Тема: Регулярні вирази. Перевірка даних.

**Мета:** Ознайомлення з принципами використання регулярних виразів для перевірки рядка на відповідність шаблону.

**1 ВИМОГИ  
1.1 Розробник**

Інформація про розробника:

- Когутенко Олександр Олексійович;

- КІТ-119Д;

- 11 варіант.

**1.2 Загальне завдання**

Продемонструвати ефективне (оптимальне) використання регулярних виразів для перевірки коректності (валідації) даних, що вводяться, перед записом в [domain-об'єкти відповідно до призначення кожного поля](https://oop-khpi.gitlab.io/%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F/task07/" \l "_4) для заповнення [розробленого контейнера](https://oop-khpi.gitlab.io/%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F/task09/):

* при зчитуванні даних з текстового файла в автоматичному режимі;
* при введенні даних користувачем в діалоговому режимі.

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП

Використовуеться наслідування, інтерфейс, поліморфізм.

2.2 Ієрархія та структура класів

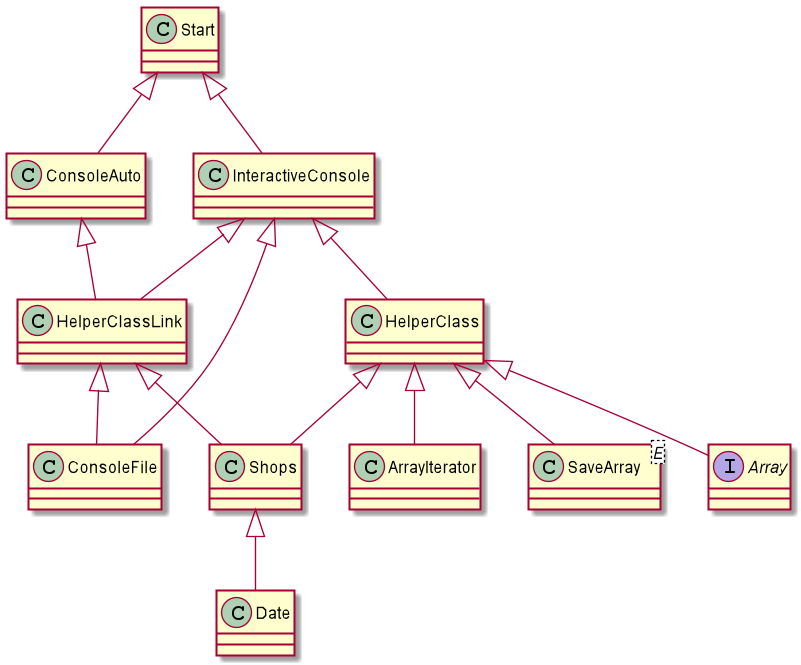


Рисунок 11.1 - иерархія класів

Використовую 11 классів: Array, ArrayIterator, HelperClass, InteractiveConsole, Start, SaveArray, Date, Shops, ConsoleFile, HelperClassLinkб, ConsoleAuto.

* Array використовую як інтерфейс для класу контейнеру.
* ArrayIterator використовую як особисту реалізацію ітератора.
* HelperClass допоміжний класс для розрахунків.
* InteractiveConsole клас для налагодженого спілкування програми з користувачем та методами нестандартних протоколів серіалізації.
* Start клас який має точку входу у програму.
* SaveArray клас контейнер який має все необхідні методи маніпулятори.
* Date клас використовуется для збереження дати.
* Shops клас приклодної галузі.
* ConoleFile клас за допомогою якого користувач може наблюдати за відображенням вмісту каталогів.
* HelperClassLink використовуеться за для реалізації зв’язного списку з методами стандартних методів серіалізації.
* ConsoleAuto класс для автоматичної роботи із списком.

2.3 Важливі фрагменти програми

Є багато вставок за “регулярними” перевірками, але вставлю один прикла класу з виразами:

package ua.khpi.oop.kogutenko11;  
  
import java.io.Serializable;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Scanner;  
import java.util.Set;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
*/\*\*  
 \* The type Shops.  
 \*/*public class Shops implements Serializable {  
 ...

