Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Лабораторна робота №3

на тему:

«Вступ до Java»

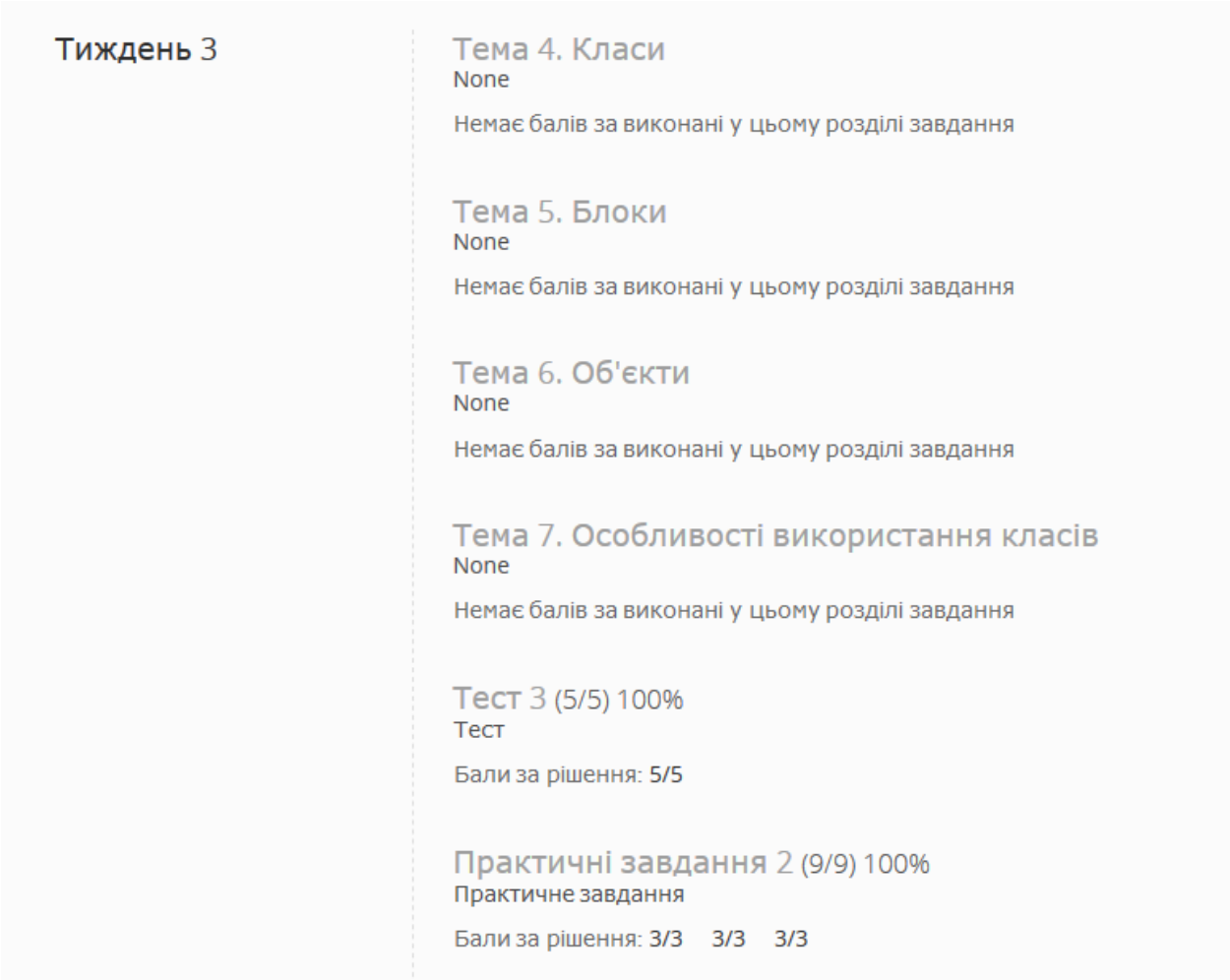
з курсу:

«ООП»

Виконав:
ст. гр. КН-110
Георгій Брусенцов
Прийняв:
Гасько Р.Т

Львів – 2018 р.

