# תרגיל בית 3

**שאלה 1** (25 נק')

להלן מימוש של תבנית העיצוב Facade עבור רובוט שמעביר חפץ ממקום למקום תוך שימוש במצלמה ויד רובוטית עם צבת (פלייר)

```
# Subsystem: Camera
class Camera:
    def detect_object(self):
        print("Camera: Detecting object...")
        return "Object detected at position (10, 15)"

    def detect_destination(self):
        print("Camera: Detecting destination...")
        return "Destination detected at position (50, 75)"
```

```
# Subsystem: Robot Arm with Plier
class RobotArmWithPlier:
    def move_to_position(self, position):
        print(f"Robot Arm: Moving to position {position}...")

    def grab_object(self):
        print("Robot Arm: Grabbing object...")

    def release_object(self):
        print("Robot Arm: Releasing object...")
```

```
# Facade: Robot System
class RobotFacade:
    def __init__(self):
        self.camera = Camera()
        self.robot_arm = RobotArmWithPlier()

    def move_object(self):
        object_position = self.camera.detect_object()

        self.robot_arm.move_to_position(object_position)
        self.robot_arm.grab_object()

        destination_position = self.camera.detect_destination()

        self.robot_arm.move_to_position(destination_position)
        self.robot_arm.release_object()
```

הפעלה של השורות

```
robot = RobotFacade()
robot.move_object()
```

תחזיר

```
Camera: Detecting object...
Robot Arm: Moving to position Object detected at position (10,15)...
Robot Arm: Grabbing object...
Camera: Detecting destination...
Robot Arm: Moving to position Destination detected at position (50, 75)...
Robot Arm: Releasing object...
```

נשנה כעת את המחלקות של תתי המערכות לעבוד עם פונקציות סטטיות

```
# Subsystem: Camera
class Camera:
  @staticmethod
  def detect_object():
    print("Camera: Detecting object...")
    return "Object detected at position (10, 15)"

  @staticmethod
  def detect_destination():
    print("Camera: Detecting destination...")
    return "Destination detected at position (50, 75)"
```

```
# Subsystem: Robot Arm with Plier
class RobotArmWithPlier:
  @staticmethod
  def move_to_position(position):
    print(f"Robot Arm: Moving to position {position}...")

  @staticmethod
  def grab_object():
    print("Robot Arm: Grabbing object...")

  @staticmethod
  def release_object():
    print("Robot Arm: Releasing object...")
```

שנו באופן דומה את המחלקה RobotFacade כך שהפונקציה move_object תהיה סטטית והפעלת המערכת ע"י המשתמש תתבצע ע"י:

```
RobotFacade.move_object()
```

**שאלה 2** (25 נק')

להלן מימוש חלקי של תבנית העיצוב Composite עבור אריזות ופריטים. העלה הוא פריט ואריזה (חבילה) היא המחלקה המורכבת, יכולה להכיל בתוכה חבילות נוספות.

```
class PackageComponent:

    def get_price(self):
        pass

    def show_details(self, indent=0):
        pass
```

```
class Item(PackageComponent):
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def get_price(self):
        return self.price

    def show_details(self, indent=0):
        print(f"Item: {self.name}, Price: {self.price}")
```

- השלימו את הקוד של המחלקה החסרה.

```
class Package(PackageComponent):
    def __init__(self, name):
        self.name = name
        self.components = []

    def add_component(self, component: PackageComponent):
            ____________________

    def remove_component(self, component: PackageComponent):
            ____________________

    def get_price(self):
            ____________________

    def show_details(self, indent=0):
            ____________________
```

- ודאו כי הרצת השורות

```
#Create Items
item1 = Item("Laptop", 1200)
item2 = Item("Mouse", 25)
item3 = Item("Keyboard", 75)
item4 = Item("Monitor", 200)

# Create a package and add items
package1 = Package("Office Essentials")
package1.add_component(item2)
package1.add_component(item3)

# Create another package and add items/packages
main_package = Package("Workstation Setup")
main_package.add_component(item1)
main_package.add_component(item4)
main_package.add_component(package1)

# Show details and total price
main_package.show_details()
print(f"Total Price of Workstation Setup: {main_package.get_price()}")
```

מציגה את הפלט הבא

```
Package: Workstation Setup, Total Price: 1500
Item: Laptop, Price: 1200
Item: Monitor, Price: 200
Package: Office Essentials, Total Price: 100
Item: Mouse, Price: 25
Item: Keyboard, Price: 75
Total Price of Workstation Setup: 1500
```

**שאלה 3** (30 נק')

נתון הקוד הבא המציג מימוש של Singelton עבור תור הדפסות למדפסת תוך שימוש ב __new__.

