**📝 Activity: Interface Segregation Principle**

**🎯 Objective:**

Understand and apply the Interface Segregation Principle (ISP) by analyzing and refactoring a flawed Python design using smaller, more specific interfaces.

**📌 Instructions:**

1. **Study the Given Code Below:**

from abc import ABC, abstractmethod

# Monolithic interface

class SmartDevice(ABC):

@abstractmethod

def turn\_on(self):

pass

@abstractmethod

def turn\_off(self):

pass

@abstractmethod

def change\_channel(self, channel):

pass

@abstractmethod

def set\_temperature(self, temp):

pass

@abstractmethod

def start\_wash\_cycle(self):

pass

class SmartTV(SmartDevice):

def turn\_on(self):

print("TV is now ON")

def turn\_off(self):

print("TV is now OFF")

def change\_channel(self, channel):

print(f"Changing channel to {channel}")

def set\_temperature(self, temp):

raise NotImplementedError("TV can't set temperature")

def start\_wash\_cycle(self):

raise NotImplementedError("TV can't wash clothes")

class SmartFridge(SmartDevice):

def turn\_on(self):

print("Fridge is now ON")

def turn\_off(self):

print("Fridge is now OFF")

def change\_channel(self, channel):

raise NotImplementedError("Fridge can't change channels")

def set\_temperature(self, temp):

print(f"Fridge temperature set to {temp}°C")

def start\_wash\_cycle(self):

raise NotImplementedError("Fridge can't wash clothes")

class SmartWasher(SmartDevice):

def turn\_on(self):

print("Washer is now ON")

def turn\_off(self):

print("Washer is now OFF")

def change\_channel(self, channel):

raise NotImplementedError("Washer can't change channels")

def set\_temperature(self, temp):

raise NotImplementedError("Washer doesn't control temperature like a fridge")

def start\_wash\_cycle(self):

print("Wash cycle started")

1. **Analyze the Problem:**
   * Why is this code violating the Interface Segregation Principle?
   * Which classes are forced to implement methods they don’t use?
   * What’s the problem with using raise NotImplementedError()?
2. **Refactor the Code:**
   * Create **smaller interfaces** (abstract classes)
   * Let each device class implement **only the interfaces it needs**.
   * Refactor so that **no method should be NotImplementedError**.
3. **Enhance the Program:**
   * Add **one new device class** of your choice (e.g., SmartOven, SmartFan, SmartSpeaker) and implement only the relevant interface(s).
4. **Submit Your Work:**
   * File 1: python\_activity21(GivenCode).py (copy the code above)
   * File 2: Python\_Activity21(refactored\_solution).py (your corrected and improved version)