

Основы программирования - Java

ФИСТ 1 курс

Власенко Олег Федосович

Лекция 9

ООП.

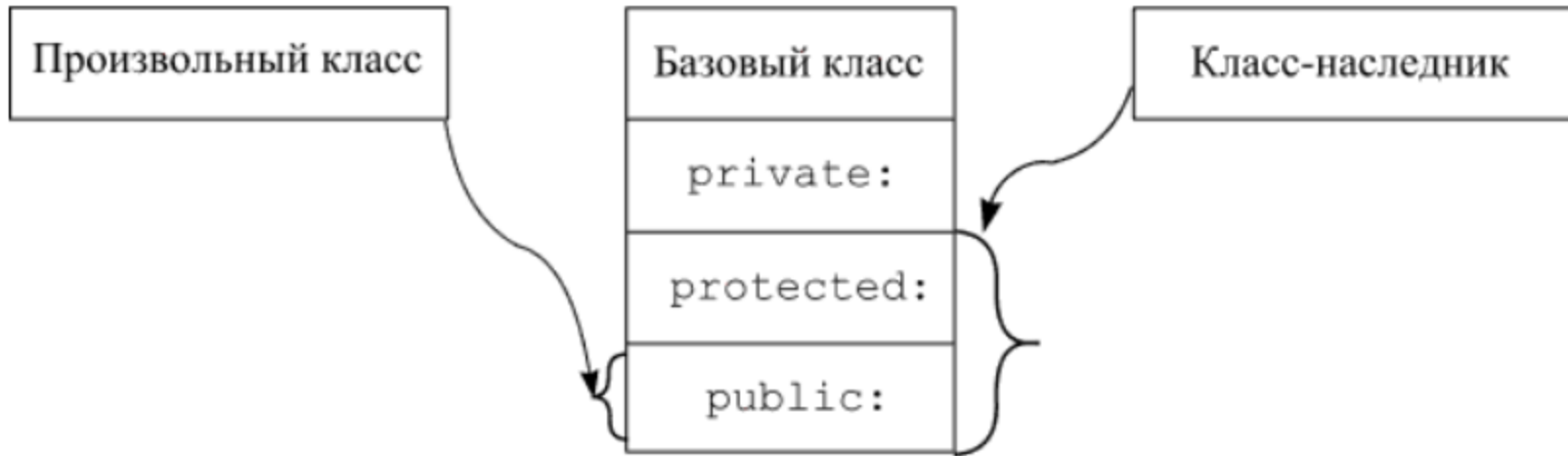
Списки.

Знакомство с unit тестами

Детали ООП в Java

Модификаторы доступа

- **private**: члены класса доступны только внутри класса;
- **«default»** (package-private) (модификатор, по-умолчанию): члены класса видны внутри пакета;
- **protected**: члены класса доступны внутри пакета и в наследниках;
- **public**: члены класса доступны всем;



Модификаторы доступа

- **static** - ссылка этого поля у любого экземпляра класса будет ссылаться на одно и то же значение
- **final** – это модификатор, позволяющий объявлять константные поля в классе.

Элементы класса

```
public class Sample {  
    private int x; // переменная экземпляра класса  
    private int y = 0; // переменная экземпляра класса  
    public final int CURRENT_YEAR = 2012; // константа  
    protected static int bonus; // переменная класса  
    static String version = "Java SE 7"; // переменная класса  
    protected Calendar now;  
    public int method(int z) {  
        return z++;  
    }  
}
```

Примеры использования статических методов

```
public class StaticSamples {  
    public static void main(String[] args) {  
        double phi = Math.PI / 6;  
        System.out.printf("sin(%f)=%f\n", phi, Math.sin(phi));  
        phi = Math.PI / 4;  
        System.out.printf("tan(%f)=%f\n", phi, Math.tan(phi));  
        phi = Math.PI / 2;  
        System.out.printf("tan(%f)=%f\n", phi, Math.tan(phi));  
        System.out.printf("sin(%f)=%f\n", phi, Math.sin(phi));  
        System.out.printf("cos(%f)=%f\n", phi, Math.cos(phi));  
    }  
}
```

