( hotel );
  travel.append( book);
}
catch ( TransactionException e ) {
   System.out.println( e.getMessage() );
   System.out.println( e.getTransaction() );
}
```

```
Duplicate transaction
Reservar voo
```

- O método **`find`** retorna **`true`** se encontrar uma sub-transação que obedeça ao predicado (ver o exemplo):

```
if( ! travel.find(s-> s.equals("Viagem")) &&
     travel.find(s-> s.equals("Reservar hotel")) &&
     travel.find(s-> s.equals("Cancelar voo")) ) System.out.println( "Find is CORRECT" );
```

- O método **`getDescription`** retorna uma *string* com a descrição pormenorizada da transação (ver *output*):

```
System.out.println( travel.getDescription("-> ") );
```

```
-> "Viagem"
  -> "Marcar voo"
    -> "Reservar voo" - "Cancelar voo"
    -> "Pagamento voo" - "Devolver pagamento voo"
  -> "Marcar hotel"
    -> "Reserva hotel" - "Cancelar hotel"
    -> "Pagamento estadia" - "Devolver pagamento estadia"
```

- O método **`compensationToString`** retorna uma *string* com a concatenação das descrições das compensações das sub-transacções separadas pelo carácter ponto e virgula (';').

```
System.out.println( travel.compensationToString() );
```

```
"Cancelar voo"; "Devolver pagamento voo"; "Cancelar hotel"; "Devolver pagamento estadia"
```