1. Тести :



Код:

3.1:

LinkedList.java:

```
package com.tasks3.linkedlist;

public class LinkedList {
   Node head;
   Node node = new Node();

   public LinkedList() {
      head = new Node();
   }

   public void add(Integer data) {
      if (head.getData() == null && head.getNext() == null) {
         head.setData(data);
      } else if (head.getData() != null && head.getNext() == null) {
         head.setNext(node);
         node.setData(data);
      } else {
         while (node.getNext() != null && node.getData() != null) {
```

```
            node = node.getNext();
        }
        Node node_next = new Node();
        node.setNext(node_next);
        node_next.setData(data);
    }
}

public Integer get(int index) {
    Node arr[] = new Node[index + 1];
    arr[0] = head;
    for (int i = 1; i < index + 1; i++) {
        if (arr[i - 1].getNext() != null) {
            if (arr[i - 1] == head) {
                arr[i] = head.getNext();
            } else arr[i] = arr[i - 1].getNext();
        } else return null;
    }
    return arr[index].getData();

}

public boolean delete(int index) {
    int counter = 0;
    if (head.getData() != null) {
        counter++;
        if (head.getNext() == null) {
        } else {
            counter++;
            node = head.getNext();
            while (node.getNext() != null) {
                node = node.getNext();
                counter++;
            }
        }
    }
    int count = 1;
    if (index == 0) {
        head = head.getNext();
        return true;
    } else if (counter < index + 1) {
        return false;
    } else if (index + 1 > 0) {
        Node prev = new Node();
        prev = head;
        //Node nodeDel = new Node();
        node = prev.getNext();
        if (node != null){
            count++;
            while (count < index + 1 && node.getNext() != null){
                prev = node;
                node = node.getNext();
```

```
                count++;
            }
            if (count == index + 1){
                prev.setNext(node.getNext());
                return true;
            } else return false;
        } else return false;
    } else return false;


}

public int size() {
    int counter = 0;
    if (head.getData() != null) {
        counter++;
        if (head.getNext() == null) {
            return counter;
        } else {
            counter++;
            node = head.getNext();
            while (node.getNext() != null) {
                node = node.getNext();
                counter++;
            }
            return counter;
        }
    } else return counter;
}
}
```

Node.java :

```
package com.tasks3.linkedlist;

class Node {
    private Node next;
    private Integer data;

    public Node() {
    }

    public Node getNext() {
        return next;
    }

    public void setNext(Node next) {
        this.next = next;
    }

    public Integer getData() {
        return data;
```

```
    }

    public void setData(Integer data) {
        this.data = data;
    }
}
```

3.2:
Deck.java:

```
package com.tasks3.carddeck;

public class Deck {

    private static final int DECKSIZE = 36;
    private int decksize = 36;

    Card[] deck = new Card[DECKSIZE];

    public Deck(){
        for (int i = 0; i < 4; i++){
            for (int j = 0; j < 9; j++){
                deck[i*9+j] = new Card(Rank.values[j], Suit.values[i]);
            }
        }
    }

    public void shuffle() {
        Card temp = new Card(Rank.ACE, Suit.CLUBS);
        for (int i = 0; i < Math.random()*100; i++){
            int randomNumberA = (int) (Math.random() * DECKSIZE);
            int randomNumberB = (int) (Math.random() * DECKSIZE);
            temp = deck[randomNumberA];
            deck[randomNumberA] = deck[randomNumberB];
            deck[randomNumberB] = temp;
        }
    }

    /* * Впорядкування колоди за мастями та значеннями
     * Порядок сотрування:
     * Спочатку всі карти з мастю HEARTS, потім DIAMONDS, CLUBS, SPADES
     * для кожної масті порядок наступний: Ace,King,Queen,Jack,10,9,8,7,6
     * Наприклад
     * HEARTS Ace
     * HEARTS King
     * HEARTS Queen
     * HEARTS Jack
     * HEARTS 10
     * HEARTS 9
     * HEARTS 8
     * HEARTS 7
     * HEARTS 6
```

```
     * І так далі для DIAMONDS, CLUBS, SPADES */
    public void order() {
        for (int i = 0; i < 4; i++){
            for (int j = 0; j < 9; j++){
                deck[i*9+j] = new Card(Rank.values[j], Suit.values[i]);
            }
        }
    }

    //Повертає true у випадку коли в колоді ще доступні карти
    public boolean hasNext() {
        if (decksize > -1){
            return true;
        }
        else return false;
    }

    //"Виймає" одну карту з колоди, коли буде видано всі 36 карт повертає null
    //Карти виймаються з "вершини" колоди. Наприклад перший виклик видасть
SPADES 6 потім
    //SPADES 7, ..., CLUBS 6, ..., CLUBS Ace і так далі до HEARTS Ace
    public Card drawOne() {
        decksize--;
        if (decksize > -1) {
            return deck[decksize];
        }
        else return null;
    }
}
```

3.3:

```
package com.tasks3.fibonacci;

public class Fibonacci
{
    long first = 1;
    long second = 1;
    long result;
    long iter = 2;

    public long getNumber(int position){
        if (position <= 30 && position > 0) {
            if (position == 1) {
                return first;
            } else if (position == 2) {
                return second;
            } else if (iter != position) {
                result = first + second;
                first = second;
                second = result;
                iter++;
                return getNumber(position);
```

```
            } else return result;
        } else return - 1;
    }
}
```