Aula Prática 6

Objetivos

Análise e concepção de um problema usando conceitos de programação por objetos em Java.
Programação Funcional – Expressões Lambda

Problema 6.1

Considere as seguintes entidades e características representativas de alimentos:

- Carne, tem variedade (vaca, porco, peru, frango, outra), proteínas (double), calorias (double), peso (double).
- Peixe, tem tipo (congelado ou fresco), proteínas (double), calorias (double), peso (double).
- Cereal, tem nome (String), proteínas (double), calorias (double), peso (double). É um alimento vegetariano.
- Legume, tem nome (String), proteínas (double), calorias (double), peso (double). É um alimento vegetariano.
- Prato, tem um nome (String) e composição (conjunto de alimentos)
- PratoVegetariano, tem um nome (String) e composição (conjunto de alimentos vegetarianos).
- PratoDieta, tem um nome (String) e composição (conjunto de alimentos) e limite máximo de calorias (double).
- Ementa – tem um nome (String), um local (String) e uma lista de pratos associados a cada dia da semana. Este devem ser mantidos ordenados por dia.

a) Analise o problema cuidadosamente e modele as interfaces e classes necessárias, as suas associações (herança, composição) bem como todos os atributos e métodos. Implemente todas as classes necessária, seguindo as seguintes considerações:

- Os valores de calorias e proteínas usados devem ser relativos a 100gr.
- Para cada prato deve ser possível obter informações sobre alimentos, peso total, calorias, proteínas,...
- Implemente os métodos hashCode(), equals(), toString() em todas as classes.
- Os pratos devem respeitar a interface Comparable para permitir usar o método *UtilCompare.sortArray* desenvolvido anteriormente (a ordenação será por calorias)
- As listas devem ser feitas preferencialmente com listas ligadas (usando a classe No como interna de Lista).

b) Teste a implementação com o seguinte programa:

```
public class Test {

    public static void main(String[] args) {
        Ementa ementa = new Ementa("Especial Caloiro", "Snack da UA");
        Prato[] pratos = new Prato[10];
```

```
        for (int i=0; i < pratos.length; i++){
            pratos[i] = randPrato(i);
            int cnt = 0;

            while (cnt <2){ // Adicionar 2 Ingredientes a cada Prato
                Alimento aux = randAlimento();
                if (pratos[i].addIngrediente(aux))
                    cnt++;
                else
                    System.out.println("ERRO: Não é possivel adicionar '" +
                                                aux + "' ao -> " + pratos[i]);
            }

            ementa.addPrato(pratos[i], DiaSemana.rand());      // Dia Aleatório

        }

        System.out.println("\n" + ementa);
    }

    // Retorna um Alimento Aleatoriamente
    public static Alimento randAlimento() {
        switch ((int) (Math.random() * 4)) {
            default:
            case 0:
                return new Carne(VariedadeCarne.frango, 22.3, 345.3, 300);
            case 1:
                return new Peixe(TipoPeixe.congelado, 31.3, 25.3, 200);
            case 2:
                return new Legume("Couve Flor", 21.3, 22.4, 150);
            case 3:
                return new Cereal("Milho", 19.3, 32.4, 110);
        }
    }

    // Retorna um Tipo de Prato Aleatoriamente
    public static Prato randPrato(int i) {
        switch ((int) (Math.random() * 3)) {
            default:
            case 0:
                return new Prato("Prato N." + i);
            case 1:
                return new PratoVegetariano("Prato N." + i + " (Vegetariano)");
            case 2:
                return new PratoDieta("Prato N." + i + " (Dieta)", 90.8);

        }
    }
}
```

