Primeira Prova de MAB 225 — Computação II

Fabio Mascarenhas

20 de Maio de 2015

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	Total
Pontos:	6	4	10
Nota:			

1. Um programa assistente financeiro gerencia as contas bancárias e investimentos de uma pessoa. Ele classifica as entidades que ele gerencia em duas interfaces: a interface `Conta` representa contas que aceitam depósitos e retiradas, e a interface `Investimento` representa algo cujo saldo é corrigido regularmente de acordo com determinada taxa diária de juros. A versão inicial do programa define três classes concretas para representar contas e investimentos: a classe `ContaCorrente` implementa a interface `Conta`, a classe `Poupanca` implementa tanto a interface `Conta` quanto `Investimento`, e a classe `CD` (de certificado de depósito) implementa apenas a interface `Investimento`. As três classes têm as definições a seguir (o número é um inteiro, o correntista é uma string, o saldo, limite e a taxa são decimais, e o vencimento é uma string como "2015-05-18"):

```
class ContaCorrente:
    def __init__(self, numero, correntista,
                 saldo, limite):
        self.numero = numero
        self.correntista = correntista
        self.saldo = saldo
        self.limite = limite
```

```
class Poupanca:
    def __init__(self, numero, correntista,
                 saldo):
        self.numero = numero
        self.correntista = correntista
        self.saldo = saldo
        self.taxa = 0.016
```

```
class CD:
    def __init__(self, numero, correntista, saldo, taxa, vencimento):
        self.numero = numero
        self.correntista = correntista
        self.saldo = saldo
        self.taxa = taxa
        self.vencimento = vencimento
```

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

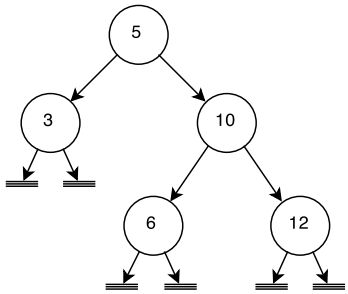
(b) (1 ponto) Suponha que a interface `Conta` possui um método `deposita(self, valor)` para depósito na conta, aumentando seu saldo e não retornando nada. Diga em qual(is) da(s) classe(s) acima esse método deve ser implementado, e dê sua(s) implementação(ões).

(c) (2 pontos) Suponha que a interface `Conta` possui um método `retira(self, valor)` para retiradas, que diminui o saldo da conta e retorna `True` se a retirada poder ser feita, deixando o saldo inalterado e retornando `False` em caso contrário. Não se pode retirar de uma conta corrente mais do que o saldo atual somado ao limite, e não se pode retirar da poupança mais que seu saldo atual. Implemente o método `retira`.

(d) (2 pontos) Suponha que a interface `Investimento` possui um método `corrige(self)`, responsável por calcular e aplicar a correção dos juros de um dia. Um certificado de depósito só deve ser corrigido se a data de hoje é menor que a data de vencimento (assuma que existe uma função `hoje()` que retorna a data de hoje). Implemente o método `corrige`.

2. Uma *árvore binária* é uma das estruturas de dados mais usadas em computação. Podemos representar árvores de de maneira recursiva, de modo similar à que usamos para representar listas em sala de aula. Uma árvore pode ser uma instância da classe `ArvoreVazia`, ou uma instância da classe `ArvoreNodo`, que contém três partes: o *elemento* que encabeça aquela árvore, a árvore que está *à esquerda* e a árvore que está *à direita*. Um exemplo de árvore com elementos inteiros é dado pela expressão e pelo diagrama a seguir (no diagrama os círculos representam instâncias de `ArvoreNodo` e os traços instâncias de `ArvoreVazia`):

```
arv = ArvoreNodo(5,
                 ArvoreNodo(3,
                            ArvoreVazia(),
                            ArvoreVazia()),
                 ArvoreNodo(10,
                            ArvoreNodo(6,
                                       ArvoreVazia(),
                                       ArvoreVazia()),
                            ArvoreNodo(12,
                                       ArvoreVazia(),
                                       ArvoreVazia())))
```



(a) (2 pontos) Implemente as classes `ArvoreVazia` e `ArvoreNodo`, dando seus construtores e um método `tamanho(self)`, que retorna o número de elementos da árvore (uma árvore vazia tem 0 elementos, e `arv` do exemplo acima tem 5 elementos). Use recursão estrutural para implementar `tamanho`.

(b) (2 pontos) Implemente o método `percorrer(self, acao)` para árvores binárias. Esse método executa `acao` em cada elemento da árvore, onde `acao` é um objeto que possui um método `executa(self, elem)`, que recebe o elemento e não retorna nada. Também implemente a classe `AcaoLista`, que é uma ação que acumula todos os elementos passados para ela em uma lista criada vazia no seu construtor.

BOA SORTE!