

# Введение в объектно-ориентированное программирование. Классы. Инкапсуляция



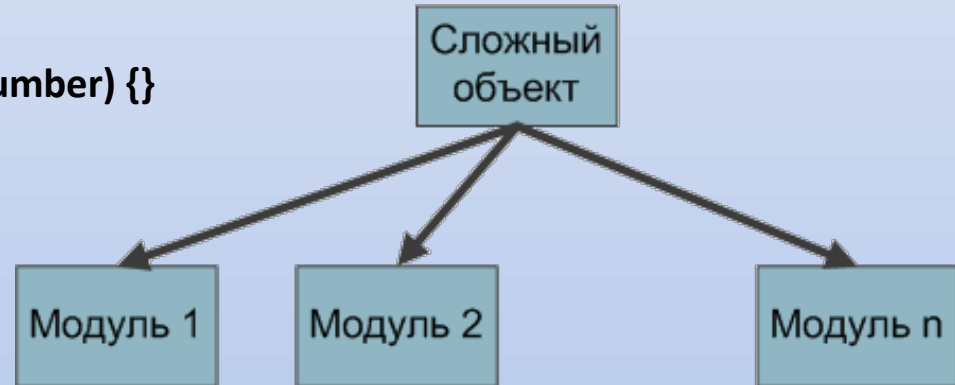
Infosklad.org

Скачивай платные курсы, тренинги и другие материалы - бесплатно!

</> DevStudy.net

# Проблема

```
public class TicTacToe {  
    public static void main(String[] args) {}  
    public static void printGameTable() {}  
    public static boolean handleHumanTurn(int number) {}  
    public static char getWinner(char ch) {}  
    public static boolean isCellFree(int number) {}  
    public static int readHumanTurn() {}  
    public static boolean isDraw() {}  
    public static void makeComputerTurn() {}  
    public static void makeRandomComputerTurn() {}  
    public static void makeTurn(int number, char ch) {}  
    public static boolean tryWin(char ch) {}  
    public static boolean trySetToRow(int row, char ch) {}  
    public static boolean trySetToCol(int col, char ch) {}  
    public static boolean trySetToLeftTopDiagonal(char ch) {}  
    public static boolean trySetToRightTopDiagonal(char ch) {}  
    public static int[][] getVariants(int i, int j) {}  
}
```



Infoklad.org

Скачивай платные курсы, тренинги и другие материалы - бесплатно!

</> DevStudy.net

# Проблема

```
public class Test {  
  
    public static void main(String[] args) {  
        String firstName1 = "Ivan";  
        String lastName1 = "Ivanov";  
        int age1 = 22;  
  
        String firstName2 = "Petr";  
        String lastName2 = "Petrov";  
        int age2 = 21;  
  
        String firstName3 = "Sergey";  
        String lastName3 = "Sergeev";  
        int age3 = 19;  
    }  
}
```



Infosklad.org

Скачивай платные курсы, тренинги и другие материалы - бесплатно!

</> DevStudy.net

# Определение

- **Объектно-ориентированное программирование (ООП)** — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования

# Класс

- **Класс** - универсальный, комплексный тип данных, состоящий из тематически единого набора «полей» (переменных более элементарных типов) и «методов» (функций для работы с этими полями), то есть он является моделью информационной сущности с внутренним и внешним интерфейсами для оперирования своим содержимым.
- Класс – ссылочный тип данных;
- Переменная данного типа данных называется экземпляром (объектом) данного класса.

