Fundação CECIERJ - **Vice Presidência de Educação Superior a Distância**

Curso de Tecnologia em Sistemas de Computação
AD1 de Programação III
2° semestre de 2015

Nome:
Pólo:

Exercício (ENTREGAR OS ARQUIVOS EM MÍDIA, PARA FINS DE TESTE, JUNTAMENTE COM A AD IMPRESSA):
Um indivíduo estacionou seu Food Truck nas proximidades do seu Pólo. Especializado em lanches, o empreendimento dele permite que o cliente selecione o tipo de lanche e quais ingredientes farão parte daquela refeição. O cardápio afixado na parede do veículo mostra as seguintes opções a serem combinadas:

Opções de Lanche	Opções de Ingredientes			
	Tipos de Carne	Tipos de Queijo	Tipos de Salada	Tipos de Molho
Pizza	Boi	Cheddar	Alface	Barbecue
Sanduíche	Peixe	Mozzarella	Azeitona	Caesar
-	Frango	Prato	Pimentão	Tomate

O preço do lanche depende das opções de ingredientes escolhidas pelo cliente, sendo calculado como a soma dos valores individuais dos itens que compõe o lanche, mais 40% de lucro para o caso de pizza e 30% de lucro para o caso de sanduíche.
O custo individual da porção de cada ingrediente é apresentado na tabela abaixo:

Opções de Ingredientes			
Tipos de Carne	**Tipos de Queijo**	**Tipos de Salada**	**Tipos de Molho**
R$1,00	R$1,00	R$0,50	R$0,90
R$2,00	R$0,50	R$0,75	R$1,00
R$1,50	R$0,50	R$0,75	R$0,50

Nesta questão:

a. Estruture o esquema de lanches em uma hierarquia de classes contendo duas árvores distintas de classes: uma para tipos de lanche e outra para tipos de ingredientes.
   - A hierarquia de lanches deve ter dois níveis: um abstrato e outro concreto. Deduza da primeira tabela que classes pertencem a cada nível e implemente todas as classes nessa hierarquia.
   - A hierarquia de ingredientes deve ter três níveis: dois abstratos e um concreto. Deduza da primeira tabela que classes pertencem a cada nível.

b. Conforme mostra a segunda tabela, todo tipo de ingrediente possui um preço unitário.
   - Esse atributo deve ser colocado de forma adequada na estrutura de classes, de modo a explorar os princípios de herança e encapsulamento.
   - O preço unitário é informado no momento da criação do objeto. Ele não pode ser menor do que zero, pode ser alterado após a criação do objeto através do método `void setPrecoUnitaro(double valor)` e pode ser consultado a partir do método `double getPrecoUnitario()`.

c. Os objetos que são instâncias de classes na hierarquia de tipos de lanche agregam

uma coleção de ingredientes escolhidos pelo cliente.

- Esse atributo deve ser colocado de forma adequada na estrutura de classes, de modo a explorar os princípios de herança e encapsulamento.
- Os ingredientes escolhidos são informados no momento da criação do objeto por meio de um array simples que deve ser copiado para dentro do objeto. Essa coleção não pode ser alterada após a criação do objeto de lanche. Também não há necessidade de consulta a seu conteúdo.

d. Todas as classes devem conter apenas um construtor cada. A escolha adequada por construtor padrão, de inicialização ou se cópia faz parte da avaliação.
   - As classes na hierarquia de tipos de lanche possuem um comportamentos que é mapeado para o métodos **`double`** `getPrecoTotal()`. Observe que o valor retornado pela função varia em função do tipo de lanche. Utilize polimorfismo para implementar essa variação na forma de cálculo de preço sem a inclusão de um atributo extra representando o percentual de ajuste do valor.

Não são permitidas a definição de classes fora das hierarquias indicadas e a declaração de atributos, construtores e métodos diferentes dos especificados nos itens (b) a (d).