Printer יכול להוסיף מסמך לתור וגם להדפיס את המסמך שנמצא בראש התור.

```
class PrinterQueue:
    _instance = None

    def __new__(cls):
        if cls._instance is None:
            cls._instance = super().__new__(cls)
            cls._instance.queue = []
        return cls._instance

    def add_to_queue(self, document):
        self.queue.append(document)
        print(f"Added '{document}' to the queue.")

    def print_next(self):
        if self.queue:
            document = self.queue.pop(0)
            print(f"Printing: {document}")
        else:
            print("The queue is empty. No documents to print.")

    def view_queue(self):
        print("Current queue:", self.queue)
```

להלן דוגמה להרצת הקוד

```
# First instance
printer1 = PrinterQueue()
printer1.add_to_queue("Document1.pdf")
printer1.add_to_queue("Document2.pdf")

# Second instance
printer2 = PrinterQueue()
printer2.view_queue()

# Demonstrating the singleton property
printer2.print_next()
printer1.view_queue()

print(printer1 is printer2)
print(printer1.__dict__)
print(printer2.__dict__)
```

והפלט המתקבל:

```
Added 'Document1.pdf' to the queue.
Added 'Document2.pdf' to the queue.
Current queue: ['Document1.pdf',
'Document2.pdf']
Printing: Document1.pdf
Current queue: ['Document2.pdf']

True
{'queue': ['Document2.pdf']}
{'queue': ['Document2.pdf']}
```

<u>סעיף א:</u>
ממשו את הקוד ע"י מחלקה בתוך מחלקה (השיטה הראשונה למימוש סינגלטון שראינו בכיתה)

<u>סעיף ב:</u>
הסבירו מדוע תוצאות ההדפסה של שלשת השורות

```
print(printer1 is printer2)
print(printer1.__dict__)
print(printer2.__dict__)
```

שונות.

<u>סעיף ג:</u>
הדפיסו את המילון של האובייקט המחזיק את

```
{'queue': ['Document2.pdf']}
```

במילים אחרות - במילון של איזה אובייקט יושב עכשיו התור?
בצעו הדפסה מתאימה.

**<u>שאלה 4</u>** (20 נק')

הקובץ MediatorThreeParties מכיל מימוש של Mediator לטווח בין שלש קבוצות, כאן שלש מדינות.

עברו על הקוד וודאו כי אתם מבינים אותו, שימו לב שניתן לשלוח הודעה לאחת הקבוצות או לכל הקבוצות.

הוסיפו ארבע שורות המריצות את הקוד ובהן מתבצע:

- CountryA שולח ל- CountryB

"We propose a ceasefire agreement".

- CountryB שולח ל- CountryC

"We agree to the ceasefire. Let's involve CountryC."

- CountryC שולח לכולם

"We support the ceasefire. Let's finalize the terms."

- CountryA שולח לכולם

"Thank you all. Let's sign the agreement and ensure peace."

# אופן הגשת תרגיל הבית

יש להגיש את התרגיל בקבוצות עד לתאריך 18.12.24 בשעה 23:59.

יש להגיש קובץ zip **יחיד** המכיל תקייה אחת בשם groupXX לדוגמא group01 או group21

התקייה תכיל את 4 קבצי הקוד שהעלינו למודל ועוד קובץ PDF עבור שאלה 3 סעיף ב'

הנכם מוזמנים להיעזר בפורום הקורס לשאול שאלות ולעקוב אחר תשובות שנענו עבור שאלות קודמות.

**בהצלחה!**