Assignment 4

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

    def __init__(self, resource_url):
        # Initialize with the resource URL
        self.resource_url = resource_url
        self.available_dates = []

    def get_available_dates(self):
        # Return the fetched dates
        if not self.available_dates:
            self.fetch_available_dates()
        return self.available_dates
```

```
import requests
from bs4 import BeautifulSoup

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

    def __init__(self, url):
        # Store the URL to scrape data from
        self.url = url
        self.page_content = None

    def fetch_page(self):
        # Fetch the content of the webpage
        try:
            response = requests.get(self.url)
            response.raise_for_status()  # Check for request errors
            self.page_content = response.text
        except requests.exceptions.RequestException as e:
            print(f"Failed to fetch the page: {e}")
            self.page_content = None

    def parse_page(self):
        # Parse the webpage content using BeautifulSoup
        if self.page_content:
            soup = BeautifulSoup(self.page_content, 'html.parser')
            return soup
        else:
            print("No content to parse")
            return None
```

```
import pandas as pd
class ExcelInterface:
    """
    Handles data extraction to Excel.
    """
    def __init__(self, file_path):
        # Initialize with the file path where the Excel file will be saved
        self.file_path = file_path
        self.data = None
    def save_data_to_excel(self, data):
        # Save the data to an Excel file
        try:
            df = pd.DataFrame(data)
            df.to_excel(self.file_path, index=False)
            print(f"Data saved to {self.file_path}")
        except Exception as e:
            print(f"Failed to save data to Excel: {e}")
    def load_data_from_excel(self):
        # Load data from an Excel file
        try:
            self.data = pd.read_excel(self.file_path)
            print(f"Data loaded from {self.file_path}")
        except Exception as e:
            print(f"Failed to load data from Excel: {e}")
            self.data = None

    def get_data(self):
        # Return the loaded data
        if self.data is None:
            self.load_data_from_excel()
        return self.data
```

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

    def __init__(self, product_url):
        # Initialize with the product URL
        self.product_url = product_url
        self.product_details = {}

    def fetch_product_details(self):
        # Pretend to fetch product details from the URL (placeholder)
        # In a real scenario, you'd scrape the page and extract details
        self.product_details = {
            'name': 'Sample Product',
            'price': '123.45',
            'availability': 'In Stock'
        }

    def get_product_details(self):
        # Return the fetched product details
        if not self.product_details:
            self.fetch_product_details()
        return self.product_details
```

```
class DiscordInterface:
    """
    Manages the interactions between the bot and the user on Discord.
    """
    def __init__(self, message):
        # Store the message received from the user
        self.message = message
        # Store the message as the command directly
        self.command = message
        self.response = None
    def generate_response(self):
        # Generate a response based on the command
        # Mostly examples to give ideas, will be dynamic and changed soon
        if self.command.lower() == 'hello':
            self.response = 'Hey there!'
        elif self.command.lower() == 'help':
            self.response = 'These are the commands you can use: hello, help, latest price, last checked, share url...'
        elif self.command.lower() == 'latest price':
            # Placeholder for fetching the latest price from the system
            self.response = 'The latest price for the tracked product is $123.45.'
        elif self.command.lower() == 'last checked':
            # Placeholder for fetching the last checked time
            self.response = 'The last time we checked the price was at 10:30 AM, August 10, 2024.'
        elif self.command.lower().startswith('share url'):
            # Assuming the URL is shared in the format "share url <url>"
            url = self.command[len('share url'):].strip()
            # Placeholder response for URL processing
            self.response = f'Thank you for sharing the URL: {url}. We are fetching the details...'
        else:
            self.response = 'Sorry, I didn't understand that command.'
```

return self.response

Oguz Kaan Yildirim