Assignment 4

```
import requests
from bs4 import BeautifulSoup


class BrowserInterface:
    """
    Handles interactions between the bot and the browser for scraping web data.
    """

    def __init__(self, browser_type, url):
        # Initialize the browser interface with browser type and URL
        self.__browser_type = browser_type
        self.__url = url

    def launch_browser(self):
        # Launch the browser with the specified URL
        if self.__url:
            print(f"Launching {self.__browser_type} browser with URL: {self.__url}")
        else:
            raise ValueError("URL must not be null.")

    def close_browser(self):
        # Close the browser
        print(f"Closing {self.__browser_type} browser.")
        self.__browser_type = None
        self.__url = None

    def login(self, account):
        # Use the Account class to log in to a website
        if account.get_username() and account.get_password():
            print(f"Logging in with username: {account.get_username()}")
```

```
        # Placeholder for actual login logic using account credentials
        # This would involve interacting with the web page elements to enter the username and password
    else:
        raise ValueError("Account credentials must not be null.")

  def display_data_in_html(self, data):
    # Create an HTML page to display the data read from Excel
    html_content = "<html><head><title>Product Data</title></head><body>"
    html_content += "<h1>Product Data</h1><table border='1'>"
    html_content += "<tr><th>Timestamp</th><th>URL</th><th>Price</th><th>Product</th></tr>"

    for row in data:
        html_content += f"<tr><td>{row['Timestamp']}</td><td>{row['URL']}</td><td>{row['Price']}</td><td>{row['Product']}</td></tr>"

    html_content += "</table></body></html>"

    # Save the HTML content to a file
    with open("product_data.html", "w") as file:
        file.write(html_content)

    print("Data displayed in HTML page (product_data.html).")
```

```
class DateInfoInterface:
    """
    Manages the input and output for date availability requests.
    """

    def __init__(self):
        # Initialize with empty date info
        self.__date_info = ""

    def fetch_date_info(self, date):
        # Fetch date information (Placeholder for actual date info retrieval logic)
        if date:
            self.__date_info = f"Availability checked for {date}"
            print(self.__date_info)
        else:
            raise ValueError("Date must not be null.")

    def get_date_info(self):
        # Return the fetched date information
        if self.__date_info:
            return self.__date_info
        else:
            raise ValueError("Date information has not been fetched yet.")
```

```
import discord

class DiscordInterface:
    """
    Manages the interactions between the bot and the user on Discord.
    """

    def __init__(self, interface_name, discord_bot):
        # Initialize with the interface name and Discord bot instance
        self.__interface_name = interface_name
        self.__discord_bot = discord_bot

    def connect(self):
        # Connect the bot to Discord
        if self.__discord_bot:
            print(f"Connecting {self.__interface_name} to Discord...")
            self.__discord_bot.run()
        else:
            raise ValueError("Discord bot instance must not be null.")

    def disconnect(self):
        # Disconnect the bot from Discord
        if self.__discord_bot:
            self.__discord_bot.close()
            print(f"Disconnecting {self.__interface_name} from Discord...")
        else:
            raise ValueError("Discord bot instance must not be null.")
```

```
import pandas as pd
from datetime import datetime

class ExcelInterface:
    """
    Handles data extraction to and from Excel files.
    """

    def __init__(self, file_path):
        # Initialize with the file path where the Excel file will be saved
        self.__file_path = file_path

    def save_data_to_excel(self, data):
        # Save the data to an Excel file with additional details
        if data:
            df = pd.DataFrame(data)
            df['Timestamp'] = datetime.now()  # Add a timestamp column
            df.to_excel(self.__file_path, index=False)
            print(f"Data saved to {self.__file_path}")
        else:
            raise ValueError("Data must not be null.")

    def load_data_from_excel(self):
        # Load data from an Excel file
        try:
            data = pd.read_excel(self.__file_path).to_dict(orient="records")
            print(f"Data loaded from {self.__file_path}")
            return data
        except Exception as e:
            print(f"Failed to load data from Excel: {e}")
            return None
```

```
class ProductInfoInterface:
    """
    Manages the input and output for product information requests.
    """

    def __init__(self):
        # Initialize with empty product details
        self.__product_details = ""

    def fetch_product_info(self, product_url):
        # Fetch product details from the URL (Placeholder for actual scraping logic)
        if product_url:
            self.__product_details = f"Details fetched from {product_url}"
            print(self.__product_details)
        else:
            raise ValueError("Product URL must not be null.")

    def get_product_details(self):
        # Return the fetched product details
        if self.__product_details:
            return self.__product_details
        else:
            raise ValueError("Product details have not been fetched yet.")
```

Oguz Kaan Yildirim