1. **SOLID – O – Open/Closed**

class Restaurante:

    def \_\_init\_\_(self, nombre):

        self.nombre = nombre

    def realizar\_pedido(self, tipo\_pedido, detalles):

        if tipo\_pedido == "para\_llevar":

            # Lógica para manejar pedidos para llevar

            print(f"Procesando pedido para llevar: {detalles}")

        elif tipo\_pedido == "comer\_en\_local":

            # Lógica para manejar pedidos para comer en el local

            print(f"Procesando pedido para comer en el local: {detalles}")

        elif tipo\_pedido == "entrega\_a\_domicilio":

            # Lógica para manejar pedidos de entrega a domicilio

            print(f"Procesando pedido de entrega a domicilio: {detalles}")

        else:

            print("Tipo de pedido no válido")

from abc import ABC, abstractmethod

# clase maestra

class ManejadorPedidos(ABC):

    @abstractmethod

    def procesar\_pedido(self, detalles):

        pass

class PedidoParaLlevar(ManejadorPedidos):

    def procesar\_pedido(self, detalles):

        print(f"Procesando pedido para llevar: {detalles}")

class PedidoLocal(ManejadorPedidos):

    def procesar\_pedido(self, detalles):

        print(f"Procesando pedido para comer en el local: {detalles}")

class PediEntregaADomicilio(ManejadorPedidos):

    def procesar\_pedido(self, detalles):

        print(f"Procesando pedido para entrega a domicilio: { detalles}")

class PedidoEspecial(ManejadorPedidos):

    def procesar\_pedido(self, detalles):

        print(f"Procesando pedido para evento especial: {detalles}")

class Restaurante:

    def \_\_init\_\_(self, nombre) -> None:

        self.nombre = nombre

        self.manejadores\_pedido = []

    def registrar\_pedidos(self, tipo\_pedido):

        self.manejadores\_pedido.append(tipo\_pedido)

    def realizar\_pedido(self, tipo\_pedido, detalles):

        tipo\_pedido.procesar\_pedido(detalles)

restaurante = Restaurante("Mi restaurante de pastas ")

restaurante.registrar\_pedidos(PedidoParaLlevar()

restaurante.registrar\_pedidos(PedidoEspecial())

restaurante.realizar\_pedido(PedidoParaLlevar(), "Plato de pasta grande")

restaurante.realizar\_pedido(PedidoEspecial(), "Plato especial de mariscos")

class Superhero:

    def \_\_init\_\_(self, name, health, attack\_type):

        self.name = name

        self.health = health

        self.attack\_type = attack\_type

    def attack(self):

        if self.attack\_type == "punch":

            return f"{self.name} attacks with a powerful punch!"

        elif self.attack\_type == "laser":

            return f"{self.name} attacks with a laser beam!"

        else:

            return f"{self.name} attacks with a regular attack!"

class Game:

    def \_\_init\_\_(self):

        self.superheroes = []

    def add\_superhero(self, superhero):

        self.superheroes.append(superhero)

    def superhero\_actions(self):

        for superhero in self.superheroes:

            print(superhero.attack())

game = Game()

superhero1 = Superhero("Superman", 100, "punch")

superhero2 = Superhero("Cyclops", 80, "laser")

game.add\_superhero(superhero1)

game.add\_superhero(superhero2)

game.superhero\_actions()

from abc import ABC, abstractmethod

class atackManager(ABC):

    @abstractmethod

    def attack(self, superhero):

        pass

# Implementaciones concretas

class PunchAttack(atackManager):

    def attack(self, superhero):

        return f"{superhero.name} attacks with a powerfull punch!!!! "

class LaserAttack(atackManager):

    def attack(self, superhero):

        return f"{superhero.name} attacks with a long laser!!!"

class FireballAttack(atackManager):

    def attack(self, superhero):

        return f"{superhero.name} attacks with a big fire ball!!!"

class IceballAttack(atackManager):

    def attack(self, superhero):

        return f"{superhero.name} attacks with a big ice ball!!!"

class Superhero:

    def \_\_init\_\_(self, name, health, atackManager) -> None:

        self.name = name

        self.health = health

        self.atackManager = atackManager

    def attack(self):

        return self.atackManager.attack(self)

class Game:

    def \_\_init\_\_(self) -> None:

        self.superheroes = []

    def add\_superheroes(self, superhero):

        self.superheroes.append(superhero)

    def superheroe\_action(self):

        for superhero in self.superheroes:

            print(superhero.attack())

game = Game()

superman = Superhero("Superman", 100, PunchAttack())

cyclops = Superhero("Cyclops", 80, LaserAttack())

fireman = Superhero("Fireman", 120, PunchAttack())

iceman = Superhero("Iceman", 120, IceballAttack())

game.add\_superheroes(superman)

game.add\_superheroes(cyclops)

game.add\_superheroes(fireman)

game.add\_superheroes(iceman)

game.superheroe\_action()