Verifique se obteve um resultado similar a este:

```
ERRO: Não é possivel adicionar 'Carne frango, Proteínas 22.3, calorias 345.3, Peso
300.0' ao -> Dieta (0.0 Calorias) Prato 'Prato N.0 (Dieta)' composto por 0 Ingredientes
ERRO: Não é possivel adicionar 'Carne frango, Proteínas 22.3, calorias 345.3, Peso
300.0' ao -> Vegetariano Prato 'Prato N.3 (Vegetariano)' composto por 0 Ingredientes
ERRO: Não é possivel adicionar 'Peixe congelado, Proteínas 31.3, calorias 25.3, Peso
200.0' ao -> Vegetariano Prato 'Prato N.3 (Vegetariano)' composto por 0 Ingredientes
ERRO: Não é possivel adicionar 'Peixe congelado, Proteínas 31.3, calorias 25.3, Peso
200.0' ao -> Vegetariano Prato 'Prato N.3 (Vegetariano)' composto por 1 Ingredientes
ERRO: Não é possivel adicionar 'Peixe congelado, Proteínas 31.3, calorias 25.3, Peso
200.0' ao -> Vegetariano Prato 'Prato N.3 (Vegetariano)' composto por 1 Ingredientes
ERRO: Não é possivel adicionar 'Carne frango, Proteínas 22.3, calorias 345.3, Peso
```

```
300.0' ao -> Vegetariano Prato 'Prato N.3 (Vegetariano)' composto por 1 Ingredientes
ERRO: Não é possivel adicionar 'Carne frango, Proteínas 22.3, calorias 345.3, Peso
300.0' ao -> Vegetariano Prato 'Prato N.4 (Vegetariano)' composto por 1 Ingredientes
ERRO: Não é possivel adicionar 'Carne frango, Proteínas 22.3, calorias 345.3, Peso
300.0' ao -> Vegetariano Prato 'Prato N.4 (Vegetariano)' composto por 1 Ingredientes
ERRO: Não é possivel adicionar 'Peixe congelado, Proteínas 31.3, calorias 25.3, Peso
200.0' ao -> Vegetariano Prato 'Prato N.5 (Vegetariano)' composto por 1 Ingredientes
ERRO: Não é possivel adicionar 'Carne frango, Proteínas 22.3, calorias 345.3, Peso
300.0' ao -> Dieta (0.0 Calorias) Prato 'Prato N.9 (Dieta)' composto por 0 Ingredientes

Dieta (84.2 Calorias) Prato 'Prato N.0 (Dieta)' composto por 2 Ingredientes, dia Segunda
Prato 'Prato N.1' composto por 2 Ingredientes, dia Terça
Prato 'Prato N.7' composto por 2 Ingredientes, dia Terça
Vegetariano Prato 'Prato N.2 (Vegetariano)' composto por 2 Ingredientes, dia Quarta
Prato 'Prato N.8' composto por 2 Ingredientes, dia Quarta
Vegetariano Prato 'Prato N.3 (Vegetariano)' composto por 2 Ingredientes, dia Quinta
Vegetariano Prato 'Prato N.5 (Vegetariano)' composto por 2 Ingredientes, dia Quinta
Dieta (84.2 Calorias) Prato 'Prato N.6 (Dieta)' composto por 2 Ingredientes, dia Quinta
Dieta (71.28 Calorias) Prato 'Prato N.9 (Dieta)' composto por 2 Ingredientes, dia Sexta
Vegetariano Prato 'Prato N.4 (Vegetariano)' composto por 2 Ingredientes, dia Domingo
```

c) Construa um programa Ementa que permita, genericamente:
   - Ingrediente
     - Adicionar Carne
     - Adicionar Peixe
     - Adicionar Cereal
     - Adicionar Legume
   - Prato
     - Cria Prato
     - Apaga Prato
     - Seleciona Prato
     - Adiciona Ingrediente
     - Remove Ingrediente
   - Ementa
     - Adiciona Prato
     - Remove Prato
     - Imprime Ementa

d) Inclua na aplicação desenvolvida a possibilidade de tornar persistente a Ementa, incluindo todos os Pratos e Alimentos definidos, i.e. que permita guardar e carregar essa Ementa em ficheiro.

