Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Лабораторна робота №2

з курсу:

“ОБ’ЄКТНО ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ”

Виконав:
ст. гр. КН-110
Шаварський
Максим
Прийняв:
Гасько Р.Т.

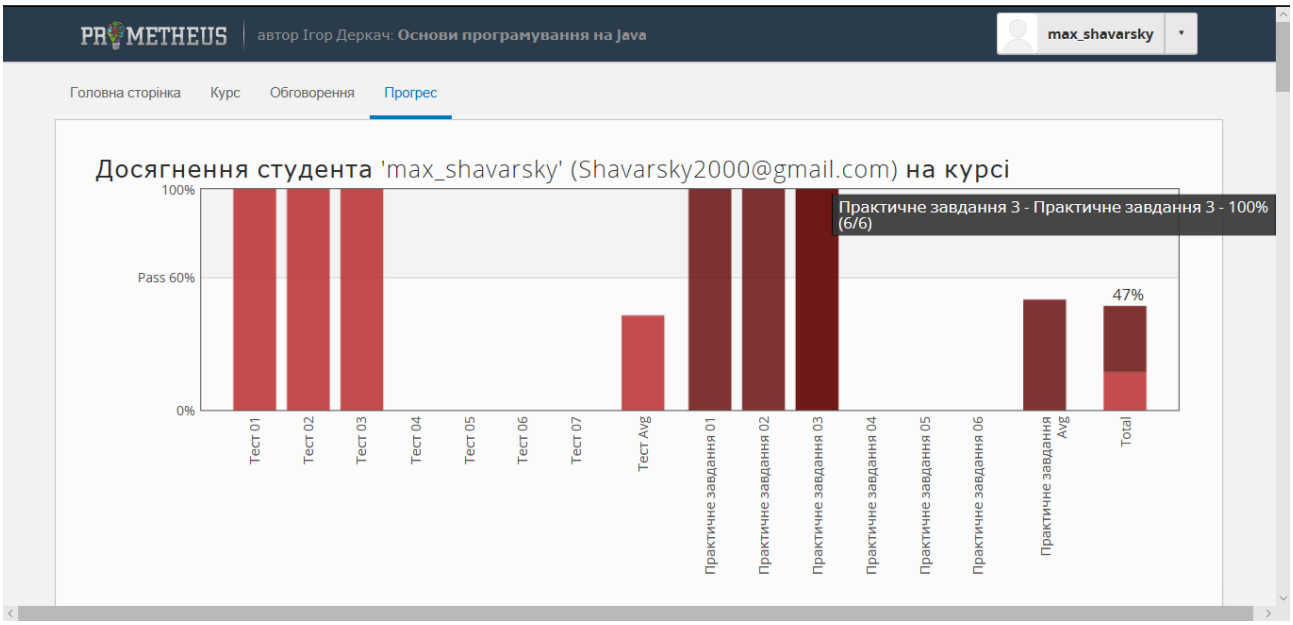
Львів – 2018 р.

Лабораторна робота № 2

Завдання: Пройти 3 тиждень курсу на платформі Prometheus.

Виконання роботи

GitHub link - https://github.com/MaxShavarsky/OOP_JAVA-New-



Лабораторна №2,Тиждень 3,Завдання 1.

```
package com.tasks3.linkedlist;
public class LinkedList {
	private Node golova;
	private Node hvist;
	private int count = 0;
	public LinkedList() {

	}
	public void add(Integer data) {
		Node Spysok = new Node();
		Spysok.setData(data);

		if (golova == null) {
			golova = Spysok;
			hvist = Spysok;
		}
		else {
			hvist.setNext(Spysok);
			hvist = Spysok;
		}
		count++;

	}

	public Integer get(int index) {
```

```
        Node Spysok = golova;
        if (index > count) {
                return null;
        }
        else {
                for (int i = 0; i <= index; i++) {
                        if (i == index) {
                                return Spysok.getData();

                        }
                        Spysok = Spysok.getNext();
                }

        }
        return null;

}
public int size() {
        return count;

}

public boolean delete(int index) {
        Node mysor = null;
        Node mysor2 = golova;
        if (index>count) {

                return false;
        }
        else if (index == 0) {
                golova = golova.getNext();
                if (golova == null) {

                        hvist = null;
                }
                count--;
                return true;

        }
        else if (index == count - 1) {
                for (int i = 0; i <= index; i++) {
                        if (i == index) {
                                hvist = mysor;
                                hvist.setNext(null);
                                count--;
                                return true;
                        }
                        mysor = mysor2;
                        mysor2 = mysor2.getNext();
                }
        }
        else {
                for (int i = 0; i <= index; i++) {
                               if (i == index) {
                                mysor.setNext(mysor2.getNext());
                                count--;
                                return true;

                        }
                        mysor = mysor2;
                        mysor2 = mysor2.getNext();

                }

        }
```

```
		return false;

	}
```

```
}
```

