**Лабораторна робота №1**

**Побудова моделі структури інформаційної системи**

**Мета:** Навчитися аналізувати архітектуру існуючої інформаційної системи та представляти її у вигляді моделі структури.

**Завдання на лабораторну роботу**

1. Вибрати інформаційну систему для побудови її моделі.

2. Описати загальне призначення системи.

3. Побудувати модель структури системи на рівні програмно-технічної архітектури.

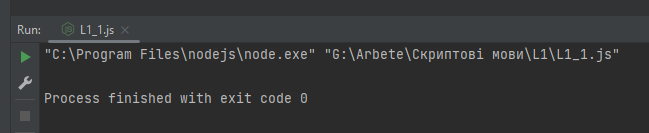
4. Побудувати модель структури цієї системи на будь-якому іншому рівні моделювання. Форма представлення результатів моделювання у разі вибирається довільно.

5. Описати змістовно, як у побудованих моделях враховані особливості побудови моделі структури.

**Результат виконання роботи**

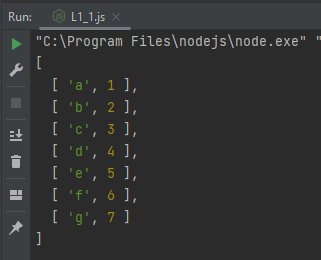
1. **Завдання 1.**

|  |
| --- |
| class Pair {  constructor(fst, snd){  if (this instanceof Pair) {  if (Array.isArray(fst) && fst.length === 2 && typeof snd == 'undefined') {  this[0] = fst[0];  this[1] = fst[1];  } else {  this[0] = fst;  this[1] = snd;  }  this.length = 2;  } else { return new Pair(fst, snd); }}  get fst(){ return this[0]; }  get snd(){ return this[1]; }  // статичний метод (функція) of, який повертає пару зі значень  static of(fst, snd){ return new Pair(fst,snd); }  // статичний метод (функція) fst, який повертає fst з класу  static fst(pair){ return pair.fst; }  // статичний метод (функція) snd, який повертає snd з класу  static snd(pair){ return pair.snd; }  }  a = new Pair(2,3);  if ((a.fst !==2) || (a.snd !==3)) {console.log("Error, Pair does not works as needed")}  b = new Pair(a,a);  if ((b.fst.snd !==3) || (b.snd.fst !==2)) {console.log("Error, Pair does not works as needed")} |



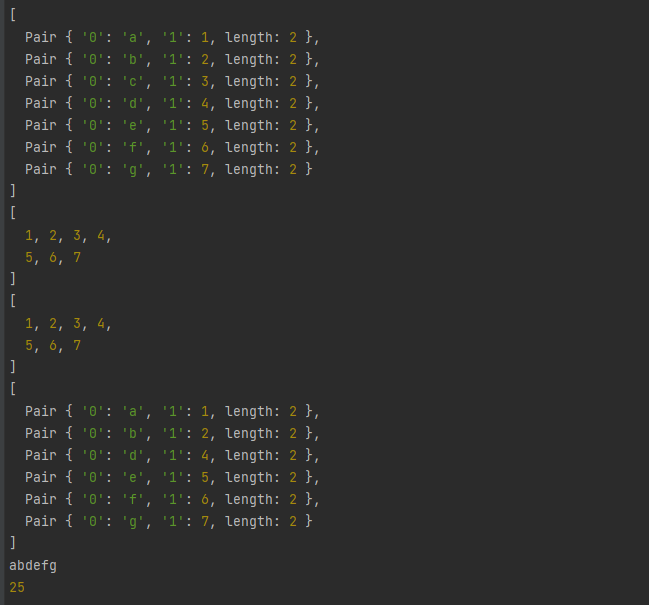
Далі підключаємо бібліотеку та спробуємо використати її:

|  |
| --- |
| let zipped = R.***zip***(["a", "b", "c", "d", "e", "f", "g"], [1, 2, 3, 4, 5, 6, 7]); ***console***.log(zipped); |



Далі пробуємо більш складні дії:

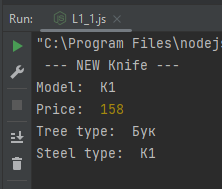
|  |
| --- |
| let zipped\_pairs = R.***map***(Pair.*of*, zipped); ***console***.log(zipped\_pairs); ***console***.log(R.***map***(Pair.*snd*, zipped\_pairs)) //map example ***console***.log(R.***map***(pair=>pair.snd, zipped\_pairs)) ***console***.log(R.***filter***(p=>Pair.*snd*(p) !==3, zipped\_pairs)) //filter example (now we have a let no\_third = R.***filter***(p => Pair.*snd*(p) !== 3, zipped\_pairs); ***console***.log(R.***reduce***((acc, el)=>acc+Pair.*fst*(el),"", no\_third)) //reduce example - we c ***console***.log(R.***reduce***((acc, el)=>acc+Pair.*snd*(el),0, no\_third)) |



1. **Завдання 2.**

Створiть клас, що вiдображає об’єкт (зберiгайте об’єкт у виглядi пар (в реальних проектах так робити не слiд, але в даному випадку це допоможе навчитись правильно комбiнувати виклики функцiй)) об’єкт має мiстити як мiнiмум 4 поля i хочаб одне з них числове (можливий будь який об’єкт, (кожному студенту придумати свiй варiант об’єкта)) далi створити масив з кiлькох таких об’єктiв.

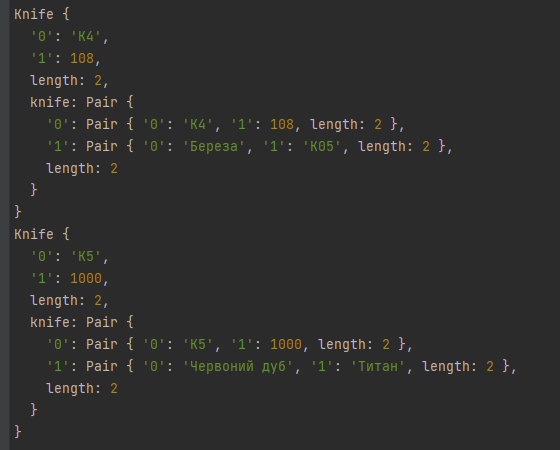
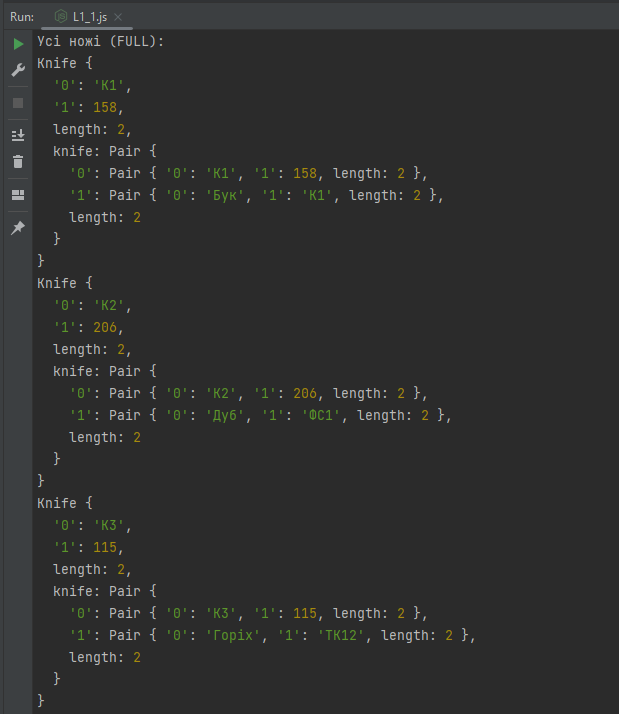
|  |
| --- |
| class Knife extends Pair {  constructor(model, price, tree\_type, steel\_type){  super(model, price);  this.knife = new Pair(new Pair(model, price), new Pair(tree\_type, steel\_type));  }   // get  get model(){ return this.knife.fst.fst; }  get price(){ return this.knife.fst.snd; }  get tree\_type(){ return this.knife.snd.fst; }  get steel\_type(){ return this.knife.snd.snd; }   // static  of(model, price, tree\_type, steel\_type){ return new Knife(model, price, tree\_type, steel\_type); }  static *model*(knife){ return knife.fst.fst; }  static *price*(knife){ return knife.fst.snd; }  static *tree\_type*(knife){ return knife.snd.fst; }  static *steel\_type*(knife){ return knife.snd.snd; } }  a = new Knife("K1", 158, "Бук", "К1") ***console***.log(" --- NEW Knife ---") ***console***.log("Model: ", a.model) ***console***.log("Price: ", a.price) ***console***.log("Tree type: ", a.tree\_type) ***console***.log("Steel type: ", a.steel\_type) |



