

Universidade Federal de Goiás Instituto de Informática POO –
Laboratório (22/10/2024):

Padrões de Projeto (Design Pattern) Estudo de Caso: Abstract Factory (Em grupo, o mesmo do Trabalho Final da Disciplina (TFD))

Estudantes:

1. Jair Eugênio Ferreira – 202401791
2. João Vitor Sousa de Oliveira – 202401794
3. Lucas Almeida Oliveira Isaac – 202401802
4. Lucca Magnino – 202401805

Turma: 2024-1 Ciências da Computação

1) Responda as seguintes questões sobre o código-fonte AbstractFactory_java.zip disponibilizado na página da disciplina no SIGAA. Obs.: Antes de responder à questão: • Crie um Projeto Java no Eclipse • Copie o arquivo compactado AbstractFactory_java.zip e descompacte os arquivos no mesmo.

a) Qual é a classe .java que deve ser executada para gerar um output? Por que?

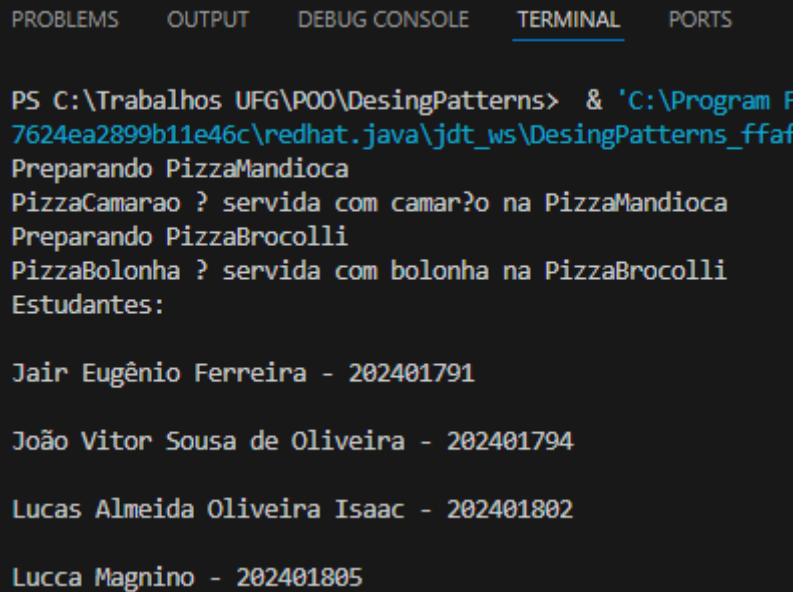
A classe que deve ser executada é “Pizzaria.java”, visto que nessa classe ocorre a criação de uma “main”, sendo responsável por chamar o método criado na própria classe “fazerPizzas”, utilizando de todas as outras classes e, conseqüentemente, realizando a impressão no output.

b) Faça uma pequena modificação no fonte da classe da letra “a” inserindo e imprimindo a(s) matrícula(s) e o nome(s) do grupo. Capture a tela () da janela de execução ao rodar o output (saída) na linguagem Java.

A modificação foi adicionar as seguintes linhas de código à main:

```
System.out.println("Estudantes: \n");
System.out.println ("Jair Eugênio Ferreira - 202401791 \n");
System.out.println ("João Vitor Sousa de Oliveira - 202401794 \n");
System.out.println ("Lucas Almeida Oliveira Isaac - 202401802 \n");
System.out.println ("Lucca Magnino - 202401805 \n");
```

Segue a print do output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Trabalhos UFG\POO\DesingPatterns> & 'C:\Program F
7624ea2899b11e46c\redhat.java\jdt_ws\DesingPatterns_ffaf
Preparando PizzaMandioca
PizzaCamarao ? servida com camar?o na PizzaMandioca
Preparando PizzaBrocolli
PizzaBolonha ? servida com bolonha na PizzaBrocolli
Estudantes:

Jair Eugênio Ferreira - 202401791

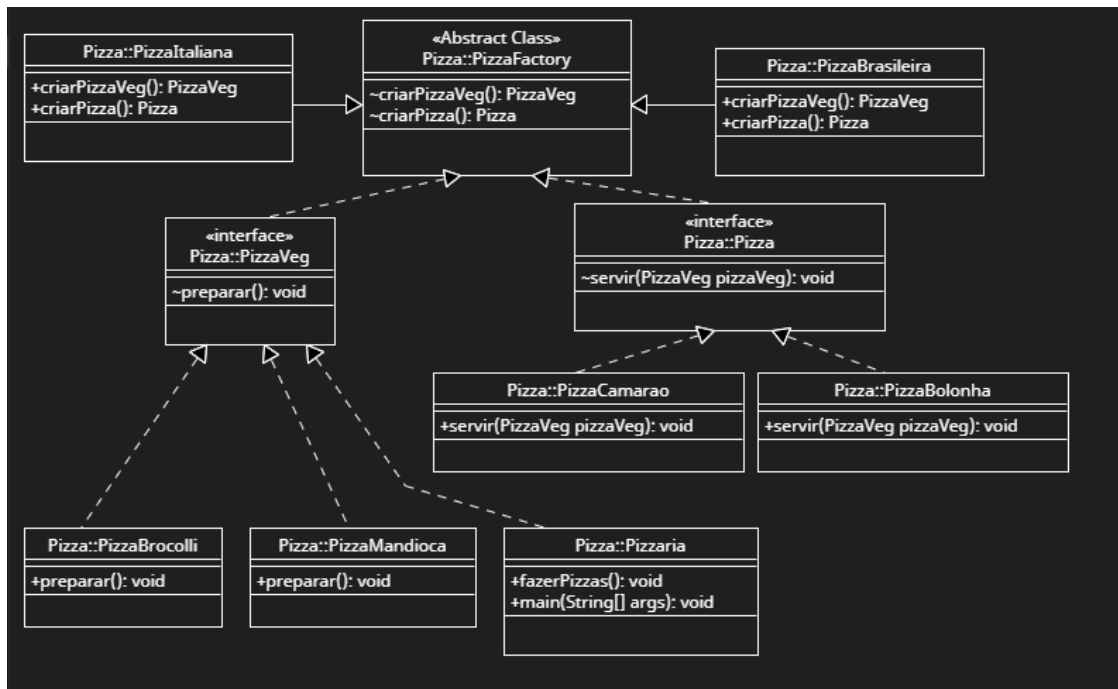
João Vitor Sousa de Oliveira - 202401794

Lucas Almeida Oliveira Isaac - 202401802

Lucca Magnino - 202401805
```

c) Qual é o arquivo que contém o Diagrama de Classe no UMLet da Abstract Factory? Capture a tela () com a figura do Diagrama de Classe do UMLet.

O arquivo que contém o diagrama de classe no UMLet é o “AbstractFabricaPizza.uxf”. Segue a print do diagrama:



2) Responda as seguintes questões sobre o código-fonte AbstractFactory_Python.zip disponibilizado na página da disciplina no SIGAA. Obs.: Antes de responder à questão:

- Instale a extensão do Python da própria Microsoft no Visual Studio Code, caso já não esteja instalada.
- Crie um diretório para o projeto Python
- Copie o arquivo compactado AbstractFactory_Python.zip e descompacte os arquivos no mesmo.

a) Qual é a classe .py que deve ser executada para gerar um output? Por que?

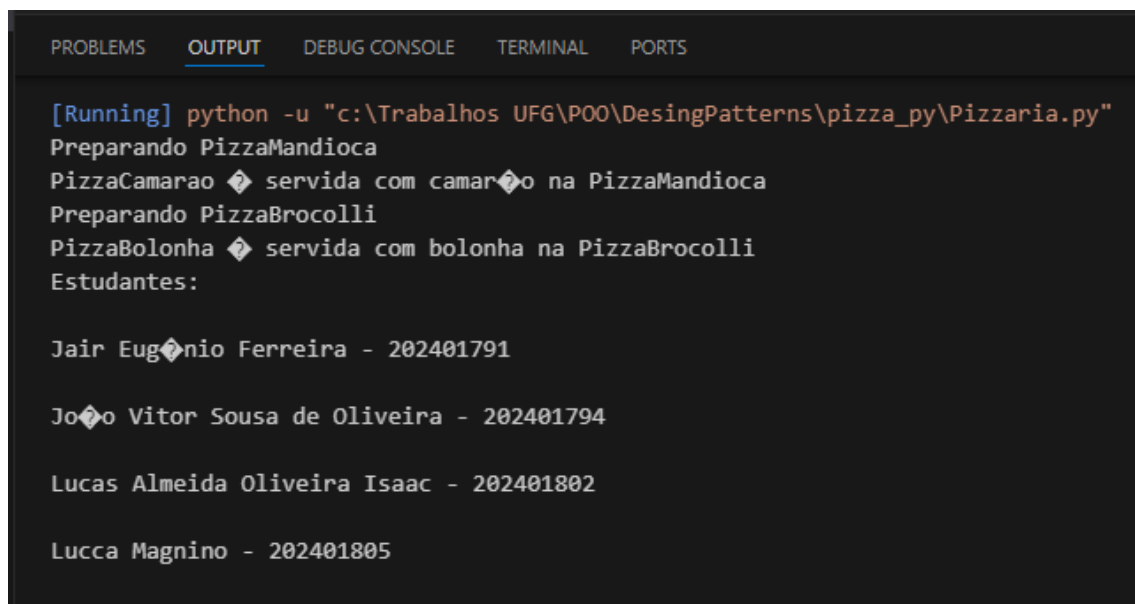
A classe que deve ser executada é a “Pizzaria.py”, visto que chama as outras funções criadas pelas outras classe, sendo executadas apenas na classe citada.

b) Faça uma pequena modificação no fonte da classe da letra “a” inserindo e imprimindo a(s) matrícula(s) e o nome(s) do grupo. Capture a tela () da janela de execução ao rodar o output (saída) na linguagem Python.

Para imprimir o que foi pedido, adicionei as seguintes linhas de código a classe:

```
print ("Estudantes: \n")
print ("Jair Eugênio Ferreira - 202401791 \n")
print ("João Vitor Sousa de Oliveira - 202401794 \n")
print ("Lucas Almeida Oliveira Isaac - 202401802 \n")
print ("Lucca Magnino - 202401805 \n")
```

Segue a print do output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

[Running] python -u "c:\Trabalhos UFG\P00\DesingPatterns\pizza_py\Pizzaria.py"
Preparando PizzaMandioca
PizzaCamarao 🍷 servida com camarão na PizzaMandioca
Preparando PizzaBroccoli
PizzaBolonha 🍷 servida com bolonha na PizzaBroccoli
Estudantes:

Jair Eugênio Ferreira - 202401791

João Vitor Sousa de Oliveira - 202401794

Lucas Almeida Oliveira Isaac - 202401802

Lucca Magnino - 202401805
```

c) Copie o Diagrama de Classe do UMLet da letra “c” do exercício 1. Modifique o Diagrama de Classe (use as propriedades de classes necessárias com a paleta do UMLet) de modo a ele se tornar compatível com o desenho das classes do arquivo AbstractFactory_Python.zip. Copie o arquivo do UMLet, renomeie para inserir no nome + “_Python.ufx”. Coloque a(s) matrícula(s) e nome(s) do(s) estudante(s) do grupo como note no Diagrama de Classe. Salve o arquivo .ufx e envie dentro do arquivo compactado (zipado) com a solução do grupo para esta letra da questão.

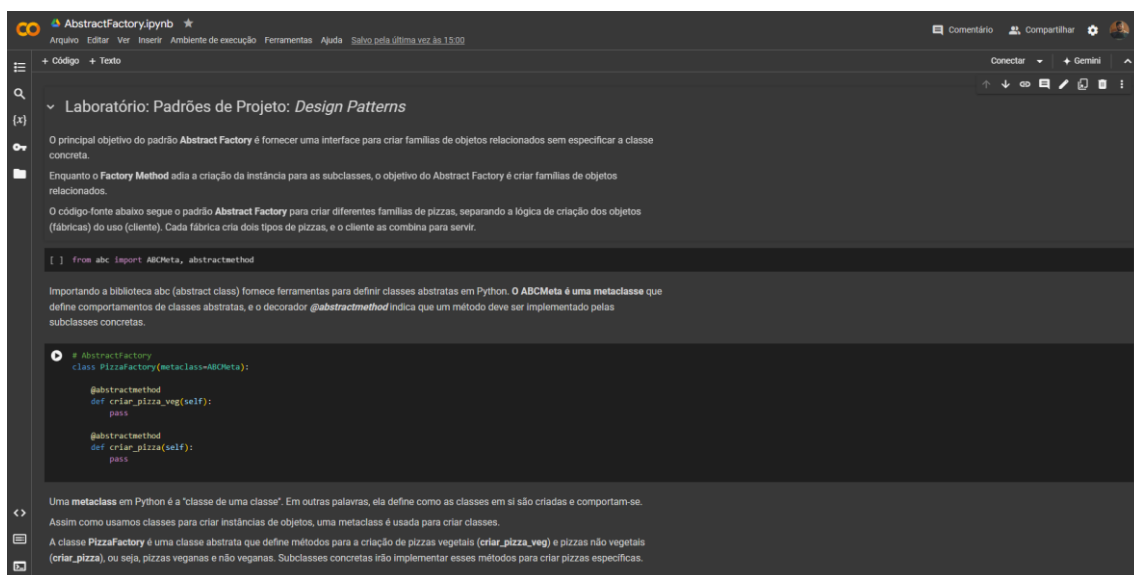
Não entendi muito bem como modificar para .py, segue no arquivo .zip com o comentário pedido.

d) Como é implementado o conceito de interface usando a linguagem Python?