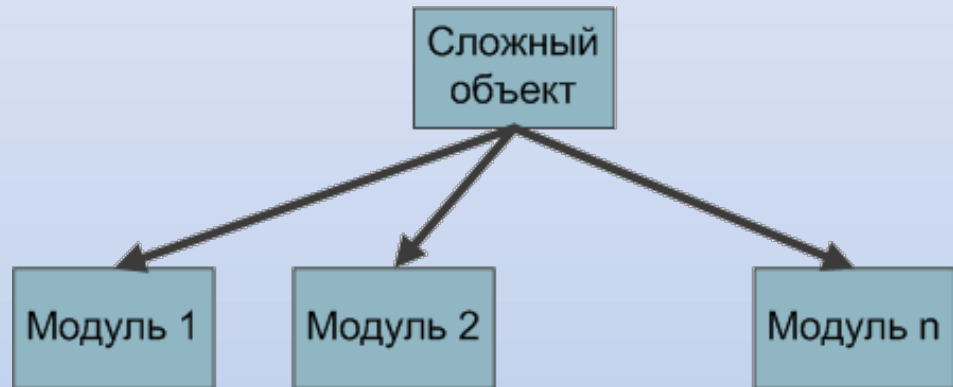
# Классы игры TicTacToe

```
public class GameTable {  
    public void printGameTable() {  
    }  
    public void isCellFree(int number) {  
    }  
}
```

```
public class WinnerChecker {  
    public void getWinner(char ch) {  
    }  
    public void isDraw() {  
    }  
}
```

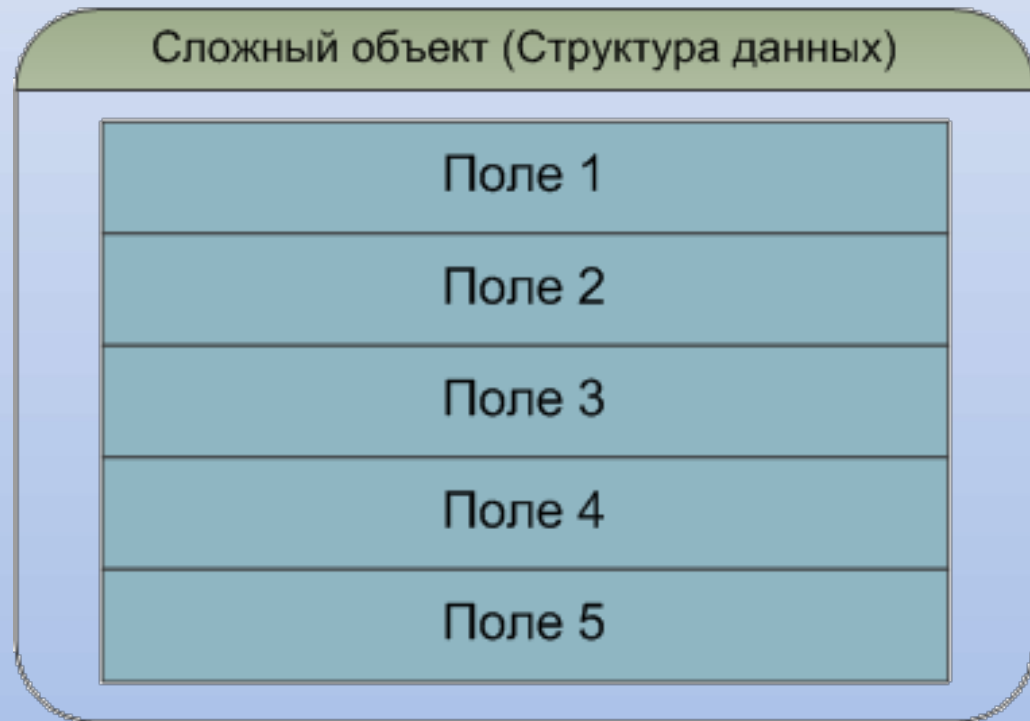
```
public class HumanTurn {  
    public static void readHumanTurn() {  
    }  
    public void make Turn() {  
    }  
}
```

```
public class ComputerTurn {  
    public void makeTurn() {  
    }  
    public void tryWin(char ch) {  
    }  
    public void trySetToRow(int row, char ch) {  
    }  
    public void trySetToCol(int col, char ch) {  
    }  
    public void trySetToLeftTopDiagonal(char ch) {  
    }  
    public void trySetToRightTopDiagonal(char ch) {  
    }  
    public void getVariants(int i, int j) {  
    }  
    public void makeRandomComputerTurn() {  
    }  
}
```



