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