Primeira Prova de MAB 240 — Computação II

Fabio Mascarenhas

28 de Março de 2014

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:	7	3	10
Nota:			

1. Um *iterador* é uma maneira de se percorrer uma sequência de elementos. Um objeto iterador possui dois métodos: um produz os elementos da sequência um por um, dando um erro se a sequência já acabou, e o outro diz se já se ainda há mais elementos a serem produzidos. Como exemplo, iteradores para sequências de inteiros podem ser modelados pela seguinte interface:

```
interface Iterador {
  int proximo();
  boolean final();
}
```

   (a) (2 pontos) Escreva a classe `Naturais` que implementa `Iterador` e representa uma sequência de números naturais começando em um natural n passado ao construtor. A sequência é infinita. Um exemplo de uso dessa classe:

```
Iterador nats = new Naturais(1);
// Imprime os números de 1 a 10
for(int i = 0; i < 10; i++)
  System.out.println(nats.proximo());
// Imprime 11
System.out.println(nats.proximo());
```

   (b) (2 pontos) Escreva o corpo da função abaixo, que recebe um iterador e retorna uma lista com todos os elementos que ele pode produzir:

```
static ArrayList<Integer> listaDeEnum(Iterador e) {
  // corpo
}
```

(c) (2 pontos) Escreva a classe `IteradorFiltro` que implementa `Iterador` e filtra uma outra sequência de acordo com um filtro, ambos passados no construtor. A sequência filtrada só produz elementos que passam pelo filtro. Um exemplo de uso:

```
Naturais nats = new Naturais(1);
Filtro impar = new Filtro() {
  public boolean aplica(int x) { return x % 2 == 1; }
};
Iterador impares = new IteradorFiltro(nats, impar);
// Imprime 1, 3, 5, 7, 9
for(int i = 0; i < 5; i++)
  System.out.println(impares.proximo());
```

(d) (1 ponto) Dê a definição da interface `Filtro` usada na questão anterior.

2. Podemos representar listas de strings usando a interface abaixo:

```
interface Lista {
  int tamanho();
  Lista adiciona(String s);
  boolean contem(String s);
}
```

O método `tamanho` retorna o número de elementos de uma lista. O método `adiciona` cria uma nova lista onde `s` é o primeiro elemento, e o resto da lista são os elementos da lista corrente. O método `contém` retorna `true` se `s` é um elemento da lista e `false` caso contrário.

(a) (2 pontos) Defina a classe `ListaCons` que implementa `Lista` e possui dois campos: o primeiro elemento da lista (um inteiro) e o resto da lista (uma instância de `Lista`). Implemente o construtor dessa classe e seus métodos.

(b) (1 ponto) Defina a classe `ListaVazia` que implementa `Lista` e representa uma lista sem elementos. Implemente seus métodos (dica: use `ListaCons` na implementação de `adiciona`).