# Класс Student

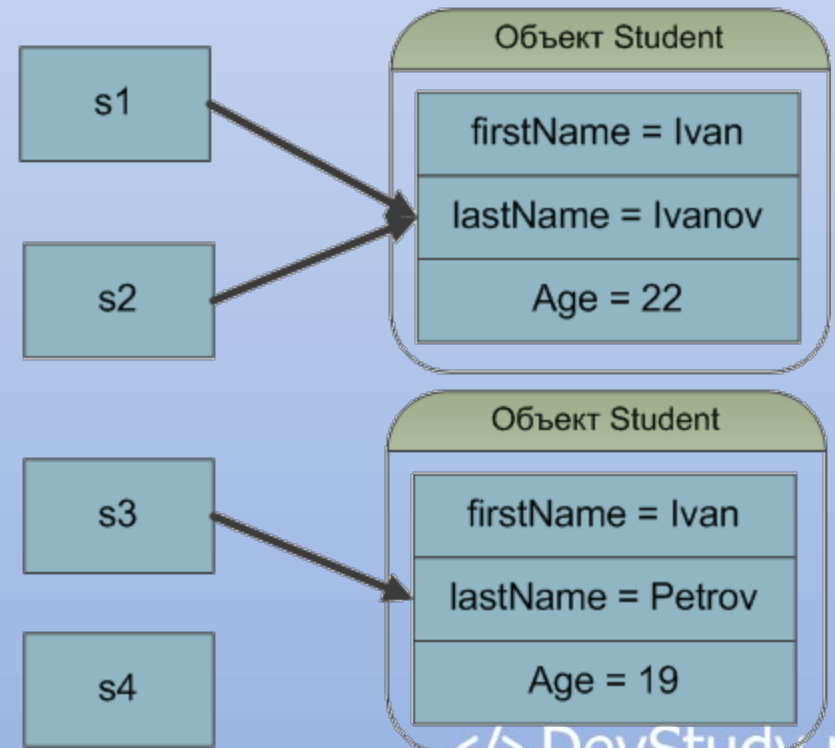
```
public class Student {  
    String firstName;  
    String lastName;  
    int age;  
}  
  
public class Test2 {  
    public static void main(String[] args) {  
        Student s1 = new Student();  
        s1.firstName = "Ivan";  
        s1.lastName = "Ivanov";  
        s1.age = 22;  
  
        Student s2 = new Student();  
        s2.firstName = "Petr";  
        s2.lastName = "Petrov";  
        s2.age = 21;  
  
        Student s3 = new Student();  
        s3.firstName = "Sergey";  
        s3.lastName = "Sergeev";  
        s3.age = 19;  
    }  
}
```



# Объект, Ссылка, new и null, GC

```
public class Test3 {  
    public static void main(String[] args) {  
        Student s1 = new Student();  
        s1.firstName = "Ivan";  
        s1.lastName = "Ivanov";  
        s1.age = 22;  
  
        Student s2 = s1;  
  
        Student s3 = new Student();  
        s3.firstName = s2.firstName;  
        s3.lastName = s1.lastName;  
        s3.age = 19;  
  
        Student s4 = null;  
    }  
}
```

s1	Student (id=19)
age	22
firstName	"Ivan" (id=22)
lastName	"Ivanov" (id=27)
s2	Student (id=19)
age	22
firstName	"Ivan" (id=22)
lastName	"Ivanov" (id=27)
s3	Student (id=21)
age	19
firstName	"Ivan" (id=22)
lastName	"Petrov" (id=26)
s4	null




