Конструкторы

```
public class Quest {  
    // конструктор без параметров (по умолчанию)  
    public Quest() {  
        System.out.println("Вызван конструктор без параметров!!!");  
    }  
    // конструктор с параметрами  
    public Quest(int idc, String txt) {  
        super(); /* вызов конструктора супер класса явным образом  
                   необязателен, компилятор вставит его  
                   автоматически*/  
        System.out.println("Вызван конструктор с параметрами!!!");  
        System.out.println(id + " " + txt);  
    }  
}
```

Порядок инициализации класса

```
public class Department {  
    { System.out.println("logic"); }; //2  
    static { System.out.println("static logic"); } //1  
    private int id = 7;  
    public Department(int d) {  
        id = d;  
        System.out.println("конструктор"); //3  
    }  
    int getId() { return id; }  
    { id = 10; System.out.println("logic"); } //2  
}
```


Абстрактный класс

```
public abstract class AbstractCourse {  
    private String name;  
    public AbstractCourse() {  
    }  
    public abstract void changeTeacher(int id);  
    /*определение метода отсутствует */  
    public void setName(String n) {  
        name = n;  
    }  
}
```

Особенности наследования в Java

```
public class ArrayList<E> extends  
AbstractList<E> implements List<E>,  
RandomAccess, Cloneable, Serializable {  
    ...  
}
```

Внимание: В Java класс наследуется от
ОДНОГО класса, но реализует
произвольное количество интерфейсов.

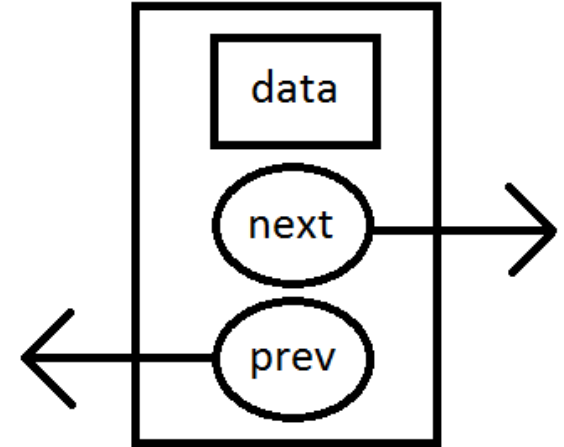
Список на Java

Интерфейс списка

```
public interface IList {  
  
    void insertToHead(int key);  
    void deleteFromHead();  
    int getHeadElement();  
    boolean contains(int key);  
    String toString();  
  
}
```

Класс узла

```
class Node {  
    int key;  
  
    Node next;  
    Node prev; // previous
```



```
    public Node(int key, Node next, Node prev) {  
        this.key = key;  
        this.next = next;  
        this.prev = prev;  
    }  
}
```

Класс списка (1)

```
public class List implements IList {
```

```
    Node head; // first
```

```
    Node tail; // last
```

```
    public List() {
```

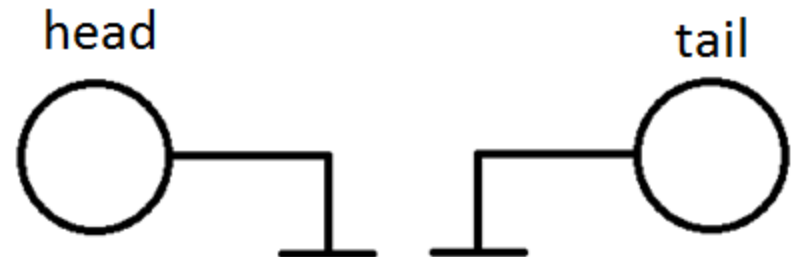
```
        head = new Node(0, null, null);
```

```
        tail = new Node(0, head, head);
```

```
        head.next = tail;
```

```
        head.prev = tail;
```

```
    }
```



