МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 1**

з дисципліни “Основи web-програмування”

тема “**Основи JavaScript**”

|  |  |  |
| --- | --- | --- |
| Виконав  студент II курсу  групи КП-91  Поштар Руслан Станіславович (*прізвище, ім’я, по батькові*)  варіант № 18 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

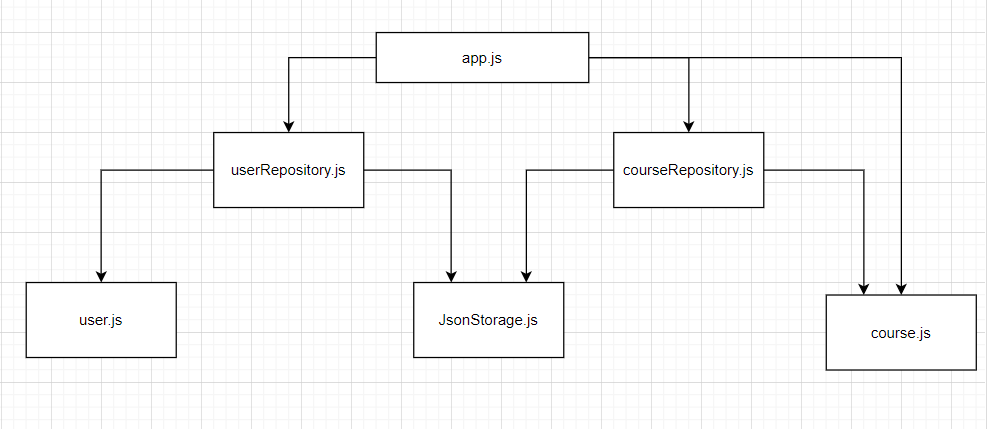
Познайомитись з мовою програмування JavaScript та середовищем Node.js.  
Навчитись створювати модулі, синхронно працювати із файловими потоками та обробляти JSON-текст.

**Постановка завдання**

Реалізувати на мові JavaScript (або TypeScript) консольну програму для керування двома типами сутностей із файлового сховища у форматі JSON: сутність "Користувач системи" та сутність за варіантом.

В якості сутностей за варіантом використати основну або додаткову сутність за темою вашої курсової роботи з "Основ програмування".