# Значения по умолчанию

```
public class DefaultTest {
    Student student;
    String parameterName;
    int intValue;
    double doubleValue;
    boolean booleanValue;
    int[] intArray;
    Student[] studentArray;
    boolean[] booleanArray;
}

public class DefaultTestMain {
    public static void main(String[] args) {
        DefaultTest a;
        DefaultTest t = null;
        DefaultTest b = new DefaultTest();
        int A;
        int[] arr1;
        int[] arr2 = new int[3];
        A = 0;
        System.out.println();
    }
}
```

 t	null
 b	DefaultTest (id=19)
 booleanArray	null
 booleanValue	false
 doubleValue	0.0
 intArray	null
 intValue	0
 parameterName	null
 student	null
 studentArray	null
 A	0
 arr2	(id=21)
 [0]	0
 [1]	0
 [2]	0

# Какой размер массива?

```
public class FindNumbers {  
    public static void main(String[] args) {  
        String text = "assdsew12345ht 67 jhhggv 9 mknj ,jn ljkh 4 ooki 0 6666";  
        System.out.println(Arrays.toString(find(text)));  
    }  
    public static int [] find(String text) {  
        int [] numbers = new int[???????];  
        return res;  
    }  
}
```

# Динамический массив

```
public class DynaArray {
    int[] array;
    int size;
    void add(int element) {
        if(array == null) {
            clear();
        } else if (size == array.length) {
            int[] temp = array;
            array = new int[temp.length * 2];
            for(int i=0;i<temp.length;i++){
                array[i] = temp[i];
            }
        }
        array[size++] = element;
    }
    int get(int index) {
        return array[index];
    }
    int size(){
        return size;
    }
    void clear(){
        size = 0;
        array = new int[10];
    }
    int[] toArray(){
        return Arrays.copyOf(array, size);
    }
}
```

```
public class DynaArrayTest {

    public static void main(String[] args) {
        DynaArray arr = new DynaArray();

        for(int i=0;i<100;i++){
            arr.add(i);
        }

        System.out.println(Arrays.toString(arr.toArray()));
    }
}
```

# Важность инкапсуляции

// ЛОМАЕМ ДИНАМИЧЕСКИЙ КЛАСС

```
public class DynaArrayTest {  
  
    public static void main(String[] args) {  
        DynaArray arr = new DynaArray();  
  
        for(int i=0;i<100;i++){  
            arr.array[i] = i;  
        }  
  
        System.out.println(Arrays.toString(arr.toArray()));  
    }  
}
```

Модификаторы	Тот же класс	Тот же пакет	Подкласс	Другие пакеты
<b>public</b>	ДА	ДА	ДА	ДА
<b>protected</b>	ДА	ДА	ДА	НЕТ
<b>default(package)</b>	ДА	ДА	НЕТ	НЕТ
<b>private</b>	ДА	НЕТ	НЕТ	НЕТ

# Защищаем DynaArray

```
public class DynaArray {  
    private int[] array;  
    private int size;  
    public void add(int element) {  
        if(array == null) {  
            clear();  
        } else if (size == array.length) {  
            int[] temp = array;  
            array = new int[temp.length * 2];  
            for(int i=0;i<temp.length;i++){  
                array[i] = temp[i];  
            }  
        }  
        array[size++] = element;  
    }  
    public int get(int index) {  
        return array[index];  
    }  
    public int size(){  
        return size;  
    }  
    public void clear(){  
        size = 0;  
        array = new int[10];  
    }  
    public int[] toArray(){  
        return Arrays.copyOf(array, size);  
    }  
}
```

# Конструкторы

```
public class Student {  
    String firstName;  
    String lastName;  
    int age;  
    public Student(String first, String last, int _age) {  
        firstName = first;  
        lastName = last;  
        age = _age;  
    }  
    public Student(String first, String last) {  
        firstName = first;  
        lastName = last;  
    }  
    public Student() {  
  
    }  
}
```