Класс списка (2)

@Override

```
public String toString() {
```

```
    String str = "<<";
```

```
    Node p = head.next;
```

```
    while (p != tail) {
```

```
        str = str + p.key + " ";
```

```
        p = p.next;
```

```
    }
```

```
    str = str + ">>";
```

```
    return str;
```

```
}
```

Класс списка (3)

@Override

```
public void insertToHead(int key) {
```

```
    Node p = new Node(key, head.next, head);
```

```
    head.next.prev = p;
```

```
    head.next = p;
```

```
}
```


Класс списка (4)

@Override

```
public void deleteFromHead() {
```

```
    if (head.next == tail) {  
        return;  
    }
```

```
    Node delNext = head.next.next;  
    delNext.prev = head;  
    head.next = delNext;
```

```
}
```

Класс списка (5)

@Override

```
public int getHeadElement() {  
    return head.next.key;  
  
}
```

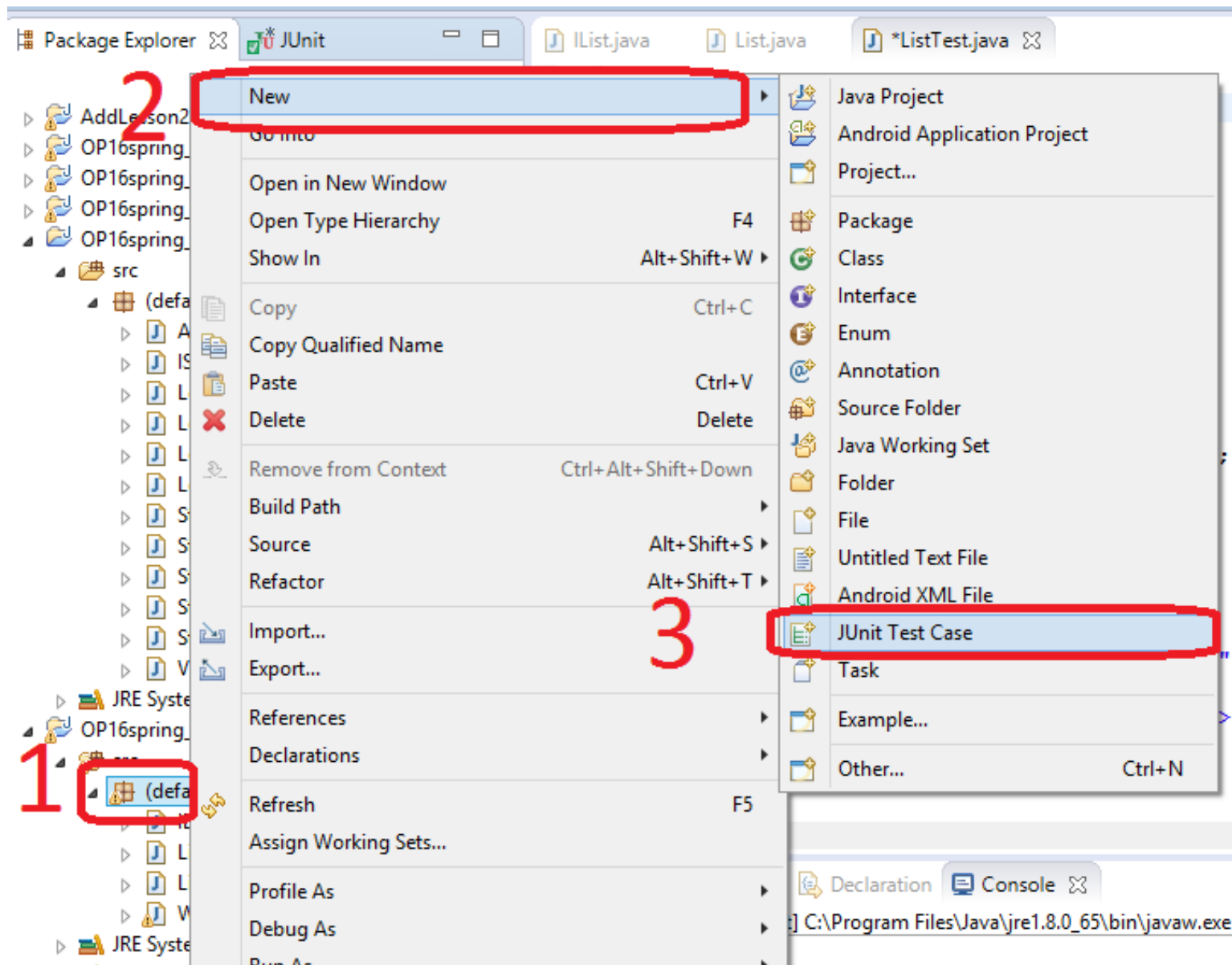
Класс списка (6)