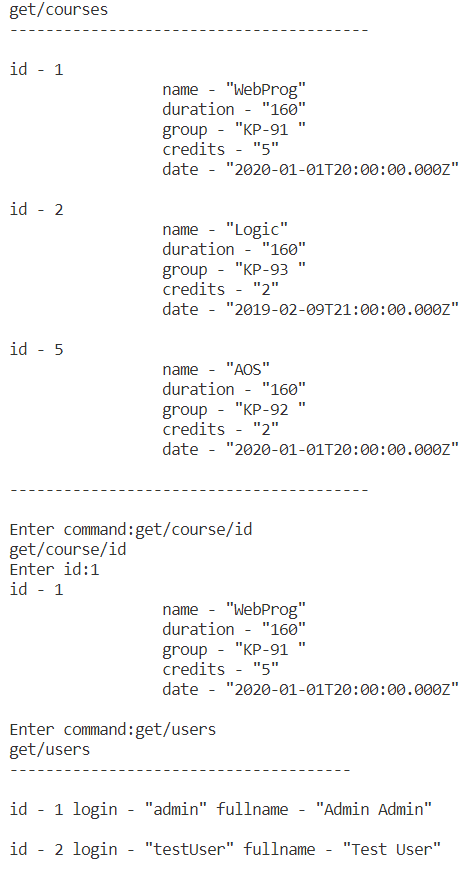
**Діаграма залежсностей модулів**

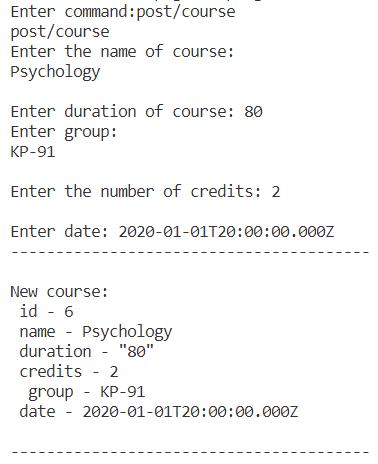
****

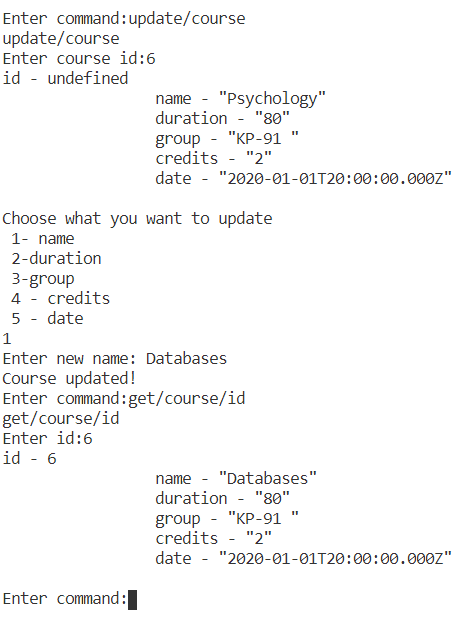
**Текст коду програми:**

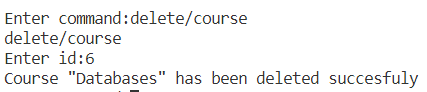
|  |
| --- |
| **app.js** |
| const readline = require('readline-sync')  const user = require("./models/user.js")  const UserRepository = require('./repositories/userRepository.js')  const users = new UserRepository("./data/users.json")    const courseRepository = require('./repositories/courseRepository.js')  const course = require('./models/course.js')  const courses = new courseRepository("./data/courses.json")  function getUsers()  {      const items = users.getAllUsers()      console.log("--------------------------------------\n")      if(items === null)      {          console.log("list empty now!")      }      else{          for(const item of items)          {              console.log(`id - ${item.id} login - "${item.login}" fullname - "${item.fullname}"\n`)          }      console.log("---------------------------------------\n")      }      }  function getUsersid(id)  {      id = readline.question("Enter id:")      if( id < 0 )      {          console.log("Invalid input!")      }      else if(id === 0)      {          console.log("Invalid input!")      }        else{      const Id = parseInt(id)      const item = users.getUserById(Id)      console.log("--------------------------------------\n")      console.log(item)      console.log("--------------------------------------\n")      }  }  function getCourses()  {      const items = courses.getAllCourses()      if(items === null)      {          console.log("list is empty!")      }      else{      console.log("----------------------------------------\n")      for(const item of items)      {      console.log(`id - ${item.id}                   name - "${item.name}"                   duration - "${item.duration}"                   group - "${item.group} "                   credits - "${item.credits}"                   date - "${item.date}"\n`)      }      console.log("----------------------------------------\n")      }    }  function getCourseById(id)  {      id = readline.question("Enter id:")      if(id < 0)      {          console.log("Invalid input!")      }      else if( id === 0)      {          console.log("Invalid input!")      }      else if(isNaN(id))      {          console.log("Invalid input!")      }      else{          const Id = parseInt(id)          const item = courses.getCourseId(Id)          console.log(`id - ${item.id}                   name - "${item.name}"                   duration - "${item.duration}"                   group - "${item.group} "                   credits - "${item.credits}"                   date - "${item.date}"\n`)      }    }  function courseDeletion(id)  {      id = readline.question("Enter id:")      if(id < 0)      {          console.log("Invalid input!")      }      else if( id === 0)      {          console.log("Invalid input!")      }      else if(isNaN(id))      {          console.log("Invalid input!")      }      else{          const Id = parseInt(id)          const item = courses.deleteCourse(Id)          if(item === null)          {              console.log("Course not found!")          }          else{              console.log(`Course "${item.name}" has been deleted succesfuly`)          }      }  }  function getDate() {      let date, parsedDate;      while (true) {          date = readline.question("Enter the new date: ")          parsedDate = Date.parse(date)          if (date.length === 0)          {              console.log('Error. The data field is empty.\n')          }          else{              break          }      }      date = new Date(parsedDate).toISOString();      return date;  }  function getUpdate(id){      id = readline.question("Enter course id:")  const Id = parseInt(id)  const item = courses.getCourseId(Id)      if (item === null)          console.log("Course not found!");      else {          console.log(`id - ${item.id}                   name - "${item.name}"                   duration - "${item.duration}"                   group - "${item.group} "                   credits - "${item.credits}"                   date - "${item.date}"\n`)          let numberOfData = 0;          while (true) {              numberOfData = readline.question("Choose what you want to update \n 1- name \n 2-duration \n 3-group \n 4 - credits \n 5 - date \n");              if (isNaN(numberOfData))                  console.log("Invalid input!")              else if (numberOfData === '1') {                  const name = readline.question("Enter new name: ")                  item.name = name              }              else if (numberOfData === '2') {                  const duration = readline.question("Enter new duration: \n")                  if(isNaN(duration))                  {                      console.log("Invalid input!")                  }                  else{                  item.duration = duration                  }              }              else if (numberOfData === '3')              {                  const group = readline.question("Enter new group: \n")                  item.group = group              }                else if (numberOfData === '4'){                  const credits = readline.question("Enter new nimber of credits: \n")                  if (isNaN(credits))                  {                      console.log("Invalid input!")                  }                  else{                      item.credits = credits;                  }              }              else if (numberOfData === '5'){                  item.date = getDate()              }              else              {                  console.log("Invalid input!")              }              courses.UpdateCourse(item)              console.log("Course updated!")              break;          }      }  }  function postCourse()  {      let newCourse = new course()      while (true) {          newCourse.name = readline.question("Enter the name of course: \n")          if (newCourse.name.length === 0)              {                  console.log("Empty string!")              }          else break        }      while (true) {          newCourse.duration = readline.question("\nEnter duration of course: ")          if (newCourse.duration.length === 0)              console.log("Empty string!")          else break      }        while (true) {          newCourse.group = readline.question("Enter group: \n")          if (newCourse.group.length === 0)              console.log("Empty string!")          else break      }      let tmp = ''      while (true) {          tmp = readline.question("\nEnter the number of credits: ")          if (isNaN(tmp))              console.log("Invalid input!")          else if (tmp < 0)              console.log("Invalid input!")          else if (tmp.length === 0)              console.log("Empty string!")          else break;      }      newCourse.credits = tmp      let parsedDate = '';      while (true) {          tmp = readline.question("\nEnter date: ")          parsedDate = Date.parse(tmp)          if (isNaN(parsedDate) || new Date(tmp).TryParseExact === 'exactundefined')              console.log(`Invalid input!