Problema 6.2

Pretendemos desenvolver um bloco de código genérico (um método) que satisfaça os seguintes requisitos:

a) Receba dois argumentos: uma estrutura de dados tipo List (java.util.List) e um filtro;

b) Retorne um novo objeto do tipo List contendo os elementos da lista de entrada que satisfazem os critérios definidos no argumento filtro;

c) Funcione com uma List contendo qualquer tipo de objetos;

Teste o método desenvolvido com o seguinte programa:

```
public class Test {
        public static void main(String[] args) {
           List<Figura> lista = new ArrayList<Figura>();
           lista.add(new Circulo(2)); lista.add(new Circulo(1, 3, 1));
           lista.add(new Quadrado(5)); lista.add(new Quadrado(3, 4, 2));
           lista.add(new Rectangulo(2, 3)); lista.add(new Rectangulo(3, 4, 5, 3));
           lista.add(new Rectangulo(1, 1, 5, 6));

           System.out.println("Figuras Filter 1:");
           List<Figura> ret = ListsProcess.filter(lista, f -> f.area() > 20);
           printList(ret);

           System.out.println("\nFiguras Filter 2:");
           ret = ListsProcess.filter(lista, f -> f.perimetro() < 15);
           printList(ret);

           System.out.println("\nFiguras Filter 3:");
           ret = ListsProcess.filter(lista, f -> f.perimetro() < 15 && f.area() > 10);
           printList(ret);

           List<Estudante> lista2 = new ArrayList<Estudante>();
           lista2.add(new Estudante("Andreia", 9855678, new Data(18, 7, 1974)));
           lista2.add(new Estudante("Monica", 75266454, new Data(11, 8, 1978)));
           lista2.add(new Estudante("Jose", 85265426, new Data(15, 2, 1976)));
           lista2.add(new Bolseiro("Maria", 8976543, new Data(12, 5, 1976)));
           lista2.add(new Bolseiro("Xico", 872356522, new Data(21, 4, 1975)));

           System.out.println("\nEstudante Filter 1:");
           List<Estudante> ret2 = ListsProcess.filter(lista2, e -> e.getNMec() < 103);
           printList(ret2);

           System.out.println("\nEstudante Filter 2:");
           ret2 = ListsProcess.filter(lista2,
                                      e -> e.getClass().getSimpleName().equals("Bolseiro"));
           printList(ret2);
        }

        private static <T> void printList(List<T> myList) {
           for (T e : myList)
                System.out.println(e);
        }
}
```

Verifique se obteve o seguinte resultado:

```
Filter 1:
Quadrado de Centro x: 0.0, y:0.0 e de lado 5.0
Rectangulo de Centro x: 1.0, y:1.0, altura 6.0, comprimento 5.0

Filter 2:
Circulo de Centro x: 0.0, y:0.0 e de raio 2.0
Circulo de Centro x: 1.0, y:3.0 e de raio 1.0
Quadrado de Centro x: 3.0, y:4.0 e de lado 2.0
Rectangulo de Centro x: 0.0, y:0.0, altura 3.0, comprimento 2.0

Filter 3:
Circulo de Centro x: 0.0, y:0.0 e de raio 2.0

Estudante Filter 1:
Andreia, BI: 9855678, Nascido na Data: 18/7/1974, NMec: 100, Inscrito em Data:
20/10/2016
Monica, BI: 75266454, Nascido na Data: 11/8/1978, NMec: 101, Inscrito em Data:
20/10/2016
Jose, BI: 85265426, Nascido na Data: 15/2/1976, NMec: 102, Inscrito em Data: 20/10/2016

Estudante Filter 2:
Maria, BI: 8976543, Nascido na Data: 12/5/1976, NMec: 103, Inscrito em Data: 20/10/2016,
Bolseiro com bolsa de 0 Euros
Xico, BI: 872356522, Nascido na Data: 21/4/1975, NMec: 104, Inscrito em Data:
20/10/2016, Bolseiro com bolsa de 0 Euros
```