Universidade Federal do Rio Grande do Norte
Instituto Metropole Digital
IMD0040 - Linguagem de Programação 2
Aula19 - Lista de exercícios

- Esta lista de exercício é composta por 3 questões: uma fácil, uma média e uma difícil.
- As questões valem, respectivamente, 20%, 30% e 50%.
- Não se assustem com o tamanho dos textos.
- Serão aceitos apenas arquivos **.zip**.
- Não será aceita nenhuma questão feita no BlueJ.
- Não serão aceitos arquivos enviados por e-mail.

ATENÇÃO! Os exercícios desta aula prática de hoje, 26/09, estão demasiadamente fáceis. O principal motivo é que vocês não percam tanto tempo devido ao trabalho não tão trivial que estão desenvolvendo. **Entretanto, por ser fácil, a correção será binária.**

Fácil - A teoria (20%)

Escreva, em um .TXT, um resumo sobre Classes, Classes Abstratas, Interfaces e Herança. O que é uma classe? Como funciona a herança de uma classe? Quais as peculiaridades das classes abstratas? Interfaces podem herdar algo? Classes abstratas podem ter métodos concretos? E vice versa?

Faça o resumo com todo seu conhecimento sobre o assunto.

Médio - A utilização da Interface (30%)

Crie uma interface chamada *Calculavel*. Esta interface deverá conter a assinatura dos principais métodos de uma calculadora (somar, subtrair, multiplicar, dividir…).

Crie uma classe que implemente essa interface e faça o uso dela. Essa sua classe deve-se chamar *Calculadora* e precisa, obrigatoriamente, ter uma memória (uma pilha, procure na API). Esta pilha deve ser alimentada sempre que uma operação for chamada, armazenando os resultados.

Sua classe precisará ter um método para, quando requisitado, desempilhar a pilha e exibir os resultados das operações feitas até então. Quando este método for chamado, a pilha deverá ser esvaziada.

Difícil - Ordenando coleções com a interface Comparable (50%)

A classe java.util.Collections contém métodos estáticos que ajudam a manipular coleções, entre eles temos o método sort, que permite realizar a ordenação dos elementos presentes na coleção.

Porém qual é o critério de ordenação utilizado, haja vista que uma coleção é parametrizada e pode armazenar qualquer tipo de objeto? Para que o sort seja chamado corretamente, é necessário que nossa classe implemente a interface java.lang.Comparable, que define quem será nossa "ordem natural". A interface Comparable possui apenas um método, chamado compareTo:

```
public interface Comparable<T> {
    int compareTo(T outro);
}
```

Se o objeto atual for "menor" que o outro, devemos retornar -1 (ou qualquer int negativo, indicando que this deve vir “antes” de "outro"), se for "maior" retornamos 1 (ou qualquer int positivo) e se for igual retornamos 0.

Sabendo disso, desenvolva uma classe chamada NotaAluno. Essa classe deve conter os atributos:

```
int matricula;
String nomeAluno;
double nota1;
double nota2;
double nota3;
```

A classe NotaAluno deve implementar a interface Comparable com critério de ordenação de acordo com a média aritmética (em ordem crescente).

Para teste crie 5 objetos NotaAluno, e armazene-os em uma lista. Em seguida chame o método sort da classe java.util.Collections e visualize na tela o resultado (sobrescreva o método toString da classe NotaAluno para facilitar a visualização no console).