Guiões P3 - Best Of

[Vamos treinar para o exame final, suponho que queiram isto]

Ccrie uma workspace no desktop com o nome treino99999 quem que 99999 são é o seu numero mecanográfico crie um pacote para cada exercício no fim faça zip ao workspace e submeta

Devemos usar os slides sempre que necessário

EX1) 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:

```
import java.io.*;

public class Test {

	public static void main(String[] args) throws IOException{
		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);
                ementa.toFile();
        }

        // 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. **USAR SERIALIZAÇÃO**

Ex2) Problema 9.2

Considere a seguinte classe:

```
public class Gelataria {
        public static void main(String args[]) {
        Gelado ice;
        ice = new GeladoSimples("Baunilha");
        ice.base(2);
        new Copo(ice).base(3);
        new Cone(ice).base(1);
        new Topping(ice, "Canela").base(2);
        ice = new Topping(ice, "Nozes");
        ice.base(1);
        ice = new Topping(new Cone(new GeladoSimples("Morango")), "Fruta");
        ice.base(2);
        ice = new Topping(
                new Topping(
                        new Copo(new GeladoSimples("Manga")), "Chocolate"), "Natas");
        ice.base(4);
        ice = new Topping(ice, "Pepitas");
        ice.base(3);
        }
}
```

Desenvolva classes que representem adequadamente o problema e que para o programa anterior forneça a seguinte saída de dados:

2 bolas de gelado de Baunilha
3 bolas de gelado de Baunilha em copo
1 bola de gelado de Baunilha em cone
2 bolas de gelado de Baunilha com Canela
1 bola de gelado de Baunilha com Nozes
2 bolas de gelado de Morango em cone com Fruta
4 bolas de gelado de Manga em copo com Chocolate com Natas
3 bolas de gelado de Manga em copo com Chocolate com Natas com Pepitas

REVER IMPLEMENTAÇÕES DE FACTORY, SINGLETON E ETC.

Ex3) Problema 12.2[não é importante,se der tempo]

Tome como referência o seguinte código (IPlugin.java e Plugin.java). Construa um conjunto de classes que implementem IPlugin e que permitam mostrar as funcionalidades do programa principal.

```
// IPlugin.java
package reflection;

public interface IPlugin {
        public void fazQualQuerCoisa ();
}
```

```
// Plugin.java
package reflection;
import java.io.File;
import java.util.ArrayList;
import java.util.Iterator;

abstract class PluginManager {
	public static IPlugin load(String name) throws Exception {
		Class<?> c = Class.forName(name);
		return (IPlugin) c.newInstance();
	}
}
public class Plugin {
public static void main(String[] args) throws Exception {
	File proxyList = new File("reflection/plugins");
	ArrayList<IPlugin> plgs = new ArrayList<IPlugin>();
	for (String f: proxyList.list()) {
		deti.ua, programação 3, 2019/20 35
		try {
			plgs.add(PluginManager.load("reflection."+f.substring(0,f.lastIndexOf('.'))))
		;
		}
		catch (Exception e) {
			e.printStackTrace();
		}
	}
		Iterator<IPlugin> it = plgs.iterator();
		while (it.hasNext()) {
			it.next().fazQualQuerCoisa();
		}
	}
}
```

EX4) Problema 11.2

Implemente os métodos solicitados utilizando a nova Java 8 Stream API.

a) Reimplemente o método maiorFiguraJ7.

```
private static Figura maiorFiguraJ7(List<Figura> figs) {
	Figura maior = figs.get(0);
		for (Figura f : figs) {
			if (f.compareTo(maior) >= 1)
			maior = f;
		}
		return maior;
	}
```

b) Reimplemente a alínea a) mas utilizando agora o perímetro como elemento de

comparação (isto sem mexer na implementação de Figura).
c) Implemente um método que retorne a soma das áreas das figuras da lista.

```
private static double areaTotalJ8(List<Figura> figs) {
...
}
```

d) Altere a alínea c) para só calcular a área total do subtipo de Figura passado como argumento.

```
private static double areaTotalJ8(List<Figura> figs,String subtipoNome){
...
}
```

EX5) Problema 13.3 [ignorar não vou fazer]

Para cada uma das tarefas seguintes, defina a(s) estrutura(s) de dados mais adequada(s), e teste-a usando 3 ou mais elementos, com as operações adicionar, listar e remover.

a) A empresa Brinca&Beira (BB) precisa de um registo com os nomes de todos os seus empregados.
b) Em cada mês é selecionado aleatoriamente um funcionário para receber um brinquedo grátis. Deve ser possível guardar todos os pares funcionário-brinquedo.
c) A empresa decidiu atribuir o primeiro nome de um empregado a cada produto. Prepare uma lista destes nomes sabendo que um nome só poderá ser usado uma vez.
d) A BB decide, entretanto, que só quer usar os nomes mais populares para os seus brinquedos. Precisamos de uma estrutura com o número de funcionários que têm cada primeiro nome.
e) A empresa adquire ingressos para a próxima temporada da equipa local de futebol, para serem distribuídos rotativamente pelos funcionários. Crie a estrutura mais adequada – pode usar uma ordem qualquer.