@Override

```
public boolean contains(int key) {  
    Node p = head.next;  
    while (p != tail) {  
        if (p.key == key) {  
            return true;  
        }  
        p = p.next;  
    }  
    return false;  
}
```

```
} // public class List implements IList {
```

Unit тесты



Создание test case в JUnit

New JUnit Test Case

The use of the default package is discouraged.

☐ New JUnit 3 test ☒ New JUnit 4 test

Source folder:

Package:

Name:

Superclass:

Which method stubs would you like to create?

☐ setUpBeforeClass() ☐ tearDownAfterClass()
☐ setUp() ☐ tearDown()
☐ constructor

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Class under test:

Создан test case в JUnit

```
import static org.junit.Assert.*;  
import org.junit.Test;  
  
public class Test1 {  
  
    @Test  
    public void test() {  
        fail("Not yet implemented");  
    }  
  
}
```

Test case для List (1)

```
import static org.junit.Assert.*;  
import org.junit.Test;
```

```
public class ListTest {
```

```
    //      @Test  
    //      public void test() {  
    //          fail("Not yet implemented");  
    //      }  
    @Test  
    public void testToString() {  
        List list = new List();  
        assertEquals(list.toString(), "<<>>");  
    }
```

Test case для List (2)

@Test

```
public void testInsertToHead() {  
    List list = new List();  
    list.insertToHead(1);  
    assertEquals(list.toString(), "<<1 >>");  
    list.insertToHead(2);  
    assertEquals(list.toString(), "<<2 1 >>");  
}
```


Test case для List (3)

@Test

```
public void testDeleteFromHead() {  
    List list = new List();  
    list.insertToHead(1);  
    list.insertToHead(2);  
    assertEquals(list.toString(), "<<2 1 >>");  
    list.deleteFromHead();  
    assertEquals(list.toString(), "<<1 >>");  
}
```

Test case для List (4)