Лабораторна №2,Тиждень 3,Завдання 2.

```
package com.tasks3.carddeck;

import java.util.Comparator;
import java.util.LinkedList;
import java.util.TreeSet;
public class Deck {
	private final int SIZE = 36;
	private LinkedList<Card> cardList;
	public Deck() {
		cardList = new LinkedList<>();

		for (int suit = 0; suit < Suit.values.length; suit++) {
			for (int rank = 0; rank < Rank.values.length; rank++) {
				cardList.add(new Card(Rank.values[rank], Suit.values[suit]));
			}
		}

	}
	public void shuffle() {
		Double RndCrdInd;
		Double numberOfShuffles = Math.random() * 100 + 30;
		for (int i = 0; i < numberOfShuffles.intValue(); i++) {
			RndCrdInd = Math.random() * SIZE;
			cardList.addLast(cardList.remove(RndCrdInd.intValue()));
		}
	}
	public void order() {
		class CardComparator implements Comparator<Card> {
			@Override
				public int compare(Card card1, Card card2) {
				int Rank_Card_1 = 0;
				int Rank_Card_2 = 0;

				for (int i = 0; i < Rank.values.length; i++) {
					if
(card1.getRank().getName().equals(Rank.values[i].getName()))
						Rank_Card_1 = i;
					if
(card2.getRank().getName().equals(Rank.values[i].getName()))
						Rank_Card_2 = i;
				}
				if (Rank_Card_1 == Rank_Card_2)
					return 0;
				else {
					if (Rank_Card_1 > Rank_Card_2)
						return 1;
					else
						return -1;
				}
			}

		}
		TreeSet<Card> spadesSet = new TreeSet<>(new CardComparator());
		TreeSet<Card> clubsSet = new TreeSet<>(new CardComparator());
		TreeSet<Card> diamondsSet = new TreeSet<>(new CardComparator());
		TreeSet<Card> heartsSet = new TreeSet<>(new CardComparator());
```

```
		while (this.hasNext()) {
			Card tempcard;

			tempcard = this.drawOne();

			if (tempcard.getSuit() == Suit.HEARTS)
				heartsSet.add(tempcard);
			else {
				if (tempcard.getSuit() == Suit.DIAMONDS)
					diamondsSet.add(tempcard);
				else {
					if (tempcard.getSuit() == Suit.CLUBS)
						clubsSet.add(tempcard);
					else {
						if (tempcard.getSuit() == Suit.SPADES)
							spadesSet.add(tempcard);
					}
				}
			}

		}
		cardList.addAll(heartsSet);

		cardList.addAll(diamondsSet);
		cardList.addAll(clubsSet);
		cardList.addAll(spadesSet);

		heartsSet = diamondsSet = clubsSet = spadesSet = null;
	}
	public boolean hasNext() {
		return !cardList.isEmpty();
	}
	public Card drawOne() {
		if (hasNext()) {
			return cardList.removeLast();
		}
		else
			return null;

	}
}
```

Лабораторна №2,Тиждень 3,Завдання 3

```
public class Fibonacci

{
	public static long getNumber(int position) {

		if (position <  1) return -1;
		if (position == 1) return 1;
		if (position == 2) return 1;
		return getNumber(position - 1) + getNumber(position - 2);

	}

}.
```