Em Python, o conceito de interface pode ser implementado de forma indireta, pois a linguagem não possui uma estrutura própria de interface como em outras linguagens, como Java. Em vez disso, Python usa classes abstratas e o conceito de duck typing para simular interfaces. Primeiramente, as Classes Abstratas com o Módulo abc (Abstract Base Classes) permite definir métodos abstratos que subclasses devem implementar, simulando o comportamento de uma interface. Além disso, Python segue o princípio do duck typing, onde a conformidade com uma interface é baseada no comportamento de um objeto, em vez de sua herança explícita. Assim, em Python, interfaces são implementadas com estes métodos, sem a necessidade de uma estrutura formal de interface.

3) Responda as seguintes questões sobre o código-fonte AbstractFactory.ipynb disponibilizado na página da disciplina no SIGAA. Obs.: Antes de responder à questão: • Abra o Google Colab usando a conta do e-mail (@ufg.br ou @gmail.com). Site: <https://colab.research.google.com/notebooks/intro.ipynb> • Faça o upload do arquivo AbstractFactory.ipynb, opção, Arquivo -> Fazer Upload de notebook.

Segue a print do arquivo no google colab:



```
[ ] from abc import ABCMeta, abstractmethod

Importando a biblioteca abc (abstract class) fornece ferramentas para definir classes abstratas em Python. O ABCMeta é uma metaclasses que define comportamentos de classes abstratas, e o decorador @abstractmethod indica que um método deve ser implementado pelas subclasses concretas.

# AbstractFactory
class PizzaFactory(metaclass=ABCMeta):
    @abstractmethod
    def criar_pizza_veg(self):
        pass

    @abstractmethod
    def criar_pizza(self):
        pass
```

Uma metaclass em Python é a "classe de uma classe". Em outras palavras, ela define como as classes em si são criadas e comportam-se. Assim como usamos classes para criar instâncias de objetos, uma metaclass é usada para criar classes.

A classe PizzaFactory é uma classe abstrata que define métodos para a criação de pizzas vegetais (criar_pizza_veg) e pizzas não vegetais (criar_pizza), ou seja, pizzas veganas e não veganas. Subclasses concretas irão implementar esses métodos para criar pizzas específicas.

a) O que é o Google Colab? Qual a sua utilidade para rodar código em Python?

Google Colab é uma ferramenta criada e disponibilizada pelo Google, sua função é de escrever, executar e compartilhar códigos estruturados de forma específica no formato ipyn (notebook Jupyter). Sua utilidade em Python é a de rodar o código pelo

navegador, visto que, sem essa ferramenta, seria necessário baixar softwares separados e específicos para essa função de executar o arquivo Python.

b) O que é uma célula de código e uma célula de texto no Google Colab? Qual é a utilidade de usá-la e quando devo usar uma ou outra?

Uma célula de código é a parte que contém os comandos em Python, mostrando o output, sendo usadas para implementar a lógica do projeto e visualizar a saída. Já uma célula de texto escreve o conteúdo explicando o código, sendo usadas para contextualizar o projeto e torna-lo mais intuitivo.

c) Como é possível executar o código .ipynb, neste arquivo qual é a célula Qual(is) é ou são a(s) célula(s) que deve(m) ser executada para gerar um output? Qual é a célula que tem a classe que permite rodar o programa?

O código é executável pelo Google Colab a partir do próprio navegador, por meio do uso de CPUs e GPUs do Google. As células que devem ser executadas para gerar um output são as que possuem fundo mais escuro e linhas de código. A célula que tem a classe que permite rodar o programa é a última (“#Cliente” ou “Pizzaria”). Segue uma print da célula sendo executada:

```
# Cliente
class Pizzaria:

    def fazer_pizzas(self):
        for factory in [PizzaBrasileira(), PizzaItaliana()]:
            self.factory = factory
            self.pizza = self.factory.criar_pizza()
            self.pizza_veg = self.factory.criar_pizza_veg()
            self.pizza_veg.preparar()
            self.pizza.servir(self.pizza_veg)

pizzaria = Pizzaria()
pizzaria.fazer_pizzas()
```