@Test

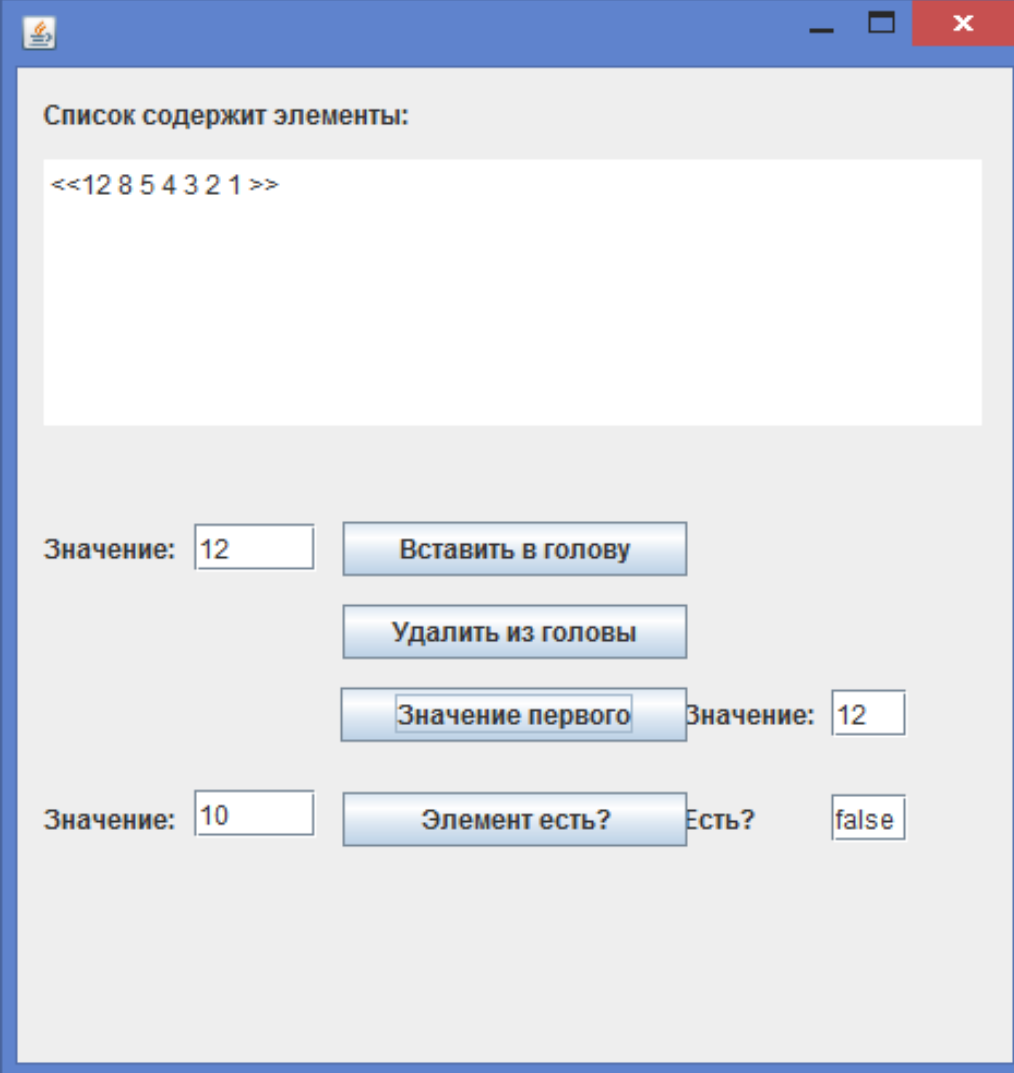
```
public void testGetHeadElement() {  
    List list = new List();  
    list.insertToHead(1);  
    list.insertToHead(2);  
    assertEquals(list.toString(), "<<2 1 >>");  
  
    assertEquals(list.getHeadElement(), 2);  
}
```

Test case для List (5)

@Test

```
public void testContains() {  
    List list = new List();  
    list.insertToHead(1);  
    list.insertToHead(2);  
    assertEquals(list.toString(), "<<2 1 >>");  
  
    assertEquals(list.contains(1), true);  
    assertEquals(list.contains(2), true);  
    assertEquals(list.contains(3), false);  
}  
  
}
```

GUI для проб со списком



Список содержит элементы:

<<12 8 5 4 3 2 1>>

Значение:

Значение:

Значение: Есть?

The image shows a graphical user interface (GUI) for a list-based application. It features a title bar with standard window controls. The main area contains a text box displaying the current list state as '<<12 8 5 4 3 2 1>>'. Below this, there are several interactive elements: a label 'Значение:' followed by a text input field containing '12' and a 'Вставить в голову' (Insert at head) button; a 'Удалить из головы' (Delete from head) button; a 'Значение первого' (First value) button followed by a 'Значение:' label and a text input field containing '12'; and finally, a 'Значение:' label followed by a text input field containing '10', an 'Элемент есть?' (Element exists?) button, and an 'Есть?' label followed by a text input field containing 'false'.

Спасибо за внимание!

Власенко Олег Федосович

E-mail: vlasenko.oleg@gmail.com

Vk: vk.com/oleg.f.vlasenko

Телефон: +7 902 246 05 47