▲ s1	Student (id=18)
▲ age	22
▶ ▲ firstName	"Ivan" (id=30)
▶ ▲ lastName	"Ivanov" (id=31)
▲ s2	Student (id=21)
▲ age	21
▶ ▲ firstName	"Petr" (id=28)
▶ ▲ lastName	"Petrov" (id=29)
▲ s3	Student (id=22)
▲ age	0
▶ ▲ firstName	"Sergey" (id=23)
▶ ▲ lastName	"Sergeev" (id=27)

```
public class StudentTest{
```

```
    public static void main(String[] args) {
```

```
        Student s1 = new Student();  
        s1.firstName = "Ivan";  
        s1.lastName = "Ivanov";  
        s1.age = 22;
```

```
        Student s2 = new Student("Petr", "Petrov", 21);
```

```
        Student s3 = new Student("Sergey", "Sergeev");  
        s3.age = 19;
```

```
    }
```

```
}
```

1. Конструктор – это специальный метод класса, который вызывается при создании объекта.
2. Конструктор без параметров называется конструктором по умолчанию (default constructor).
3. Если в классе нет ни одного конструктора, то генерируется пустой конструктор по умолчанию. Если в классе есть хотя бы один конструктор, то конструктор по умолчанию не генерируется

# Задание начальных значений

```
public class DynaArray {  
    private int[] array = new int[10];  
    private int size;  
    public void add(int element) {  
        if (size == array.length) {  
            int[] temp = array;  
            array = new int[temp.length * 2];  
            for(int i=0;i<temp.length;i++){  
                array[i] = temp[i];  
            }  
        }  
        array[size++] = element;  
    }  
    public int get(int index) {  
        return array[index];  
    }  
    public int size(){  
        return size;  
    }  
    public void clear(){  
        size = 0;  
        array = new int[10];  
    }  
    public int[] toArray(){  
        return Arrays.copyOf(array, size);  
    }  
}
```

```
public class DynaArray {  
    private int[] array;  
    private int size;  
    public DynaArray(){  
        array = new int[10];  
    }  
    public void add(int element) {  
        if (size == array.length) {  
            int[] temp = array;  
            array = new int[temp.length * 2];  
            for(int i=0;i<temp.length;i++){  
                array[i] = temp[i];  
            }  
        }  
        array[size++] = element;  
    }  
    public int get(int index) {  
        return array[index];  
    }  
    public int size(){  
        return size;  
    }  
    public void clear(){  
        size = 0;  
        array = new int[10];  
    }  
    public int[] toArray(){  
        return Arrays.copyOf(array, size);  
    }  
}
```

# this

























```
public class Student {  
    String firstName;  
    String lastName;  
    int age;  
    public Student(String firstName, String lastName, int age) {  
        this.firstName = firstName;  
        this.lastName = lastName;  
        this.age = age;  
    }  
    public Student(String firstName, String lastName) {  
        this(firstName, lastName, 0);  
    }  
    public Student() {  
        this(null, null);  
    }  
}
```



# Защищаем студента

```
public class Student {
    private String firstName;
    private String lastName;
    private int age;
    public Student(String firstName, String lastName, int age) {
        this.setFirstName(firstName);
        setLastName(lastName);
        setAge(age);
    }
    public Student() {}
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        if(firstName.length() > 1) {
            this.firstName = firstName.substring(0, 1).toUpperCase() + firstName.substring(1).toLowerCase();
        } else {this.firstName = firstName.toUpperCase();}
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        if(lastName.length() > 1) {
            this.lastName = Character.toUpperCase(lastName.charAt(0)) + lastName.substring(1).toLowerCase();
        } else {this.lastName = lastName;}
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age > 17 && age < 50 ? age : 18;
    }
    public String getFullName() {
        return getLastName() + " " + getFirstName();
    }
}
```

```
public class StudentTest {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.setFirstName("ivan");
        s1.setLastName("IVANOV");
        s1.setAge(5);
        System.out.println(s1.getFullName() + ", " + s1.getAge());
        Student s2 = new Student("PETR", "petrov", 21);
        Student s3 = new Student("Sergey", "Sergeev", 0);
        s3.setAge(19);
        System.out.println(s2.getFullName() + ", " + s2.getAge());
        System.out.println(s3.getFullName() + ", " + s3.getAge());
    }
}
```

  s1	Student (id=19)
  age	18
  firstName	"Ivan" (id=23)
  lastName	"Ivanov" (id=27)
  s2	Student (id=21)
  age	21
  firstName	"Petr" (id=28)
  lastName	"Petrov" (id=29)
  s3	Student (id=22)
  age	19
  firstName	"Sergey" (id=30)
  lastName	"Sergeev" (id=31)

