Міністерство освіти та науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



**ЗВІТ**

**з лабораторної роботи № 7**

**з дисципліни:** «Кросплатформенні засоби програмування»

**на тему:** «ПАРАМЕТРИЗОВАНЕ ПРОГРАМУВАННЯ»

Виконав: ст. гр. КІ-35

Гудима О.П.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю.С

Львів-2022

**Мета роботи:** оволодіти навиками параметризованого програмування мовою Java.

**ЗАВДАННЯ**

1. Створити параметризований клас, що реалізує предметну область задану варіантом. Клас має містити мінімум 4 методи опрацювання даних включаючи розміщення та виймання елементів. Парні варіанти реалізують пошук мінімального елементу, непарні – максимального. Написати на мові Java та налагодити програму-драйвер для розробленого класу, яка мстить мінімум 2 різні класи екземпляри яких розмішуються у 9 екземплярі розробленого класу-контейнеру. Програма має розміщуватися в пакеті Група.Прізвище.Lab6 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленого пакету.

3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.

4. Дати відповідь на контрольні запитання.

**Варіант 2:**

****

**Текст програми:**

|  |
| --- |
| ***Лістинг Main***  *public class* Main {  *public static void* main(String[] args) {  LinkedList<? *super* Numbers> list = *new* LinkedList<>();   list.insert(*new* HexNumbers("14D", 332));  list.insert(*new* BinaryNumbers("1", 56));  list.insert(*new* HexNumbers("593A", 22842));  list.insert(*new* HexNumbers("19BD", 6589));  list.insert(*new* BinaryNumbers("11 1101 1000", 984));  list.insert(*new* HexNumbers("E2F4", 58100));  list.insert(*new* BinaryNumbers("11 0101 1000", 856));  list.insert(*new* BinaryNumbers("10 0101 0111", 599));  list.insert(*new* HexNumbers("15C1", 5569));  list.insert(*new* BinaryNumbers("10 1001 1101", 669));   list.printList();  System.out.println("Min value is "+ list.FindMin());  System.out.println("Is even " + list.isEven());  } }  ***Лістинг HexNumbers***  *public class* HexNumbers *implements* Numbers {  *private* String number;  *private int* value;  *public* String toString(){   *return* Integer.toString(value);  }   *public* HexNumbers(String number, *int* value) {  *this*.number = number;  *this*.value = value;  }   *public* String getNumber() {  *return* number;  }   *public void* setNumber(String number) {  *this*.number = number;  }   *public int* getValue() {  *return* value;  }   *public void* setValue(*int* value) {  *this*.value = value;  }   *@Override  public int* getDecValue() {  *return* value;  }   *@Override  public void* PrintInfo() {  System.out.println("Number in hex: " + number + ",\tvalue in dec: " + value);  } }  ***Лістинг BinaryNumbers***  *public class* BinaryNumbers *implements* Numbers {  *private* String number;  *private int* value;  *public* String toString(){   *return* Integer.toString(value);  }   *public* BinaryNumbers(String number, *int* value) {  *this*.number = number;  *this*.value = value;  }   *public* String getNumber() {  *return* number;  }   *public void* setNumber(String number) {  *this*.number = number;  }   *public int* getValue() {  *return* value;  }   *public void* setValue(*int* value) {  *this*.value = value;  }   *@Override  public int* getDecValue() {  *return* value;  }   *@Override  public void* PrintInfo() {  System.out.println("Number in binary: " + number + ",\tvalue in dec: " + value);  } }  ***Лістинг Numbers***  *public interface* Numbers {  *int* getDecValue();  String getNumber();  *void* PrintInfo(); }  ***Лістинг LinkedList***  *public class* LinkedList<T *extends* Numbers> {  Node<T> head;  T min;  *static class* Node<T *extends* Numbers> {  T data;  Node<T> next;  Node(T d)  {  data = d;  next = *null*;  }  }   *public void* insert(T data)  {  Node<T> new\_node = *new* Node<T>(data);  new\_node.next = *null*;   *if* (*this*.head == *null*) {  *this*.head = new\_node;  }  *else* {  Node<T> last = *this*.head;  *while* (last.next != *null*) {  last = last.next;  }   last.next = new\_node;  }  }   *public void* printList()  {  Node<T> currNode = *this*.head;   System.out.print("LinkedList: \n");   *while* (currNode != *null*) {  System.out.print(" ");  currNode.data.PrintInfo();  currNode = currNode.next;  }  }   *public* T FindMin() *//метод для пошуку максимального об'єкта* {   Node<T> minNode = *this*.head;  T data = minNode.data;  *while* (minNode.next != *null*) {  *if*( data.getNumber().length() > minNode.next.data.getNumber().length())  data = minNode.next.data;   minNode = minNode.next;  }  min = data;  *return* data;  }  *public boolean* isEven(){  *return* min.getDecValue()%2 == 0;  } } |

**Результат роботи :**

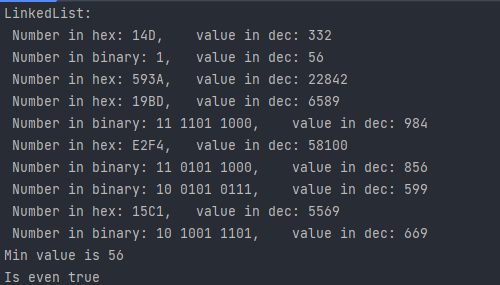
****

Рис.1 Результат роботи програми

**Відповіді на контрольні запитання:**

1. Дайте визначення терміну «параметризоване програмування».

Параметризоване програмування є аналогом шаблонів у С++. Воно полягає у написанні коду, що можна багаторазово застосовувати з об’єктами різних класів. Користувачів параметризованого програмування можна поділити на 3 рівні кваліфікації:

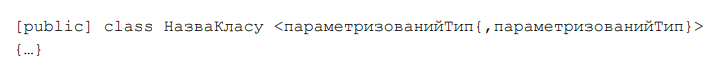
1. ті, що користуються готовими класами;

2. ті, що користуються готовими класами і вміють виправляти помилки, що виникають при їх використанні;

3. ті, що пишуть власні параметризовані класи. Для успішного застосування параметризованого програмування слід навчитися розуміти помилки, що генерує середовище при компіляції програми, що можуть стосуватися, наприклад, неоднозначності визначення спільного суперкласу для всіх переданих об’єктів. З іншої сторони необхідно передбачити захист від присвоєння об’єктів параметризованого класу, що містять об’єкти підкласу об’єктам параметризованого класу, що містять об’єкти суперкласу і дозволити зворотні присвоєння. Для вирішення цієї проблеми у мові Java введено так звані підстановочні типи. Це далеко не всі «підводні камені», що виникають при застосуванні параметризованого програмування.

1. Розкрийте синтаксис визначення простого параметризованого класу.

Параметризований клас – це клас з однією або більше змінними типу. Синтаксис оголошення параметризованого класу:



**Висновок:**

Під час виконання роботи я оволодів навиками параметризованого програмування мовою Java.