**Assignment 4**

class AvailabilityCheckControl:

    """

    Manages the process of checking the availability of dates.

    """

    def \_\_init\_\_(self, dates):

        # Initialize with a list of dates and set the initial availability status to False

        self.\_\_availability\_status = False

        self.\_\_dates = dates  # List of Date objects

    def check\_availability(self, date):

        """

        Check the availability of the provided date.

        Returns True if the date is available, otherwise False.

        """

        if date in self.\_\_dates:

            self.\_\_availability\_status = True

            print(f"Date {date} is available.")

            return True

        else:

            print(f"Date {date} is not available.")

            return False

import pandas as pd

class ExcelExportControl:

    """

    Manages the export of data to Excel files.

    """

    def \_\_init\_\_(self, file\_path):

        # Initialize with the file path where the Excel file will be saved

        self.\_\_file\_path = file\_path

        self.\_\_users = []  # Placeholder for the list of users

    def export\_to\_excel(self, data):

        """

        Export the provided data to an Excel file.

        """

        if data:

            df = pd.DataFrame(data)

            df.to\_excel(self.\_\_file\_path, index=False)

            print(f"Data exported to {self.\_\_file\_path}")

        else:

            raise ValueError("Data must not be null.")

class LoginControl:

    """

    Handles the login process for user accounts.

    """

    def \_\_init\_\_(self, accounts):

        # Initialize with a list of accounts and set the initial login status to False

        self.\_\_login\_status = False

        self.\_\_accounts = accounts  # List of Account objects

    def login(self, username, password):

        """

        Attempt to log in with the provided username and password.

        If the credentials match an account, set login status to True.

        """

        for account in self.\_\_accounts:

            if account.get\_username() == username and account.get\_password() == password:

                self.\_\_login\_status = True

                print(f"Login successful for user: {username}")

                return True

        print("Login failed. Invalid credentials.")

        return False

    def logout(self):

        """

        Log out the currently logged-in user.

        """

        if self.\_\_login\_status:

            self.\_\_login\_status = False

            print("User logged out successfully.")

        else:

            print("No user is currently logged in.")

    def is\_logged\_in(self):

        """

        Check if a user is currently logged in.

        """

        return self.\_\_login\_status

class PriceCheckControl:

    """

    Manages the process of checking product prices.

    """

    def \_\_init\_\_(self, products):

        # Initialize with a list of products and set the current price to None

        self.\_\_current\_price = None

        self.\_\_products = products  # List of Product objects

    def check\_price(self, product\_url):

        """

        Check the price of the product at the provided URL.

        Returns the price if found, otherwise raises an exception.

        """

        for product in self.\_\_products:

            if product.get\_url() == product\_url:

                self.\_\_current\_price = self.fetch\_price\_from\_url(product\_url)

                print(f"Price checked: {self.\_\_current\_price} for URL: {product\_url}")

                return self.\_\_current\_price

        raise ValueError(f"Product not found for URL: {product\_url}")

    def fetch\_price\_from\_url(self, product\_url):

        """

        Simulates fetching the price from a URL. In a real scenario, this would involve web scraping.

        """

        # Placeholder logic for price fetching

        return 123.45  # Example price

    def get\_current\_price(self):

        """

        Return the current price of the product.

        """

        return self.\_\_current\_price

class NotificationControl:

    """

    Manages notifications for users.

    """

    def \_\_init\_\_(self):

        # Initialize with an empty list of notifications

        self.\_\_notifications = []

    def send\_notification(self, notification):

        """

        Send a notification to the user and add it to the list of notifications.

        """

        if notification:

            self.\_\_notifications.append(notification)

            print(f"Notification sent: {notification.get\_content()}")

        else:

            raise ValueError("Notification cannot be null.")

    def get\_notifications(self):

        """

        Return the list of sent notifications.

        """

        return self.\_\_notifications

Oguz Kaan Yildirim