

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

Кафедра систем штучного інтелекту



**Лабораторна робота №3
з дисципліни
“Об’єктно-орієнтоване програмування”**

**Виконав:
студент групи КН-107
Антоник Анастасія-Марія
Старший викладач:
Гасько Р.Т.**

Львів 2018

3rd week Java course on prometheus tests

Тест 3

(5/5 балів)

1. Що виведе на екран наступний фрагмент коду?

```
class IntsTest {  
    int i;  
}  
  
class Main {  
    public static void main(String args[]) {  
        IntsTest test = new IntsTest();  
        System.out.println(test.i);  
    }  
}
```

☒ 0 ✓

☐ null

☐ Помилка компіляції

☐ Помилка виконання

2. Що з наведено нижче найкраще описує returnType та returnValue:

```
returnType methodName()  
{  
    return returnValue;  
}
```

☐ returnValue повинно бути того ж типу що і returnType.

☒ returnValue повинно бути типу returnType або типом, що є нащадком returnType ✓

☐ Значення returnValue може бути будь якого типу, Java приведе його до returnType

☐ Якщо returnType описано як void то тоді returnValue може бути чим завгодно.

3. Що описує стан об'єкту?

☐ Методи

☒ Поля ✓

☐ Опис класу

4. Клас повинен мати принаймні одне поле:

☐ Так

☒ Ні ✓

5. Що задає поведінку об'єкту?

☒ Методи ✓

☐ Поля

☐ Опис класу

6. Яким чином створюються екземпляри об'єкту?

☐ Використовується ключове слово create

☐ Використовується ключове слово make

☒ Використовується ключове слово new ✓

☐ Використовується ключове слово add

7. Які значення приймуть a та b після виконання main

```
class Test
{
    int a, b;
    Test()
    {
        a = 10;
        b = 20;
    }
}

class Runner
{
    public static void main(String[] args)
    {
        Test obj1 = new Test();
        Test obj2 = obj1;

        obj1.a += 1;
        obj1.b += 1;

        obj2.a += 1;
        obj2.b += 1;
    }
}
```

☐ 11, 21

☒ 12, 22 ✓

☐ 10, 20

☐ 2, 2

8. Що відбудеться якщо Ви будете повертати значення в конструкторі

```
class MyClass
{
    int MyClass()
    {
        return -1;
    }
}
```

- ☐ Помилка компіляції
- ☐ Буде працювати як звичайний конструктор
- ☐ Програма завершиться та поверне -1 як код помилки
- ☒ Це буде звичайний метод, а не конструктор ✓

9. Чи дозволено використовувати this() та super в методах?

```
void test()
{
    this();
    super();
}
```

- ☐ Так

- ☒ Ні ✓

10. Чи буде працювати наступний фрагмент класу

```
public class A
{
    A a = new A();
}
```

- ☐ Так

- ☐ Ні

- ☒ Буде помилка переповнення стеку під час виконання ✓

- ☐ Буде помилка компіляції

11. Чи можливо використовувати локальні змінні без ініціалізації?

- ☐ Так

- ☒ Ні ✓

- ☐ Так, за умови що змінна final

Task 1

```
LinkedList.java Node.java ✖
1 package nodessss;
2
3 class Node{
4     private Node next;
5     private Integer data;
6
7     public Node() {
8     }
9
10    public Node getNext() {
11        return next;
12    }
13    public void setNext(Node next) {
14        this.next = next;
15    }
16    public Integer getData() {
17        return data;
18    }
19    public void setData(Integer data) {
20        this.data = data;
21    }
22 }
```

```
LinkedList.java ✖ Node.java
1 package nodessss;
2
3 public class LinkedList {
4     private Node tail;
5     private Node head;
6     private Node current;
7     private int size=0;
8     public static void main(String[] args) {
9     }
10    public LinkedList() {
11    }
12    public void add(Integer data) {
13        Node one =new Node();
14        one.setData(data);
15        size++;
16        if (head==null) {
17            head=one;
18            tail=one;
19        } else {
20            tail.setNext(one);
21            tail=one;
22        }
23    }
24
25    public Integer get(int index) {
26        Node p=head;
27        for (int i=0;i<index;i++) {
28            p=p.getNext();
29        }
30        return p.getData();
31    }
}
```

```

32 public boolean delete(int index) {
33     Node previous = null;
34     current = head;
35     while (current != null) {
36         if (current.getData().equals(index)) {
37
38             if (previous != null) {
39                 previous.setNext(current.getNext());
40
41                 tail = previous;
42             }
43             else {
44                 head = head.getNext();
45
46                 if (head == null) {
47                     tail = null;
48                 }
49             }
50             size--;
51             return true;
52         }
53         previous = current;
54         current = current.getNext();
55
56     }
57     return false;
58 }
59 public int size() {
60     return size;
61 }
62 }

