PMR2300 - Computação para Automação
Exercício Programa 2 - 2012
Prof. Dr. Fabio Gagliardi Cozman
Prof. Dr. Newton Maruyama

[Item. 1] Deseja-se estabelecer um banco de dados para o gerenciamento das notas da disciplina PMR2300. Neste cadastro, cada registro (informações de cada aluno) deve conter as seguintes informações:

- nomedoaluno,
- nusp,
- Notas que compõem o cálculo da média: P1, P2, T, L, EP1, EP2, EP3, EP4, MF.

Inicialmente o aluno deve estabelecer uma classe que contenha as informações acima. Por exemplo, como a seguir (Listagem 1). Nessa listagem são colocados exemplos de comentários que ajudam a criar a documentação do programa.

Listing 1: Definição da classe `InfoAluno`.

```
public class InfoAluno {
/* Variaveis da classe */
private String nomedoaluno, nusp;
private double
P1,             // nota da prova 1
P2,             // nota da prova 2
EP1,            // nota do EP1
EP2,            // nota do EP2
EP3,            // nota da EP3
EP4,            // nota da EP4
T,              // media de provas
L,              // media dos EPs
MF;             // Media Final

/*
Construtores: voce deve colocar aqui os construtores adequados
*/
public InfoAluno()
{
// implementacao
}

/*
public void ColocaNome(String n):
- proposito: carrega o nome do aluno no objeto
  da classe InfoAluno
- parametros de entrada:
  n: string que contem o nome do aluno
```

```
- parametros de saida:
  nao possui
*/
public void ColocaNome(String n)
{
nomedoaluno=n;
}
/*

public String PegaNome():
- proposito: devolve o nome do aluno no objeto
  da classe InfoAluno
- parametros de entrada:
  nao possui
- parametros de saida:
  nome: string contendo o nome do aluno;
*/
public String PegaNome()
{
return(nomedoaluno)
}
```

Um dos conceitos básicos de programação orientada a objetos é o de *Tipos Abstratos de Dados* (*Abstract Data Types*). Dentro desse conceito, todas as estruturas de dados internas não podem ser manipuladas diretamente pelo mundo externo, ou seja, um outro objeto não pode acessar diretamente as variáveis `nomedoaluno`, `P1`, `P2`, `L1`, `L2`, `L3`, `L4`, `T`, `L`, `MF`. Todos os detalhes de implementação interna ficam escondidas (*information hiding*), ou sem visibilidade pelo *mundo externo*. A maneira que JAVA permite implementar esse conceito é com o uso do modificador de visibilidade `private` antes da declaração da variável ou método. A comunicação com o mundo externo deve ser feita exclusivamente através de métodos públicos (com o identificador `public`), como `ColocaNome`, `PegaNome`, etc. Dessa forma, o usuário *enxerga* as estruturas de dados através de comportamentos definidos pelas operações, ou métodos, que alteram o conteúdo (ou o estado, como em sistemas dinâmicos) das estruturas de dados. Não importa quais estruturas de dados concretas (`int`, `double`, `String`, etc.) pertencem a classe mas sim como o comportamento é definido pelas operações. Assim, deve-se criar métodos que possam manipular todas as variáveis internas da classe `InfoAluno`.

Num segundo passo, você deve criar uma outra classe denominada BancoDeDadosInfoAluno. Abaixo (Listagem 2) são indicadas as estruturas de dados e o construtor da classe em questão.

Listing 2: Definição da classe `BancoDeDadosInfoAluno`.

```
public class BancoDeDadosInfoAluno {
private final int NMax = 100; // aqui NMax e' uma constante
                      //ou seja voce pode declarar o tamanho
                     // de vetor que quiser
private InfoAluno BancoDeDados[]; // vetor do tipo InfoAluno
int NumeroDeAlunos; // Numero de alunos ja inseridos
/*
public BancoDeDadosInfoAluno(int NMax):
- proposito: atribui o valor inicial nulo
```

```
  para o numero de alunos
- parametros de entrada:
  nao possui
- parametros de saida:
  nao possui
*/
public BancoDeDadosInfoAluno()
InfoAluno BancoDeDados = new InfoAluno[NMAX];
NumeroDeAlunos=0;
}
/*
Projete um construtor alternativo que constroi o banco de dados
fazendo a leitura dos dados a partir de um arquivo
*/
\\ etc. etc.
}
```

Obviamente, várias operações são necessárias, como por exemplo:

- `Public void InsereAluno(String inpnomedoaluno, String inpnusp, double inpP1, double inpP2, double inpEP1, double inpEP2, double inpEP3, double inpEP4, double inpT, double inpL, double inpMF)`: um novo aluno é inserido sempre no final do vetor `BancoDeDados`. Não podem haver inserções se o vetor atingir o limite de sua capacidade, i.e., sempre devemos ter:

$$numerodealunos \leq Nmax. \tag{1}$$

  Inicialmente, vamos supor que temos um objeto denominado `BancoDeDadosInfoAluno c`. No construtor acima temos a seguinte linha de comando:

```
InfoAluno BancoDeDados = new InfoAluno[NMAX];
```

  A partir de agora, você tem um vetor `c.InfoAluno[]` com `NMAx` posições cujos conteúdos podem apontar para um objeto do tipo `InfoAluno`.

[Item. 2] (a) Interface com o usuário através de teclado e tela.

(b) Menu de operações deve conter:

- Inserção de aluno (inserir em ordem alfabética),
- Remoção de aluno,
- Busca de aluno pelo nome e impressão na tela (implemente um algoritmo de busca binária),
- Leitura da base de dados de um arquivo no formato ASCII,
- Escrita da base de dados em um arquivo no formato ASCII,
- Impressão do relatório na tela.

(c) Detalhamento das classes `InfoAluno` e `BancoDeDadosInfoAluno`.

(d) Métodos para cálculo da média das provas `T`, cálculo da média dos EPs `L`, cálculo da média final `MF`, cálculo da média e desvio padrão para todas as notas da turma, i.e., cálculo da média e desvio padrão de `P1`, `P2`, `T`, `EP1`, `EP2`, `EP3`, `EP4`, `T` e `MF`.

(e) Geração de relatório e gravação em arquivo. O relatório é composto pela base de dados mais as estatísticas descritas acima.

[Item. 3] **Coisas importantes:**

- **O exercício deve ser feito individualmente;**
- **Entregue ao monitor da disciplina no dia 07 de Maio de 2012**
  - **Documentação impressa descrevendo o seu projeto em uma página A4,**
  - **Código fonte e código compilado (bytecode) em um CD.**