**Activity: Classify, Justify, Design!**

📂 **Filename:** python\_activity18.py

**🎯 Objective:**

You will identify whether a given design **adheres to** or **violates** the **Open-Closed Principle (OCP)**. Then, you will **justify your answer** and **propose a better design** if a violation exists.

**📝 Instructions:**

1. You will receive **a set of items** in different formats:
   * Short Python code snippets
   * Software feature requests
   * Class or system design descriptions
2. For **each item**, do the following:
   * ✅ **Classify**: Does it follow the Open-Closed Principle?
   * 💬 **Justify**: Write **1–2 sentences** explaining your reasoning.
   * 🔁 **Design**: For each item that violates the Open-Closed Principle, do the following:
     1. Briefly explain why it violates OCP.
     2. Refactor the code so that it follows OCP.
3. Work individually or in pairs. You may discuss, but your analysis must be your own.

Here’s your **Set of Items** for the **OCP Activity:**

**🧩 Item 1 – Shape Area Calculator**

class AreaCalculator:

def calculate\_area(self, shape):

if shape['type'] == 'circle':

return 3.14 \* shape['radius'] \*\* 2

elif shape['type'] == 'rectangle':

return shape['width'] \* shape['height']

**🧩 Item 2 – Message Sender**

from abc import ABC, abstractmethod

class Message(ABC):

@abstractmethod

def send(self, text):

pass

class EmailMessage(Message):

def send(self, text):

print(f"Sending Email: {text}")

class SMSMessage(Message):

def send(self, text):

print(f"Sending SMS: {text}")

**🧩 Item 3 – Discount Strategy (Incoming Feature)**

class Checkout:

def apply\_discount(self, total\_price, discount\_type):

if discount\_type == "student":

return total\_price \* 0.9

elif discount\_type == "senior":

return total\_price \* 0.85

**🧩 Item 4 – Report Generator**

class Report:

def generate\_pdf(self, data):

print("Generating PDF...")

def generate\_csv(self, data):

print("Generating CSV...")

def generate\_html(self, data):

print("Generating HTML...")

**🧩 Item 5 – Logger**

class Logger:

def log(self, message):

print(f"[LOG] {message}")