Preparando PizzaMandioca
PizzaCamarao é servida com camarão na PizzaMandioca
Preparando PizzaBroccoli
PizzaBolonha é servida com bolonha na PizzaBroccoli

d) Ao abrir o arquivo AbstractFactory.ipynb em um editor de texto puro como no bloco de notas do Windows ou outro software que abre .txt puro. Que tipo de código fonte é encontrado ao abrir este arquivo desta forma? Existe alguma relação com o padrão markdown? O Google Colab consegue ler arquivos com tags markdown? Fonte básico de informações sobre o tema:

https://colab.research.google.com/notebooks/markdown_guide.ipynb

O código fonte encontrado possui o código python, mas contendo diversas alterações, como a adição das células. Segue uma print:

```

"cells": [
  {
    "cell_type": "markdown",
    "source": [
      "# Laboratório: Padrões de Projeto: *Design Patterns*\n",
      "O principal objetivo do padrão Abstract Factory é for\n",
      "\n",
      "Enquanto o Factory Method adia a criação da instância\n",
      "\n",
    ]
  }
]

```

Existe uma relação com a linguagem de marcação Markdown, visto que nesse padrão há o uso de “#” para definição de títulos, a criação de listas, links, imagens e tabelas. Essas tags e marcações podem ser lidas pelo Google Colab a partir da criação de células markdown.

e) Faça uma pequena modificação no fonte AbstractFactory.ipynb inserindo e imprimindo a(s) matrícula(s) e o nome(s) do grupo. Salve as mudanças e modifique e renomeie o arquivo .ipynb. renomeie para inserir no nome + “_modificado.ipynb). Coloque a(s) matrícula(s) e nome(s) do(s) estudante(s) como uma célula de texto inicial do código fonte. Salve o arquivo .ipynb e envie dentro do arquivo compactado (zipado) com a solução do grupo para esta letra da questão.

Enviado no zip.

f) Além da extensão do nome do arquivo no formato .py da questão 2 e da extensão no formato .ipynb. Qual ou quais são as diferenças nos códigos-fontes de ambos que permitem rodá-los sem erros cada um no seu ambiente?

O arquivo no formato .py utiliza a sintaxe tradicional da linguagem Python, sem nenhuma modificação, sendo executadas pelo próprio terminal ou por programas externos como por extensões do Visual Studio Code. Já no arquivo formato .ipynb utiliza um sintaxe misturada de Python com a linguagem de marcação Markdown, sendo criadas células, listas e títulos para serem executadas no Google Colab.

g) Ao analisar o output dos três exercícios responda se existe alguma diferença na lógica de programação ou as três soluções se equivalem logicamente? (Justifique a sua resposta)

A lógica de todos parece muito similar no critério “output”, visto que quando executados, ambos apresentam a mesma saída, entretanto o formato .ipynb apresenta características muito diferentes do restante, visto que demonstra maior explicação dos códigos, deixando muito mais intuitivo o programa.

4) Responda as seguintes questões em relação ao Design Patterns: Abstract Factory.

a) O que é um Design Patterns?

Design Patterns são descrições ou modelos utilizados para resolver problemas de design e arquitetura, mantendo o código mais fácil de entender.

b) Em que contexto pode-se ser útil usar uma solução usando Design Patterns do tipo Abstract Factory? Exemplifique sem usar o exemplo do lab. da Fábrica de Pizza ou Pizzaria.

O contexto que pode ser útil é na utilização da família como um todo, ou seja, sem utilizar as classes abstratas. Um exemplo seria em um jogo que precisa rodar em diferentes plataformas (consoles, celulares e computadores), o Abstract Factory será responsável por criar objetos específicos da plataforma, como o controle, interface gráfica, ajustes de áudio, entre outros.

c) Quais os principais prós e contras de uma solução envolvendo o padrão Abstract Factory?

Quanto aos prós, facilita a adição de novos objetos (basta criar um novo Abstract Factory, sem alterar o código), permite a compatibilidade entre as “fábricas” criadas e facilita a criação de testes. Quanto aos contras, introduz um maior número de código (pode aumentar a complexidade do programa) e a dificuldade em manter a hierarquia dos “produtos e fábricas” conforme a estrutura fica mais complexa.

d) Quais são as relações deste padrão Abstract Factory com outros padrões de projeto?

Obs.: Existem várias informações sobre Padrões de Projetos na internet. Boas fontes básicas de informações sobre este padrão para responder estas questões podem ser encontradas no link abaixo: <https://refactoring.guru/pt-br/design-patterns/abstract-factory>

Seguem as relações:

-.java: O factory method é usado para criar e instanciar produtos, retirando a necessidade de usar as interfaces padrões do java.

-.py: É usado para adicionar funcionalidades e comportamentos dinâmicos dos produtos, além de implementar e criar produtos.

-.ipynb: Principalmente em visualizações complexas de imagem, podendo facilitar o entendimento das células.