RESPOSTA:

```
public class AD1_2015_2{
    public static void main(String[] args){
        Ingrediente[] ingr = new Ingrediente[12];
        ingr[0] = new Boi (1.0);
        ingr[1] = new Frango (2.0);
        ingr[2] = new Peixe (1.5);

        ingr[3] = new Cheddar (1.0);
        ingr[4] = new Mozzarella (0.5);
        ingr[5] = new Prato (0.5);

        ingr[6] = new Alface (0.5);
        ingr[7] = new Azeitona (0.75);
        ingr[8] = new Pimentao (0.75);

        ingr[9]  = new Barbecue (0.9);
        ingr[10] = new Caesar (1.0);
        ingr[11] = new Tomate (0.5);

        Lanche[] l = new Lanche[2];
        l[0] = new Pizza(ingr);
         System.out.println(l[0]);

        l[1] = new Sanduiche(ingr);
        System.out.println(l[1]);
    }
}

abstract class Ingrediente {
    private double preco;

    public Ingrediente(double preco) { setPrecoUnitario(preco); }

    public final double getPrecoUnitario() { return this.preco; }
```

```
    public final void setPrecoUnitario(double preco) {
        if (preco < 0.0) {
            System.out.println("Preco invalido.");
            preco = 0.0;
        }
        this.preco = preco;
    }
}

abstract class Carne extends Ingrediente {
    public Carne(double preco) { super(preco); }
    public String toString() { return "Carne: "; }
}

class Boi extends Carne {
    public Boi(double preco) { super(preco); }
    public String toString() { return super.toString() + "Boi (" +
getPrecoUnitario() + ")\n"; }
}

class Frango extends Carne {
    public Frango(double preco) { super(preco); }
    public String toString() { return super.toString() + "Frango (" +
getPrecoUnitario() + ")\n"; }
}

class Peixe extends Carne {
    public Peixe(double preco) { super(preco); }
    public String toString() { return super.toString() + "Peixe (" +
getPrecoUnitario() + ")\n"; }
}

abstract class Queijo extends Ingrediente {
    public Queijo(double preco) { super(preco); }
    public String toString() { return "Queijo: "; }
}

class Cheddar extends Queijo {
    public Cheddar(double preco) { super(preco); }
    public String toString() { return super.toString() + "Cheddar (" +
getPrecoUnitario() + ")\n"; }
}

class Mozzarella extends Queijo {
    public Mozzarella(double preco) { super(preco); }
    public String toString() { return super.toString() + "Mozzarella (" +
getPrecoUnitario() + ")\n"; }
}

class Prato extends Queijo {
    public Prato(double preco) { super(preco); }
    public String toString() { return super.toString() + "Prato (" +
getPrecoUnitario() + ")\n"; }
}
```

```
abstract class Salada extends Ingrediente {
    public Salada(double preco) { super(preco); }
    public String toString() { return "Salada: "; }
}

class Alface extends Salada {
    public Alface(double preco) { super(preco); }
     public String toString() { return super.toString() + "Alface (" +
getPrecoUnitario() + ")\n"; }
}

class Azeitona extends Salada {
    public Azeitona(double preco) { super(preco); }
     public String toString() { return super.toString() + "Azeitona (" +
getPrecoUnitario() + ")\n"; }
}

class Pimentao extends Salada {
    public Pimentao(double preco) { super(preco); }
     public String toString() { return super.toString() + "Pimentao (" +
getPrecoUnitario() + ")\n"; }
}

abstract class Molho extends Ingrediente {
    public Molho(double preco) { super(preco); }
    public String toString() { return "Molho: "; }
}

class Barbecue extends Molho {
    public Barbecue(double preco) { super(preco); }
     public String toString() { return super.toString() + "Barbecue (" +
getPrecoUnitario() + ")\n"; }
}

class Caesar extends Molho {
    public Caesar(double preco) { super(preco); }
    public String toString() { return super.toString() + "Caesar (" +
getPrecoUnitario() + ")\n"; }
}

class Tomate extends Molho {
    public Tomate(double preco) { super(preco); }
     public String toString() { return super.toString() + "Tomate (" +
getPrecoUnitario() + ")\n"; }
}

abstract class Lanche {
    private final Ingrediente[] ingredientes;

    public Lanche(Ingrediente[] ingredientes) {
        if (ingredientes == null) System.out.println("Ingrediente nulo.");
        for (Ingrediente ingrediente : ingredientes) {
            if (ingrediente == null)
               System.out.println("Ingrediente nulo.");
        }
        this.ingredientes = ingredientes.clone();
    }
```

```
    public double getPrecoTotal() {
        double soma = 0.0;
        for (Ingrediente ingrediente : this.ingredientes) {
            soma += ingrediente.getPrecoUnitario();
        }
        return soma;
    }

    public String toString(){
       String resp = "";
       for (Ingrediente ingrediente : this.ingredientes) {
            resp += ingrediente.toString();
       }
       return resp;
    }
}

class Pizza extends Lanche {
    public Pizza(Ingrediente[] ingredientes) { super(ingredientes); }
    public double getPrecoTotal() { return super.getPrecoTotal() * 1.40; }
   public String toString(){ return "Pizza (" + getPrecoTotal() + ")\n" +
super.toString(); }
}

class Sanduiche extends Lanche {
    public Sanduiche(Ingrediente[] ingredientes) { super(ingredientes); }
    public double getPrecoTotal() { return super.getPrecoTotal() * 1.30; }
   public String toString(){ return "Sanduiche (" + getPrecoTotal() + ")\n"
+ super.toString(); }
}
```