9. Параметризація в Java

Мета: Вивчення принципів параметризації в *Java*. Розробка параметризованих класів та методів.

1 ВИМОГИ

1.1 Розробник

Інформація про розробника:

- Кулик Даніїл Ігорович
- НТУ “ХПІ” 1.КІТ102.8а
- Варіант 11

1.2 Загальне завдання

1. Створити власний клас-контейнер, що параметризується (Generic Type), на основі зв'язних списків для реалізації колекції domain-об’єктів лабораторної роботи №7.
2. Для розроблених класів-контейнерів забезпечити можливість використання їх об'єктів у циклі foreach в якості джерела даних.
3. Забезпечити можливість збереження та відновлення колекції об'єктів: 1) за допомогою стандартної серіалізації; 2) не використовуючи протокол серіалізації.
4. Продемонструвати розроблену функціональність: створення контейнера, додавання елементів, видалення елементів, очищення контейнера, перетворення у масив, перетворення у рядок, перевірку на наявність елементів.
5. Забороняється використання контейнерів (колекцій) з Java Collections Framework.

1.3 Задача

11. Прикладна галузь: Магазин. Запис в каталозі товарів: найменування; одиниця виміру; кількість; ціна одиниці; дата надходження; опис (необмежений набір характеристик).

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП

Був розроблений domain object, ітеруючий клас контейнер. Також були розроблені дві серіалізації: із використовуванням стандартного протоколу та без нього.

