Primeira Prova de MAB 240 — Computação II

Fabio Mascarenhas

20 de Abril de 2012

A prova é individual e sem consulta. Responda as questões na folha de respostas, a lápis ou a caneta. Se tiver qualquer dúvida consulte o professor.

Nome: _____

DRE: _____

Questão:	1	2	3	4	5	6	7	Total
Pontos:	½	2	2	1½	2	1	1	10
Nota:								

Um programa assistente financeiro controla as contas bancárias de determinada pessoa. Cada conta tem um número e o nome do correntista, e existem três tipos de conta: contas correntes, contas poupança, e certificados de depósito (CDs). Contas correntes também têm a informação do saldo atual e do limite mínimo, contas poupança têm o saldo atual e a taxa de juros anual, e CDs o valor depositado, a taxa de juros anual, e a data de vencimento. Podemos representar esse problema com as seguintes classes:

```
interface Conta { }

class ContaCorrente implements Conta {
  private int numero;
  private String correntista;
  private double saldo;
  private double limite;

  public ContaCorrente(int numero,
       String correntista,
       double saldo, double limite) {
    this.numero = numero;
    this.correntista = correntista;
    this.saldo = saldo;
    this.limite = limite;
  }
}
```

```
class Poupanca implements Conta {
  private int numero;
  private String correntista;
  private double saldo;
  private double taxaJuros;

  public Poupanca(int numero,
       String correntista,
       double saldo, double taxaJuros) {
    this.numero = numero;
    this.correntista = correntista;
    this.saldo = saldo;
    this.taxaJuros = taxaJuros;
  }
}
```

```
class CD implementa Conta {
  private int numero;
  private String correntista;
  private double saldo;
  private double taxaJuros;
  private String dataVencimento; // AAAA-MM-DD

  public CD(int numero, String correntista, double saldo, double taxaJuros,
            String dataVencimento) {
    this.numero = numero;
    this.correntista = correntista;
    this.saldo = saldo;
    this.taxaJuros = taxaJuros;
    this.dataVencimento = dataVencimento
  }
}
```

1. (½ ponto) Mostre como criar instâncias de cada uma dessas três classes.

2. (2 pontos) Essas classes têm várias informações em comum, o que é uma indicação de podemos abstrair as partes em comum usando classes abstratas e herança. Transforme `Conta` em uma classe abstrata com as partes em comum de todos tipos de conta, e crie mais uma classe abstrata `Investimento` que deriva de conta e tem as partes em comum entre as contas que têm correção monetária. Ajuste os construtores para aproveitar os construtores das superclasses.

3. (2 pontos) Adicione a `Conta` os métodos `void deposita(double valor)` e `void retira(double valor)`. O funcionamento desses métodos varia de acordo com o tipo de conta: o método `deposita` aumenta o saldo das contas correntes e poupança, mas nos certificados de depósito deve sinalizar uma exceção `RuntimeException` com a mensagem *"tipo de conta não aceita depósito"*. O método `retira` também sempre sinaliza uma exceção nos certificados de depósito, também do tipo `RuntimeException` com a mensagem *"tipo de conta não aceita retirada"*. Nas contas poupança uma tentativa de retirada de valor maior que o saldo atual falha e sinaliza uma `RuntimeException` com a mensagem *"saldo insuficiente"*. Nas contas corrente uma retirada falha se o valor for maior que o saldo atual mais o limite. Faça implementações nas superclasses quando possível.

4. (1½ pontos) Adicione a `Investimento` o método `void corrige()`, responsável por calcular e aplicar a correção monetária de um dia. Para obter a taxa de um dia basta dividir a taxa anual por 365. Um certificado de depósito só deve ser corrigido se a data da correção é menor que a data de vencimento. Assuma que existe uma função `Data.hoje()` que retorna uma *string* com a data de hoje. Implemente o método em `CD` usando o método da superclasse.

5. (2 pontos) Adicione um registro de lançamentos à classe `Conta`. Um lançamento tem uma data, uma descrição e um valor. Depósitos, retiradas e correções devem ser registrados com a descrição *"DEPÓSITO"*, *"RETIRADA"* e *"CORREÇÃO MONETÁRIA"*, respectivamente. Use a função `Data.hoje()`, e lembre que a interface `List` de Java tem um método `add` que acescenta um novo item no final da lista.

6. (1 ponto) Adicione um método `List<Lancamento> extrato(String dataInicio, String dataFinal)` em `Conta`, que retorna todos os lançamentos até entre essas datas (inclusive). [1,0]

7. (1 ponto) Adicione um método `double movimento(String dataInicio, String dataFinal)` em `Conta`, que dá o valor movimentado nos lançamentos entre essas datas (inclusive), usando o método `extrato` da questão anterior.