**Не користуючись стандартними циклами:**

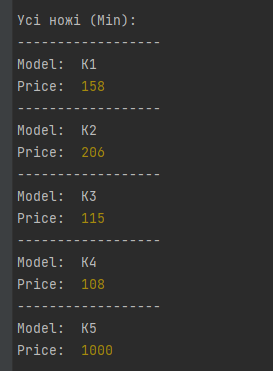
– вивести на екран об’єкти повнiстю, та по два головних для об’єкта поля (на вибiр студента)

|  |
| --- |
| a1 = new Knife("K2", 206, "Дуб", "ФС1") a2 = new Knife("K3", 115, "Горіх", "ТК12") a3 = new Knife("K4", 108, "Береза", "К05") a4 = new Knife("K5", 1000, "Червоний дуб", "Титан")  const array = [a, a1, a2, a3, a4];  ***console***.log("\n\nУсі ножі (FULL): "); array.forEach(knife => ***console***.log(knife)); |



Та по головним

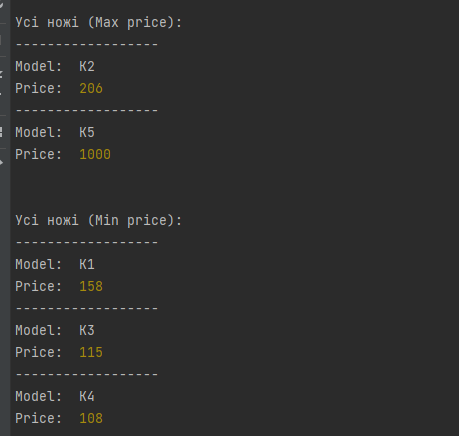
|  |
| --- |
| ***console***.log("\n\nУсі ножі (Min): "); array.forEach(knife => ***console***.log("------------------\nModel: ", knife.model, "\nPrice: ", knife.price)); |



– вiдфiльтрувати об’єкти зi створеного списка за певною ознакою (на вибiр студента)

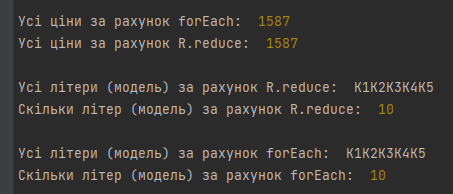
**Сортування по ціні**

|  |
| --- |
| max\_array = R.***filter***(k => k.price >= 200, array) min\_array = R.***filter***(k => k.price <= 200, array)  ***console***.log("\n\nУсі ножі (Max price): "); max\_array.forEach(knife => ***console***.log("------------------\nModel: ", knife.model, "\nPrice: ", knife.price)) ***console***.log("\n\nУсі ножі (Min price): "); min\_array.forEach(knife => ***console***.log("------------------\nModel: ", knife.model, "\nPrice: ", knife.price)) |



– порахувати середнє значення для поля з числовим значенням, та середню або максимальну кiлькiсть лiтер для рядкового поля.

|  |
| --- |
| let full\_money = 0 array.forEach(knife => full\_money += knife.price) ***console***.log("Усі ціни за рахунок forEach: ", full\_money) ***console***.log("Усі ціни за рахунок R.reduce: ", R.***reduce***((acc, el)=>acc+Pair.*snd*(el), 0, array))  ***console***.log("\nУсі літери (модель) за рахунок R.reduce: ", R.***reduce***((acc, el)=>acc+Pair.*fst*(el),"", array)) ***console***.log("Скільки літер (модель) за рахунок R.reduce: ", R.***reduce***((acc, el)=>acc+Pair.*fst*(el),"", array).length)  let full\_abc = "" array.forEach(knife => full\_abc += knife.model) ***console***.log("\nУсі літери (модель) за рахунок forEach: ", full\_abc) ***console***.log("Скільки літер (модель) за рахунок forEach: ", full\_abc.length) |