*/\*\*  
 \* Sets description.  
 \*  
 \** ***@param*** *description the description  
 \*/* public void setDescription(String description) {  
 Pattern p;  
 Matcher m;  
 String key = null, val = null, strOn = null;  
 String[] str = description.split(",");  
 String[] strAdd;  
 int count = 0;  
 for(int i = 0; i < str.length; i++) {  
 strAdd = str[i].split(" - ");  
 p = Pattern.*compile*("[\\w]{15}]");  
 key = strAdd[0];  
 m = p.matcher(key);  
 if(m.matches()){  
 key = "key";  
 }  
 val = strAdd[1];  
 m = p.matcher(val);  
 if(m.matches()){  
 val = "val";  
 }  
 this.description.put(key,val);  
 }  
 }  
  
 ...

*/\*\*  
 \* Add.  
 \*/* public void add(){  
 //regex!!!!!!  
 Pattern p;  
 Matcher m;  
 boolean regexLoop = true;  
 String id = null, name = null, unit = null, count = null, price = null, day = null, mon = null, year = null, key = null, val = null, date = null;  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("Enter info:\n");  
 while(regexLoop) {  
 System.*out*.print("Enter id: ");  
 p = Pattern.*compile*("[0**-**9]+");  
 id = sc.nextLine();  
 m = p.matcher(id);  
 if(m.matches()){  
 setId(Integer.*parseInt*(id));  
 regexLoop = false;  
 } else {  
 System.*out*.println("Enter info correctly!!! (only numbers)");  
 }  
 }  
  
 regexLoop = true;  
 while(regexLoop){  
 System.*out*.print("Enter name: ");  
 p = Pattern.*compile*("^[\\w]{3,15}$");  
 name = sc.nextLine();  
 m = p.matcher(name);  
 if(m.matches()){  
 setName(name);  
 regexLoop = false;  
 } else {  
 System.*out*.println("Enter info correctly!!! (letters and '-' or '\_')");  
 }  
 }  
  
 regexLoop = true;  
 while(regexLoop){  
 System.*out*.print("Enter unit: ");  
 p = Pattern.*compile*("kg|l|kg/l");  
 unit = sc.nextLine();  
 m = p.matcher(unit);  
 if(m.matches()){  
 setUnit(unit);  
 regexLoop = false;  
 } else {  
 System.*out*.println("Enter info correctly!!! ('kg' or 'l' or 'kg/l')");  
 }  
 }  
 regexLoop = true;  
 while(regexLoop){  
 System.*out*.print("Enter count: ");  
 p = Pattern.*compile*("[0**-**9]+");  
 count = sc.nextLine();  
 m = p.matcher(count);  
 if(m.matches()){  
 setCount(Integer.*parseInt*(count));  
 regexLoop = false;  
 } else {  
 System.*out*.println("Enter info correctly!!! (only numbers)");  
 }  
 }  
 regexLoop = true;  
 while(regexLoop){  
 System.*out*.print("Enter price: ");  
 p = Pattern.*compile*("[0**-**9]+");  
 price = sc.nextLine();  
 m = p.matcher(price);  
 if(m.matches()){  
 setPrice(Integer.*parseInt*(price));  
 regexLoop = false;  
 } else {  
 System.*out*.println("Enter info correctly!!! (only numbers)");  
 }  
 }  
  
 regexLoop = true;  
 while(regexLoop){  
 System.*out*.print("Enter date('dd-mm-yyyy' or 'dd/mm/yyyy' or 'dd.m.yyyy'): ");  
 p = Pattern.*compile*("^(?:(?:31(\\/|-|\\.)(?:0?[13578]|1[02]))\\1|(?:(?:29|30)(\\/|-|\\.)(?:0?[1,3**-**9]|1[0**-**2])\\2))(?:(?:1[6**-**9]|[2**-**9]\\d)?\\d{2})$|^(?:29(\\/|-|\\.)0?2\\3(?:(?:(?:1[6**-**9]|[2**-**9]\\d)?(?:0[48]|[2468][048]|[13579][26])|(?:(?:16|[2468][048]|[3579][26])00))))$|^(?:0?[1**-**9]|1\\d|2[0**-**8])(\\/|-|\\.)(?:(?:0?[1**-**9])|(?:1[0**-**2]))\\4(?:(?:1[6**-**9]|[2**-**9]\\d)?\\d{2})$");  
 date = sc.nextLine();  
 m = p.matcher(date);  
 if(m.matches()){  
 day = date.substring(0,2);  
 mon = date.substring(3,5);  
 year = date.substring(6,10);  
 regexLoop = false;  
 } else {  
 System.*out*.println("Enter info correctly!!!");  
 }  
 }  
 this.date.setDate(Integer.*parseInt*(day), Integer.*parseInt*(mon), Integer.*parseInt*(year));  
  