```

Task 2

Deck.java	Card.java	Rank.java	Suit.java
<pre> 1 package cards; 2 3 public class Card { 4 5 private Rank rank; 6 private Suit suit; 7 8 public Card(Rank rank, Suit suit) { 9 this.rank = rank; 10 this.suit = suit; 11 } 12 13 public Rank getRank() { 14 return rank; 15 } 16 17 public void setRank(Rank rank) { 18 this.rank = rank; 19 } 20 21 public Suit getSuit() { 22 return suit; 23 } 24 25 public void setSuit(Suit suit) { 26 this.suit = suit; 27 } 28 } </pre>	<pre> 1 package cards; 2 3 public class Rank { 4 5 public static final Rank ACE = new Rank("Ace"); 6 public static final Rank KING = new Rank("King"); 7 public static final Rank QUEEN = new Rank("Queen"); 8 public static final Rank JACK = new Rank("Jack"); 9 public static final Rank TEN = new Rank("10"); 10 public static final Rank NINE = new Rank("9"); 11 public static final Rank EIGHT = new Rank("8"); 12 public static final Rank SEVEN = new Rank("7"); 13 public static final Rank SIX = new Rank("6"); 14 15 public static Rank[] values = 16 { ACE, KING, QUEEN, JACK, TEN, NINE, EIGHT, SEVEN, SIX }; 17 18 private String name; 19 20 Rank(String name) { 21 this.name = name; 22 } 23 24 public String getName() { 25 return name; 26 } 27 } </pre>		

```

Deck.java Card.java Rank.java Suit.java
1 package cards;
2
3 public class Suit {
4     public static final Suit HEARTS = new Suit("HEARTS");
5     public static final Suit DIAMONDS = new Suit("DIAMONDS");
6     public static final Suit CLUBS = new Suit("CLUBS");
7     public static final Suit SPADES = new Suit("SPADES");
8
9     public static Suit[] values = { HEARTS, DIAMONDS, CLUBS, SPADES };
10
11     private String name;
12
13     Suit(String name) {
14         this.name = name;
15     }
16
17     public String getName() {
18         return name;
19     }
20
21 }

```

```

Deck.java Card.java Rank.java Suit.java
1 package cards;
2
3 public class Deck {
4     private Card[] deck;
5     private int currentCardNumber;
6     public static void main(String [] args) {
7
8     }
9     public Deck(){
10         deck = new Card[Suit.values.length * Rank.values.length];
11
12         currentCardNumber = deck.length - 1;
13         int k = 0;
14         for(int i = 0; i < Suit.values.length; i++){
15             for(int j = 0; j < Rank.values.length; j++) {
16                 deck[k] = new Card(Rank.values[j], Suit.values[i]);
17                 k++;
18             }
19         }
20     }
21     public void shuffle() {
22         if(currentCardNumber >= 0){
23             Card[] tempDeck = new Card[1];
24             for(int k = 0; k < currentCardNumber + 1; k++){
25                 int a = random(currentCardNumber + 1);
26                 tempDeck[0] = deck[k];
27                 deck[k] = deck[a];
28                 deck[a] = tempDeck[0];
29             }
30         }
31     }
}

```

```

33 public void order() {
34     Card[] tempDeck = new Card[deck.length];
35     int n = 0;
36     for(int i = 0; i < Suit.values.length; i++) {
37         for(int j = 0; j < Rank.values.length; j++) {
38             for(int k = 0; k <= currentCardNumber; k++){
39                 if(deck[k].getSuit().getName().equals(Suit.values[i].getName()) &&
40                     deck[k].getRank().getName().equals(Rank.values[j].getName())) {
41                     tempDeck[n] = deck[k];
42                     n++;
43                 }
44             }
45         }
46     }
47     deck = tempDeck;
48 }
49
50 public boolean hasNext() {
51     return currentCardNumber >= 0;
52 }
53
54 public Card drawOne() {
55     if(hasNext()){
56         return deck[currentCardNumber--];
57     }
58     return null;
59 }
60
61 public static int random(int deckLength){
62     return (int)(Math.random() * deckLength);
63 }
64 }
65

```

ВІРНО

Hide output



Правильно

Правильно

Відповідь:

```

T E S T S
-----
Running CardDeckTest
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.318 sec

Results :

Tests run: 3, Failures: 0, Errors: 0, Skipped: 0

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 11.284s
[INFO] Finished at: Wed Mar 21 19:58:44 EET 2018
[INFO] Final Memory: 15M/158M
[INFO] -----

```

Hide output

Task 3

```
Fibonacci.java ✕
1 package com.tasks3.fibonacci;
2
3 public class Fibonacci
4 {
5     public static long getNumber(int position){
6         if ( position == 1 ) return 1;
7         if(position <= 0) return -1;
8         if (position == 2) return 1;
9         long r = getNumber(position-2) + getNumber(position-1);
10        if (r>0) return r;
11        else return -1;
12    }
13    public Fibonacci(){
14
15
16
17    }
18    public static void main(String[] args) {
19        System.out.print(getNumber(7) + " ");
20    }
21 }
```

ВІРНО



Правильно

Правильно

Відповідь:

```
T E S T S
-----
Running com.task3.fibonacci.FibonacciTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.249 sec

Results :

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 11.815s
[INFO] Finished at: Thu Mar 15 22:27:07 EET 2018
[INFO] Final Memory: 21M/159M
[INFO] -----
```

Висновок: на цій лабораторній роботі я навчилась створювати класи, об'єкти. Вивчила основні принципи роботи з об'єктами. Написала програму, для роботи зі зв'язним списком, програму, для зберігання інформації про колоду карт, а також програму з використанням рекурсивного методу.