1. **Завдання 3.**

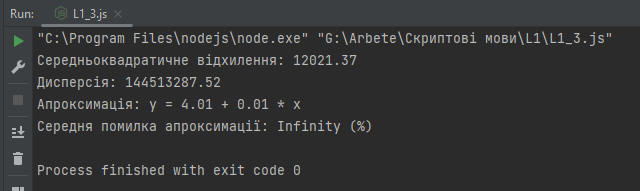
Iз сайту Kaggle (або iншого подiбного) обрати набiр даних (числових) Видiлити набори вхiдних та вихiдних даних В функцiональному стилi написати код що порахує середньоквадратичне вiдхилення, та дисперсiю для вихiдних даних Апроксимувати набір даних за допомогою методу найменших квадратiв порахувати середню похибку апроксимацiї

**Наші данні**: <https://www.kaggle.com/datasets/firozkabir1/crime-statistics-of-bangladesh-2010-2019>

**Назва**: Crime Statistics Of Bangladesh 2010-2019

**Вхідні**: Усього правопорушень.  
**Вихідні**: Правопорушення на дорозі.

|  |
| --- |
| function approximation(inpArray, xLabel, yLabel) {  let x = R.***map***(x => Number(x[xLabel]), inpArray);  let y = R.***map***(y => Number(y[yLabel]), inpArray);   const sumX = x.reduce((prev, curr) => prev + curr, 0);  const avgX = sumX / x.length;   const xDifferencesToAverage = x.map((value) => avgX - value);  const xDifferencesToAverageSquared = xDifferencesToAverage.map((value) => value \*\* 2);  const xX = xDifferencesToAverageSquared.reduce((prev, curr) => prev + curr,0);   const sumY = y.reduce((prev, curr) => prev + curr, 0);  const avgY = sumY / y.length;   const yDifferencesToAverage = y.map((value) => avgY - value);  const xAndYDifferencesMultiplied = xDifferencesToAverage.map((curr, index) => curr \* yDifferencesToAverage[index]);  const xY = xAndYDifferencesMultiplied.reduce((prev, curr) => prev + curr, 0);   const slope = xY / xX;  const intercept = avgY - slope \* avgX;  const resultFun = (x) => intercept + slope \* x;   ***console***.log(`Апроксимацiя: y = ${intercept.toFixed(2)} + ${slope.toFixed(2)} \* x`);  approximationError(resultFun, y, x);   return resultFun; }  function approximationError(func, y, x){  let A = (1 / y.length) \* R.***reduce***((acc, el) => acc + ***Math***.abs((el[1] - func(el[0])) / el[1]), 0, R.***zip***(x, y));  ***console***.log("Середня помилка апроксимацiї: " + ((A.toFixed(2))\*100) + " (%)"); } // Calculate the average of all the numbers const calculateMean = (values) => {return (values.reduce((sum, current) => sum + current)) / values.length;};  // Calculate variance const calculateVariance = (values) => {  const average = calculateMean(values);  const squareDiffs = values.map((value) => {  const diff = value - average;  return diff \* diff;  });  return calculateMean(squareDiffs); };  const R = ***require***('ramda'); const ***csvjson*** = ***require***('csvjson'); const readFile = ***require***('fs').readFile;  // Read CSV readFile('Crime.csv', 'utf-8', (err, fileContent) => {  if(err) {***console***.log(err); throw new Error(err);}  let jsonObj = ***csvjson***.toObject(fileContent);  let y = R.***map***(x => Number(x["Total"]), jsonObj);   //Середньоквадратичне відхилення та дисперсія  let variance = calculateVariance(y);  ***console***.log("Середньоквадратичне відхилення: " + ***Math***.sqrt(variance).toFixed(2));  ***console***.log("Дисперсія: " + variance.toFixed(2));   //Виклик функції для апроксимації  approximation(jsonObj, "Total", "Speedy"); }); |



**Проект на GitHub:**

<https://github.com/SING1370N/Script-Languages-JS>

**Висновки:**

У ході виконання лабораторної роботи ми ознайомилися з основами **JavaScrіpt**, функційного програмування, навчилися працювати з модулем **ramda**, створили свій клас, написали код для апроксимації набору даних.