# Соглашение об именовании

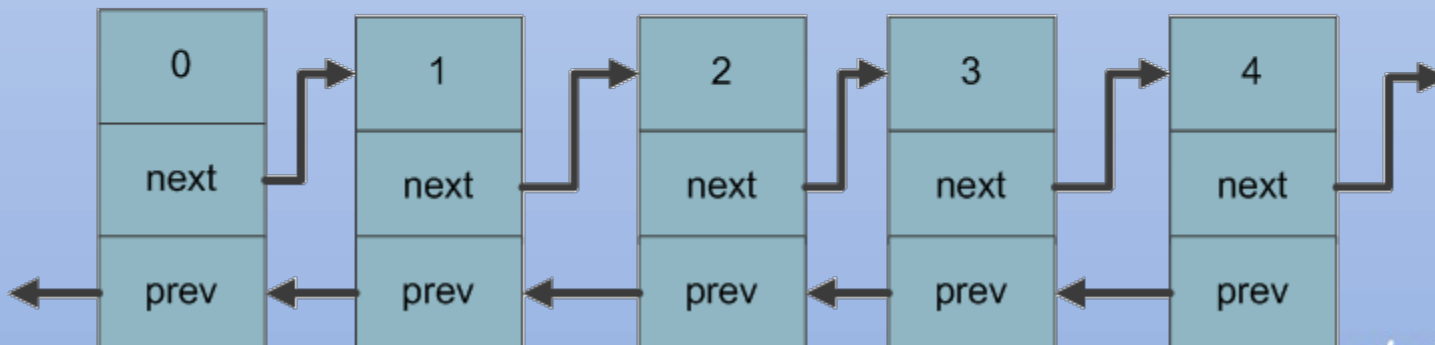
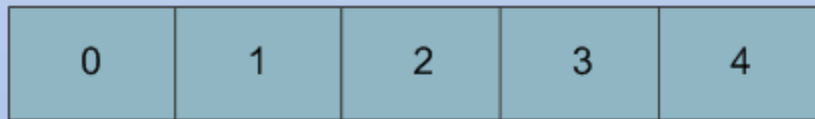
1. Имена классов должны всегда начинаться с большой буквы (например, **ConnectionFactory**);
2. Имена пакетов должны состоять только из букв в нижнем регистре и цифр. При этом каждая составляющая имени должна начинаться с буквы (например, **net.devstudy.httpserver**);
3. Обычно имена пакетов начинаются с инвертированного имени домена компании-разработчика. Так из приведенного выше примера видно, что разработчиком проекта `httpserver` является `devstudy.net`;
4. Названия методов и переменных должны начинаться с маленькой буквы и быть осмысленными. Каждое новое слово должно начинаться с большой буквы. Подчеркивания отсутствуют. (например, **insertToDatabase()**, **value**, **personName**);
5. Названия констант состоят из больших букв, цифр и знаков подчеркивания в качестве разделителей между словами **MAX\_INTEGER**;