  
 System.*out*.println("\nEnter some description: ");  
 boolean check = true;  
 while (check) {  
 regexLoop = true;  
 while(regexLoop){  
 System.*out*.print("Enter key: ");  
 p = Pattern.*compile*("[\\w]{3,15}");  
 key = sc.nextLine();  
 m = p.matcher(key);  
 if(m.matches()){  
 while(regexLoop){  
 System.*out*.print("Enter val: ");  
 p = Pattern.*compile*("[\\w]{3,15}");  
 val = sc.nextLine();  
 m = p.matcher(val);  
 if(m.matches()){  
 this.description.put(key,val);  
 regexLoop = false;  
 } else{  
 System.*out*.println("Enter info correctly!!!\nOnly numbers, letters and '\_'");  
 }  
 }  
 }else{  
 System.*out*.println("Enter info correctly!!!\nOnly numbers, letters and '\_' or '-'");  
 }  
 }  
 System.*out*.print("Do you want to add mor one description? (0 - no, 1 - yes)\n>>> ");  
 int answer = sc.nextInt();  
 if (answer == 0)  
 {  
 check = false;  
 }  
 }  
 }  
  
  
}

**3 ВАРІАНТИ ВИКОРИСТАННЯ**

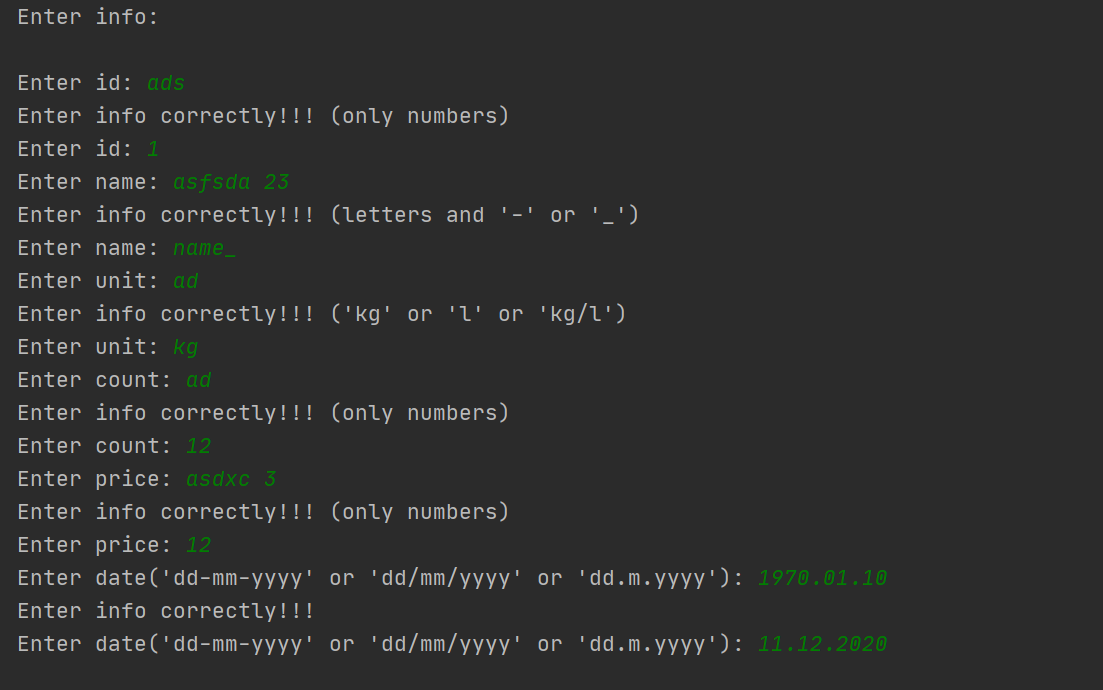


Рисунок 11.2 - ведення даних де перевырка регулярними виразами

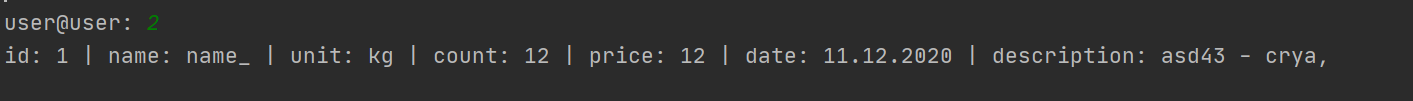


Рисунок 11.3 - ведення даних

**ВИСНОВКИ**

Ознайомився з принципами використання регулярних виразів для перевірки рядка на відповідність шаблону.