\n`)          else              break;      }      tmp = new Date(parsedDate).toISOString()      newCourse.date = tmp;      const Id = courses.addCourse(newCourse)      const addedCourse = courses.getCourseId(Id)      console.log("----------------------------------------\n")      console.log(`New course:\n id - ${addedCourse.id}\n name - ${addedCourse.name}\n duration - "${addedCourse.duration}"\n credits - ${addedCourse.credits}\n  group - ${addedCourse.group}\n date - ${addedCourse.date}\n`)      console.log("----------------------------------------\n")  }    while (true) {      const input = readline.question("Enter command:");      console.log(input);          if (input === "create")      {          const name = readline.question("Enter name:")          console.log(name)      }      else if(input === "get/courses")      {          getCourses()      }      else if(input === "get/course/id")      {          getCourseById()      }      else if(input === "delete/course")      {          courseDeletion()      }      else if(input === "update/course")      {          getUpdate()      }      else if(input === "get/users")      {          getUsers()      }      else if(input === "get/user/id")      {          getUsersid()      }      else if(input === "post/course")      {          postCourse()      }      else if(input == "exit")      {          break;      }      else      {          console.log("Unknown command!")      }    } |
| **course.js** |
| class course  {      constructor(id,name,duration,group,credits,date)      {          this.id = id          this.name = name          this.duration = duration          this.group = group          this.credits = credits          this.date = date      }  }  module.exports = course; |
| **user.js** |
| class User  {       constructor(id,login,fullname,role,registeredAt,avaUrl,isEnabled)       {           this.id = id;           this.login = login;           this.fullname = fullname;           this.role = role;           this.registeredAt = registeredAt;           this.avaUrl = avaUrl;           this.isEnabled = isEnabled;       }    }  module.exports = User; |
| **courseRepository.js** |
| const course = require("../models/course.js")  const JsonStorage = require("../JsonStorage.js")  class courseRepository  {      constructor(filePath)      {          this.storage = new JsonStorage(filePath)      }      getAllCourses()      {          const items = this.storage.readItems().items;           let courses = [];           for( const item of items)           {               let Course = new course(item.id , item.name , item.duration , item.group , item.credits , item.date);               courses.push(Course);           }           return courses;      }      getCourseId(id)      {          const items = this.storage.readItems().items;          for(const item of items)          {          if(item.id === id)            {              return new course(item.id,item.name,item.duration,item.group,item.credits,item.date)            }          }      }      addCourse(entityModel)      {          const items = this.storage.readItems();          entityModel.id = this.storage.nextId;          items.items.push(entityModel);          this.storage.writeItems(items)          this.storage.incrementNextId()          return entityModel.id      }      UpdateCourse(entityModel)      {          const items = this.storage.readItems();          for (let i = 0; i < items.items.length; i++)          {              if (items.items[i].id === entityModel.id) {                  items.items[i] = entityModel;                  this.storage.writeItems(items);                  return true;              }          }          return null;      }      deleteCourse(entityId)      {          const items = this.storage.readItems();          for (let i = 0; i < items.items.length; i++) {              if (items.items[i].id === entityId) {                  const course = items.items[i];                  items.items.splice(i, 1);                  this.storage.writeItems(items);                  return course;              }          }          return null;      }  }  module.exports = courseRepository; |
| **userRepository.js** |
| const User = require('../models/user.js');  const JsonStorage = require('../JsonStorage.js');  class UserRepository  {          constructor(filePath)          {              this.storage = new JsonStorage(filePath);          }      getAllUsers()      {           const items = this.storage.readItems().items;           let users = [];           for( const item of items)           {               let user = new User(item.id , item.login , item.fullname);               users.push(user);           }           return users;      }      getUserById(id)      {          const items = this.storage.readItems().items;          for(const item of items)          {              if(item.id === id)              {                  return new User(item.id,item.login,item.fullname,item.role,item.registeredAt,item.avaUrl,item.isEnabled)              }          }      }    }  module.exports = UserRepository; |
| **jsonStorage.js** |
| // jsonStorage.js  const fs = require('fs');  class JsonStorage {      // filePath - path to JSON file      constructor(filePath) {          this.filePath = filePath;      }      get nextId() {          // TODO: get next entity id          const nextId = this.readItems().nextId;          return nextId;      }      incrementNextId() {          try{          const items = this.readItems();          items.nextId += 1;          this.writeItems(items);          }          catch{          throw new Error("Not implemented.");          }      }      readItems() {          let jsonText = fs.readFileSync(this.filePath);          if (jsonText.length === '0')              jsonText = { "nextId": 1, "items": [] };          else              jsonText = JSON.parse(jsonText);          return jsonText;      }      writeItems(items) {          const jsonText = JSON.stringify(items, null, 4);          fs.writeFileSync(this.filePath, jsonText);      }  };  module.exports = JsonStorage; |
| **Users.json** |
| {  "nextId": 3,  "items":[        {        "id": 1,        "login": "admin",        "fullname": "Admin Admin",        "role": 1,        "registeredAt":"2019-11-01T19:00:00.000Z",        "avaUrl":"https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSQaYVmbxsqHXNZc6k9pCplmH314MX-Hz\_uoQ&usqp=CAU",        "isEnabled":"true"      },      {        "id": 2,        "login": "testUser",        "fullname": "Test User",        "role": 2,        "registeredAt":"2020-07-02T21:00:00.000Z",        "avaUrl":"https://images.fineartamerica.com/images/artworkimages/mediumlarge/3/bmw-m5-e34-tuning-bmw-5-series-tunned-m5-sissy-angelastro.jpg",        "isEnabled":"true"      }   ]  } |
| **Courses.json** |
| {      "nextId": 6,      "items": [          {              "id": 1,              "name": "WebProg",              "duration": 160,              "group": "KP-91",              "credits": 5,              "date": "2020-01-01T20:00:00.000Z"          },          {              "id": 2,              "name": "Logic",              "duration": "160",              "group": "KP-93",              "credits": "2",              "date": "2019-02-09T21:00:00.000Z"          },          {              "id": 5,              "name": "AOS",              "duration": "160",              "group": "KP-92",              "credits": "2",              "date": "2020-01-01T20:00:00.000Z"          }      ]  } |