# Выводы

1. **Объектно-ориентированное программирование** позволяет разбивать сложную программу на модули, декомпозируя ее с помощью классов, которые моделируют объекты реального мира;
2. Класс - универсальный, комплексный тип данных, состоящий из тематически единого набора «полей» (переменных более элементарных типов) и «методов» (функций для работы с этими полями);
3. Класс – ссылочный тип данных;
4. Переменная данного типа данных называется экземпляром (объектом) данного класса;
5. `null` – специальная константа, которая указывает на то, что не существует объекта, на который указывает данная ссылка;
6. Создать объект в Java можно **только** с помощью оператора **new**. Только при использовании **new** выделяется память под объект класса;
7. По умолчанию ссылочные переменные класса равны **null**, примитивы 0 и **false** соответственно;
8. **Принцип инкапсуляции** – принцип сокрытия данных, позволяет защищать внутренние переменные класса от внешнего неконтролируемого воздействия;
9. Для инициализации переменных класса можно использовать конструкторы или явную инициализацию;
10. Конструктор – это специальный метод класса, который вызывается при создании объекта;
11. Конструктор по умолчанию генерируется автоматически, если нет других явных конструкторов;
12. С помощью ключевого слова **this** можно получить ссылку на текущий объект, и с ее помощью обращаться к переменным класса, методам и вызывать конструкторы;
13. Стандартным шаблоном при создании классов предметной области в Java является использование `getter & setter` даже без дополнительных проверок. Такие классы в Java иногда называют `bean` или `POJO`.
14. Соглашения об именовании позволяют определить правила написания чистого кода.

# Домашнее задание

- 1) Добавить метод **`int remove(int index)`**, который удалит элемент из динамического массива по индексу и вернет удаленный элемент;
- 2) Двусвязный список – структура данных, которая позволяет сохранять набор элементов подобно массиву, но при этом не требует соблюдения порядка расположения элементов в памяти компьютера:



# Домашнее задание

## 3) Реализовать класс LinkedList (+LinkedListTest):

```
public class Item {  
    private Item next;  
    private int value;  
    private Item previous;  
    Item(int value) {  
        this.value = value;  
    }  
    Item getNext() {  
        return next;  
    }  
    void setNext(Item next) {  
        this.next = next;  
    }  
    int getValue() {  
        return value;  
    }  
    Item getPrevious() {  
        return previous;  
    }  
    void setPrevious(Item previous) {  
        this.previous = previous;  
    }  
}
```

```
public class LinkedList {  
    private Item first;  
    private Item last;  
    private int size;  
    public void add(int element) {  
    }  
    public int get(int index) {  
        return 0;  
    }  
    public int remove(int index) {  
        return 0;  
    }  
    public int size(){  
        return size;  
    }  
    public void clear(){  
        size = 0;  
        first = last = null;  
    }  
    public int[] toArray(){  
        return null;  
    }  
}
```

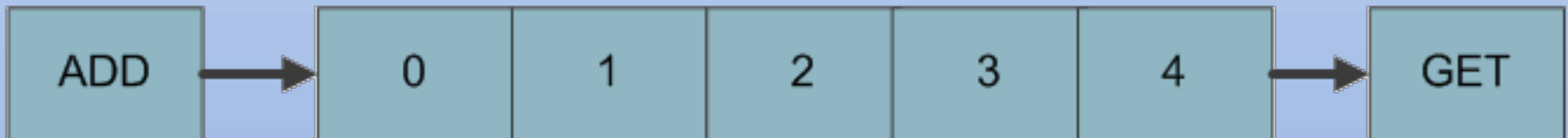
# Домашнее задание

## 4) Реализовать класс очереди Queue:

```
public class Queue {  
    public void add(int element) {  
  
    }  
    public int get(){  
        return 0;  
    }  
    public int size(){  
        return 0;  
    }  
}
```

```
public class QueueTest {  
    public static void main(String[] args) {  
        Queue q = new Queue();  
        for (int i = 0; i < 5; i++) {  
            q.add(i);  
        }  
        while (q.size() > 0) {  
            System.out.print(q.get() + " ");  
        }  
        System.out.println();  
    }  
}
```

RESULT: 0 1 2 3 4



# Домашнее задание

## 5) Реализовать класс стэка Stack:

```
public class Stack {  
    public void add(int element) {  
    }  
    public int get(){  
        return 0;  
    }  
    public int size(){  
        return 0;  
    }  
}
```

```
public class StackTest {  
    public static void main(String[] args) {  
        Stack s = new Stack ();  
        for (int i = 0; i < 5; i++) {  
            s.add(i);  
        }  
        while (s.size() > 0) {  
            System.out.print(s.get() + " ");  
        }  
        System.out.println();  
    }  
}
```

RESULT: 4 3 2 1 0

