Netcentric lab 7

Nguyen Manh Viet Khoi
ITCSIU21081

```go
package main

import (
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "net/http"
    "os"

    "golang.org/x/net/html"
)

type Manga struct {
    Title string `json:"title"`
    Link  string `json:"link"`
}

type Genre struct {
    Name  string  `json:"name"`
    Manga []Manga `json:"manga"`
}

func main() {
    genres := []string{"drama", "fantasy", "comedy",
"action", "slice_of_life", "romance", "thriller",
"horror", "superhero", "sports"}
```

```
    var results []Genre

    for _, genre := range genres {
        mangaList := fetchMangaTitles(genre)
        results = append(results, Genre{Name: genre,
Manga: mangaList})
    }

    file, _ := json.MarshalIndent(results, "", " ")
    _ = os.WriteFile("manga_titles.json", file, 0644)
}

func fetchMangaTitles(genre string) []Manga {
    url := fmt.Sprintf("https://www.webtoons.com/en/%s",
genre)
    resp, err := http.Get(url)
    if err != nil {
        fmt.Println("Error fetching genre page:", err)
        return nil
    }
    defer resp.Body.Close()

    body, err := io.ReadAll(resp.Body)
    if err != nil {
        fmt.Println("Error reading response body:", err)
        return nil
    }

    doc, err := html.Parse(bytes.NewReader(body))
    if err != nil {
        fmt.Println("Error parsing HTML:", err)
        return nil
```

```
    }

    return findMangaTitles(doc)
}

func findMangaTitles(n *html.Node) []Manga {
    var mangaList []Manga
    var walk func(*html.Node)
    count := 0

    walk = func(n *html.Node) {
        if count >= 10 {
            return
        }
        if n.Type == html.ElementNode && n.Data == "a" {
            var link, title string
            for _, attr := range n.Attr {
                if attr.Key == "href" {
                    link = attr.Val
                }
            }
            for c := n.FirstChild; c != nil; c =
c.NextSibling {
                if c.Type == html.ElementNode && c.Data ==
"div" {
                    for gc := c.FirstChild; gc != nil; gc
= gc.NextSibling {
                        if gc.Type == html.ElementNode &&
gc.Data == "p" && hasClass(gc, "subj") {
                            if gc.FirstChild != nil &&
gc.FirstChild.Type == html.TextNode {
                                title = gc.FirstChild.Data
```

```
                            }
                        }
                    }
                }
            }
            if link != "" && title != "" {
                mangaList = append(mangaList, Manga{Title:
title, Link: link})
                count++
            }
        }
        for c := n.FirstChild; c != nil; c = c.NextSibling
{
            walk(c)
        }
    }

    walk(n)
    return mangaList
}

func hasClass(n *html.Node, class string) bool {
    for _, attr := range n.Attr {
        if attr.Key == "class" && attr.Val == class {
            return true
        }
    }
    return false
}
```

Explain

This Go program is a web scraper that fetches manga titles from the website "https://www.webtoons.com/en/" for different genres and writes the results to a JSON file. Here's a breakdown of the code:

1. Two struct types, Manga and Genre, are defined to hold the scraped data. Manga has fields for the title and link of a manga, while Genre has a name and a slice of Manga.
2. The main function defines a slice of genres and then iterates over this slice. For each genre, it calls the fetchMangaTitles function and appends the result (a slice of Manga) to the results slice. The result slice is then written to a JSON file named "manga_titles.json".
3. The fetchMangaTitles function takes a genre as an argument, constructs a URL for that genre, and sends a GET request to that URL. It reads the response body and parses it as HTML. It then calls the findMangaTitles function with the parsed HTML document.
4. The findMangaTitles function traverses the HTML document using a recursive function walk. It looks for elements and extracts the href attribute and the title of the manga from the child elements. It appends each found manga to the mangaList slice and returns this slice when it has found 10 mangas or has traversed the entire document.
5. The hasClass function checks if a given HTML node has a specific class. It's used in findMangaTitles to identify the p element that contains the title of the manga.