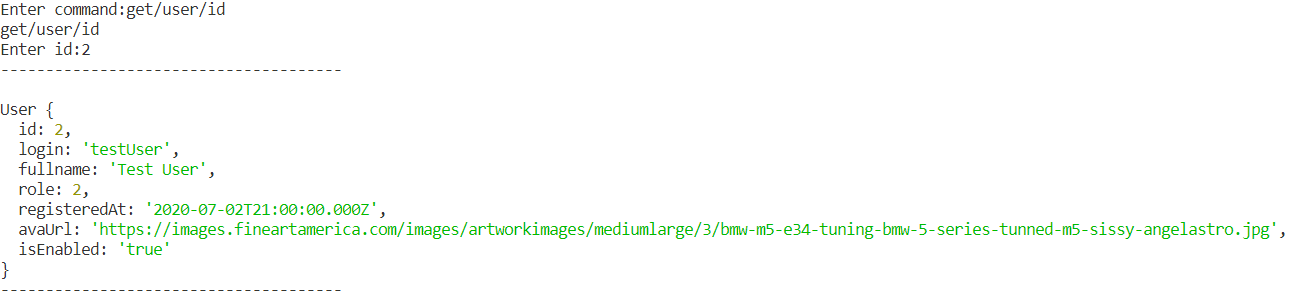
**Вивід роботи програми:**

****

****

****

****

****

**Висновки:**

При виконанні даної лабораторної роботи, було ознайомлено з мовою програмування JavaScript та середовищем Node.js. Реалізувано на мові JavaScript консольну програму для керування двома типами сутностей із файлового сховища у форматі JSON: сутність "Користувач системи" та сутність “Курси” за допомогою консольного інтерфейсу.  
Було створено модулі , для того , щоб синхронно працювати із файловими потоками та обробляти JSON-текст.