```
/*  Programming for Design
    Project 2 - Book Information Website
    Keir Herbert (u3211239)
    October 2020
*/

//  For the group - Keir Herbert, Michael Coward, & Sujith Kumar.

class bookCover {
//  This class returns the bookcover for an ISBN number in correct HTML using the openlibrary API.

    constructor(isbn, size = 'S', key = 'isbn') {                          //  Define the accepted input parameters and some default values for them.
        this.isbn = isbn;                                                  //  Pass the input values into the class properties.
        this.size = '-' + size;                                            //  Prefix the size value to meet syntax requirements.
        this.key = key.toUpperCase() + "/";                                //  Ensure that the key property is in uppercase and append the necessary slash.
        this.url_a = '<img src="http://covers.openlibrary.org/b/'          //  Define the first part of the URL for the book cover.
        this.url_b = '.jpg" />'                                            //  Define the last part of the URL for the book cover.
    }

    display() {
//  This method returns the correct HTML for this book cover.
//  Combine the two URL ends with the key, ISBN, and size properties to form a valid URL.
        var bookCoverURL = this.url_a + this.key + this.isbn + this.size + this.url_b;
        return bookCoverURL;                                               //  Return the assembled variable.
    }
}
```

```
class bookDetail {
//  This class returns the book details for an ISBN number in correct HTML using the openlibrary API.

    constructor(isbn, size = 'S', key = 'isbn') {                          //  Define the accepted input parameters and some default values for them.
        this.isbn = isbn;                                                  //  Pass the input values into the class properties.
        this.key = key.toUpperCase() + ":";                                //  Ensure that the key property is in uppercase and append the necessary colon.
        this.bc = new bookCover(isbn, size, key);                          //  Pass the output from the bookCover class into the bc property.
        this.url_a = 'https://openlibrary.org/api/books?bibkeys=';         //  Define the first part of the URL for collecting the JSON object.
        this.url_b = '&format=json&jscmd=data';                            //  Define the last part of the URL for collecting the JSON object.
//  Example of the target URL format: 'https://openlibrary.org/api/books?bibkeys=ISBN:0201558025&format=json'
        this.detail = "";                                                  //  Make available the detail property and set it to null.
    }

    size(val = "S") {                                                      //  Provide a method to alter the book cover size. Default to small.
        this.bc.size(val);                                                 //  Pass the input value into the bc sub-class property.
    }

    cover() {                                                              //  Provide a method to return the book cover image.
        return this.bc.display();
    }

    async getDetail() {
//  This method calls the getBookDetail function to collect the JSON object data.
        let dets = await getBookDetail(this.url_a, this.key, this.isbn, this.url_b);
        this.detail = dets[this.key + this.isbn];                          //  Populate this.detail with returned data.
    }
```

```
    getAuthor() {
//  Return the book's authors from the JSON object.
    var listOfAuthors = "";                                                 //  Create the listOfAuthors variable and set it to null.
        for (var authorInstance = 0;; authorInstance++) {                   //  Setup a loop with no end condition (it will always end in a caught error).
            try {                                                           //  Attempt to collect the author value from the JSON object and append it to any
previous ones.
                listOfAuthors = listOfAuthors + this.detail['authors'][authorInstance]['name'] + "<br>";
            }
            catch(err) {                                                    //  Catch the failed attempt to get the author value.
                if (authorInstance == 0) {                                  //  Has it failed on the first attempt?
                    return "No authors found";                              //  If so, return a "No authors found" response.
                }
                else {                                                      //  Otherwise, return the list of authors.
                    return listOfAuthors;
                }
            }
        }
    }


    getTitle() {
//  Return the book's title from the JSON object.
        var bookTitle = this.detail['title'];                               //  Create a new variable holding the value of the book's title.
        return bookTitle;                                                   //  Return the variable as the result.
    }


    getPublisher() {
//  Return the publisher(s) of the book from the JSON object.
```

```
    var listOfPublishers = "";                                          //  Create the listOfPublishers variable and set it to null.
    for (var publisherInstance = 0;; publisherInstance++) {             //  Setup a loop with no end condition (it will always end in a caught error).
        try {                                                           //  Attempt to collect the publisher value from the JSON object and append it to any
previous ones.
            listOfPublishers = listOfPublishers + this.detail['publishers'][publisherInstance]['name'] + "<br>";
        }
        catch(err) {                                                    //  Catch the failed attempt to get the publisher value.
            if (publisherInstance == 0) {                               //  Has it failed on the first attempt?
                return "No publishers found";                           //  If so, return a "No publishers found" response.
