**Programming Language Practical Project**

**Phase 1**

Nouraldin Hassan

Contents

[Introduction 3](#_Toc177898140)

# Evidence of Learning

**Variables (and use of API Libraries, Arrays):**

4 const fs = require('fs');  
5 const csvParser = require('csv-parser');  
6 const readline = require('readline');  
7 const Record = require('./record');

9 const records = [];  
10 const filePath = ./keystone-throughput-and-capacity.csv';

“fs” is a required module that stands for “file system”; it is used for interacting with files on the computer for various purposes, such as C.R.U.D (Create, Read, Update, Delete) procedures and file manipulation.  
all requirements are constants so the reference value cannot be changed nor reassigned.

records is an array used to store record objects. filePath is a string variable holding the relative path to the csv dataset.

**Snippet: createReadStream (Loop Structure, Exception handling, File-IO):**

16 fs.createReadStream(filePath)  
17     .pipe(csvParser())  
18     .on('data', (row) => {  
19         try {  
20             console.log('Parsed Row:', row);  
21             /\*\*  
22              \* Initialize a new Record object with data from the CSV file row.  
23              \* @type {Record}  
24              \*/  
25             const record = new Record(row.Date, row.Month, row.Year, row.Company, row.Pipeline,  
row.KeyPoint, row.Latitude, row.Longitude, row.FlowDirection, row.TradeType, row.Product,  
row.Throughput, row.CommittedVolumes, row.UncommittedVolumes, row.NameplateCapacity,  
row.AvailableCapacity, row.VarianceReason);  
26             records.push(record);  
27         } catch (error) {  
28             console.error('Error parsing row:', error);  
29         }  
30     })  
31     .on('end', () => {  
32         console.log('CSV file successfully processed');  
33         promptUser();  
34     })  
35     .on('error', (error) => {  
36         console.error('Error reading file:', error);  
37     });

The given snippet performs reading upon the csv file path variable that was provided earlier in the program javascript file. Try-catch was used to handle exceptions in case any rows had issues or invalid data so they would either not show in the console or would as “undefined”, “null”, or simply blank. Record objects get pushed into the records array during the procedure. Once the data or file is detected to be complete or reached the end, a message appears and the user is prompted to print out a table-like formatted version of the records, rather than an array version of them.

**Methods:**

function displayRecords() {  
 console.log('Records:');  
 records.forEach(record => {  
 console.log(record);  
 });  
 }

displayRecords is a function that outputs the records from a foreach loop (iterating over) to the console, where the record info was gathered from the CSV file via File-IO (see previous section above).

**Program Demonstration via Screen Shots**

* **Include screenshots** of your running program showing:
  + Records from the dataset displayed on screen.
  + Your full name visible on the screen.

**Programming:**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated** **A screen shot of a computer program

Description automatically generated** A screen shot of a computer program

Description automatically generated

**Execution:**

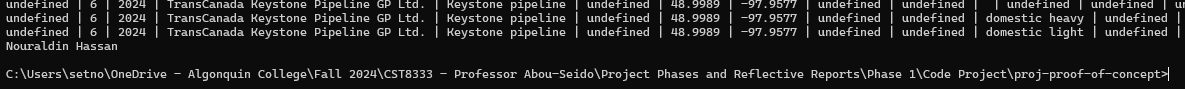
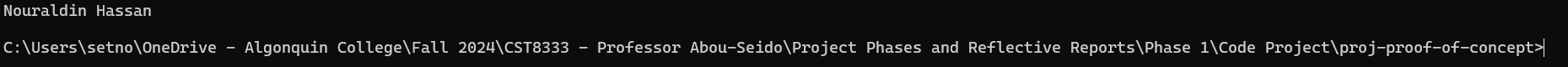
A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated  

A screenshot of a computer

Description automatically generated

 A black screen with white text

Description automatically generated

**Source Code Commenting Example**

// program.js  
/\*\*  
 \* Main program.  
 \* @author {Nouraldin Hassan}  
\*/  
const fs = require('fs');  
const csvParser = require('csv-parser');  
const readline = require('readline');  
const Record = require('./record');  
  
const records = [];  
const filePath = './keystone-throughput-and-capacity.csv';  
  
/\*\*  
 \* Read and display records from a CSV file.  
 \* Uses File-IO, exception handling, and a simple data structure.  
 \*/  
fs.createReadStream(filePath)  
    .pipe(csvParser())  
    .on('data', (row) => {  
        try {  
            console.log('Parsed Row:', row);  
            /\*\*  
             \* Initialize a new Record object with data from the CSV file row.  
             \* @type {Record}  
             \*/  
            const record = new Record(row.Date, row.Month, row.Year, row.Company,  
row.Pipeline, row.KeyPoint, row.Latitude, row.Longitude, row.FlowDirection, row.TradeType, row.Product, row.Throughput, row.CommittedVolumes, row.UncommittedVolumes, row.NameplateCapacity, row.AvailableCapacity, row.VarianceReason);  
            records.push(record);  
        } catch (error) {  
            console.error('Error parsing row:', error);  
        }

})  
    .on('end', () => {  
        console.log('CSV file successfully processed');  
        promptUser();  
    })  
    .on('error', (error) => {  
        console.error('Error reading file:', error);  
    });  
/\*\*  
 \* This function prompts the user for input.  
 \*/  
function promptUser() {  
    const rl = readline.createInterface({  
        input: process.stdin,  
        output: process.stdout  
    });  
  
    rl.question('Do you want to generate the table-like formatted records? (Y/N): ',  
(answer) => {  
        if (answer.toLowerCase() === 'y') {  
            displayRecords();  
            console.log('Nouraldin Hassan');  
        } else {  
            console.log('Table-like formatted records generation skipped.');  
            console.log('Nouraldin Hassan');  
        }  
        rl.close();  
    });  
}  
  
/\*\*  
 \* This function displays table-like formatted records.  
 \*/  
function displayRecords() {  
    console.log('Records:');  
    console.log('Date | Month | Year | Company | Pipeline | Key Point | Latitude | Longitude | Flow Direction | Trade Type | Product | Throughput | Committed Volumes | Uncommitted Volumes | Nameplate Capacity | Available Capacity | Variance Reason');  
    console.log('-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------');  
    records.forEach(record => {  
        console.log(`${record.date} | ${record.month} | ${record.year} | ${record.company} | ${record.pipeline} | ${record.keyPoint} | ${record.latitude} | ${record.longitude} | ${record.flowDirection} | ${record.tradeType} | ${record.product} | ${record.throughput} | ${record.committedVolumes} | ${record.uncommittedVolumes} | ${record.nameplateCapacity} | ${record.availableCapacity} | ${record.varianceReason}`);  
    });  
}