Segunda Prova de MAB 240 2012.2 — Computação II

Fabio Mascarenhas

1o. de Março de 2013

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. A interface parametrizada abaixo representa funções de uma variável, quaisquer que sejam seus tipos de entrada e de saída:

```
public interface Funcao<X,Y> {
  Y aplica(X x);
}
```

   (a) (2 pontos) Escreva a classe `Explode` que implementa `Funcao` e representa uma função que recebe uma string e retorna uma lista de todos os caracteres da string, em sequência. Dicas: o método `charAt(i)` da classe `String` retorna o caractere na i-ésima posição da string, o atributo `length` contém o tamanho da string, e o tipo de um caractere é `Character`. Um exemplo de uso dessa classe:

```
Explode exp = new Explode();
List<Character> cs = exp.aplica("olá mundo");
System.out.println(cs.size()); // imprime 9
System.out.println(cs.get(2)); // imprime á
```

   (b) (2 pontos) Escreva a classe `ContaCaractere` que implementa `Funcao` e representa uma função que recebe uma lista de caracteres e retorna quantas ocorrências de determinado caractere essa lista tem (o caractere sendo contado é passado no construtor). Exemplo (continuando o anterior):

```
ContaCaractere cc = new ContaCaractere('o');
Integer n = cc.aplica(cs);
System.out.println(n);          // imprime 2
```

   (c) (2 pontos) Escreva a classe parametrizada `Composicao<X,Z>` que representa a composição de duas funções `f` e `g`. Aplicar a composta de `f` e `g` a algum `x` equivale a aplicar `f` a `x`, depois aplicar `g` ao resultado. Exemplo (novamente, continuando o anterior):

```
Composicao<String,Integer> comp =
    new Composicao<String,Integer>(exp, cc);
Integer n = comp.aplica("onomatopéia");
System.out.println(n);        // imprime 3
```

2. Um *filtro* de linha de comando é um programa que lê linhas da entrada padrão, faz alguma espécie de processamento com elas, e escreve sua saída na saída padrão. Muitos dos aplicativos em um sistema *Linux* podem ser usados como filtros, alguns bastante complexos.

   A classe a seguir modela um tipo de filtro bastante simples, que força cada linha da entrada a ter uma linha de saída (que tipo de processamento é feito com a linha fica a cargo da subclasse de `Filtro`):

```
public abstract class Filtro {
  private BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

  public void filtra() throws IOException {
    String linha = br.readLine();
    while(linha != null) {
      System.out.println(processa(linha));
      linha = br.readLine();
    }
  }

  protected abstract String processa(String linha);
}
```

   (a) (2 pontos) Crie uma subclasse de `Filtro` chamada `FiltroAcumulador`, que assume que o conteúdo de cada linha é um número decimal, mantendo um somatório dos números lidos e substituindo cada linha pelo valor atual do somatório (depois de adicionar a linha corrente). Por exemplo, dada a sequência "1 3 5 7 9", com um número em cada linha, o filtro produz "1 4 9 16 25". Dica: a função `Double.parseDouble` converte uma string em um `double`.

   (b) (2 pontos) Crie uma subclasse abstrata de `Filtro` chamada `FiltroRecupera` que redefine `filtra` para capturar exceções que aconteçam durante a filtragem, recomeçando a filtragem caso a exceção não seja do tipo `IOException` após imprimir o conteúdo da mensagem da mesma (usando o método `getMessage`). Caso a exceção seja do tipo `IOException` ela deve ser passada adiante.

BOA SORTE!