            }
            else {
                return listOfPublishers;                                //  Otherwise, return the list of publishers.
            }
        }
    }
    }


    getPublishedDate() {
//  Return the book's date of publication from the JSON object.
        var publishedDate = this.detail['publish_date'];                //  Collect the published date from the JSON object.
        if (publishedDate == null) {                                    //  Check to see if a value was returned.
            return "N/A";                                               //  If it wasn't, return an "N/A" string.
        }
        else {
        return publishedDate;                                           //  Otherwise, return the published date.
        }
    }
```

```
    getPageCount() {
//  Return the book's page count from the JSON object.
        var pageCount = this.detail['number_of_pages'];                    //  Collect the page count from the JSON object.
        if (Number.isInteger(pageCount)) {                                 //  Check to see if an integer is actually returned.
            return pageCount;                                              //  If so, return that value as the result.
        }
        else {
            return "N/A";                                                  //  Otherwise return an "N/A" string.
        }
    }
}


class movieInfo {
//  This class returns movie information.
//  Example API request https://api.themoviedb.org/3/movie/550?api_key=8ed7c07ca0c6eb792d78a3b9086aff66
//  Example query https://api.themoviedb.org/3/search/movie?api_key={api_key}&query=Jack+Reacher

    constructor(movie, key = "8ed7c07ca0c6eb792d78a3b9086aff66") {         //  Accept a string (for the movie search) and a key (to use the API).
        this.url_a = "https://api.themoviedb.org/3/search/movie?api_key="; //  This is the first part of the valid query URL.
        this.key = key;                                                    //  The key authorises use of the API.
        this.url_b = "&query=";                                            //  This is the second part of the valid query URL.
        this.movie = movie.split(' ').join('+');                           //  Replace any spaces in the book title with pluses so as to satify the URL query
requirements.
    }

    async getInfo() {
```

```
//  This method calls the getMovieInfo function to collect the JSON object data.
        let inf = await getMovieInfo(this.url_a, this.key, this.url_b, this.movie);
        this.detail = inf;
    }

    getMovieTitle() {
        //  Return the movie's title from the JSON object.
        var listOfMediaTitles = "";                                         //  Create the listOfMediaTitles variable and set it to null.
        for (var titleInstance = 0;; titleInstance++) {                     //  Setup a loop with no end condition (it will always end in a caught error).
            try {                                                           //  Attempt to collect the title value from the JSON object and append it to any previous
ones.
                listOfMediaTitles = listOfMediaTitles + this.detail['results'][titleInstance]['title'] + "<br>";
            }
            catch(err) {                                                    //  Catch the failed attempt to get the title value.
                if (titleInstance == 0) {                                   //  Has it failed on the first attempt?
                    return "No associated media found";                     //  If so, return a "No associated media found" response.
                }
                else {
                    return listOfMediaTitles;                               //  Otherwise, return the list of media titles.
                }
            }
        }
    }
}

//  This function goes off and collects the book detail information from the API.
async function getBookDetail(url_a, key, isbn, url_b) {                     //  Wait for this function to complete but don't tie up the CPU.
```

```
    let url = url_a + key + isbn + url_b;                                   //  Combine the two URL parts with the key and ISBN value to form a valid URL.
    try {
        const resp = await fetch(url);                                      //  Fetch the HTML data from the API.
        const jres = await resp.json();                                     //  Convert it into the JSON format.
        return jres;                                                        //  Return the JSON object data.
    }
    catch (err) {                                                           //  Catch any errors when trying to fetch the data from the API.
        window.alert("Unable to collect data from API - " + err);           //  In which case, provide an appropriate alert message.
    }
}

//  This function goes off and collects the movie information from the API.
async function getMovieInfo(url_a, key, url_b, movie) {                     //  Wait for this function to complete but don't tie up the CPU.
    let url = url_a + key + url_b + movie;                                  //  Combine the two URL parts with the key and movie title value to form a valid URL.
    try {
        const resp = await fetch(url);                                      //  Fetch the HTML data from the API.
        const jres = await resp.json();                                     //  Convert it into the JSON format.
        return jres;                                                        //  Return the JSON object data.
    }
    catch (err) {                                                           //  Catch any errors when trying to fetch the data from the API.
        window.alert("Unable to collect data from API - " + err);           //  In which case, provide an appropriate alert message.
    }
}
```