2.2 Ієрархія та структура даних

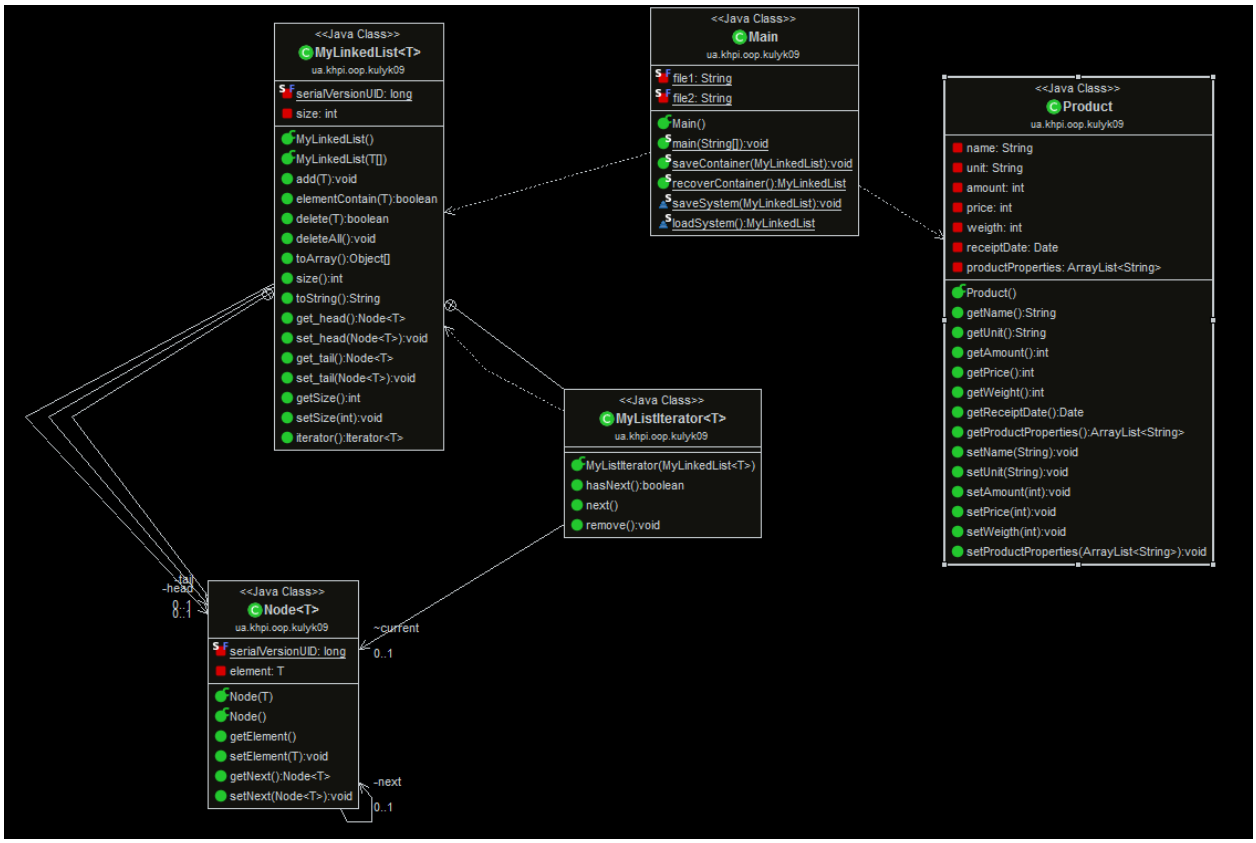

Рисунок 1 – Діаграма класів

2.3 Важливі фрагменти програми

```java
public boolean elementContain(T item) {
    Node<T> current = head;
    while (current != null) {
        if (current.element.equals(item)) {
            return true;
        }
        current = current.next;
    }
    return false;
}

public boolean delete(T item) {
    Node<T> previous = null;
    Node<T> current = head;
    while (current != null) {
        if (current.element.equals(item)) {
            if (previous != null) {
                previous.next = current.next;

                if (current.next == null) {
                    tail = previous;
                }
            }
            else {
                head = head.next;
                if (head == null) {
                    tail = null;
                }
            }
            size--;
            return true;
        }
        previous = current;
        current = current.next;
    }
    return false;
}
```

Рисунок 2 – Методи перевірки на наявність та видалення елементу

```java
@Override
public Iterator<T> iterator() {
        return new MyListIterator<T>(this);
}

@SuppressWarnings("hiding")
class MyListIterator<T> implements Iterator<T> {
    Node<T> current;
    @SuppressWarnings("unchecked")
    public MyListIterator(MyLinkedList<T> list) {
        current = (Node<T>) head;
    }
    public boolean hasNext() {
        return current != null;
    }
    public T next() {
        T data = current.getElement();
        current = current.getNext();
        return data;
    }
    public void remove() {
        throw new UnsupportedOperationException();
    }
}
```

Рисунок 3 – Ітератор у класі-контейнері

3 ВАРІАНТИ ВИКОРИСТАННЯ

Програма дозволяє створювати об’єкти – товари – , що заносяться у запис каталогу, тобто створюється масив об’єктів. Користувач може додавати об’єкти до масиву, видаляти елементи вибірково, а також очистити увесь масив одним викликом відповідної кнопки меню. Також присутня можливість серіалізувати/десеріалізувати об’єкти з файлу.

```
System.out.println(products.toString());
saveSystem(products);
products.deleteAll();
System.out.println(products.toString());
products = loadSystem();
System.out.println(products.toString());
products.add("Sausages");
System.out.println(products.toString());
System.out.println(products.delete("Sausages"));
System.out.println(products.toString());

System.out.println("----------------------------

saveContainer(products);
products.deleteAll();
System.out.println(products.toString());
products = recoverContainer();
System.out.println(products.toString());
products.add("Strawberry");
System.out.println(products.toString());
System.out.println(products.delete("Tomatoes"));
System.out.println(products.toString());

System.out.println("----------------------------

    for(Object s : products) {
            System.out.println(s.toString());
    }
```

Рисунок 4 – Виклик методів для обробки контейнеру

```
[ "Tomatoes" "Apples" "Orange" "T_shirts" ]
[ ]
[ "Tomatoes" "Apples" "Orange" "T_shirts" ]
[ "Tomatoes" "Apples" "Orange" "T_shirts" "Sausages" ]
true
[ "Tomatoes" "Apples" "Orange" "T_shirts" ]
--------------------------------------------------
[ ]
[ "Tomatoes" "Apples" "Orange" "T_shirts" ]
[ "Tomatoes" "Apples" "Orange" "T_shirts" "Strawberry" ]
true
[ "Apples" "Orange" "T_shirts" "Strawberry" ]
--------------------------------------------------
Apples
Orange
T_shirts
Strawberry
```

Рисунок 5 – Результати виконання

Рисунок 6 – Зміст файлу *Container.xml*

ВИСНОВКИ

В даній лабораторній роботі розробив та реалізував принципи параметризації в Java, параметризовані класи-контейнери на основі зв’язних списків та їх методи.